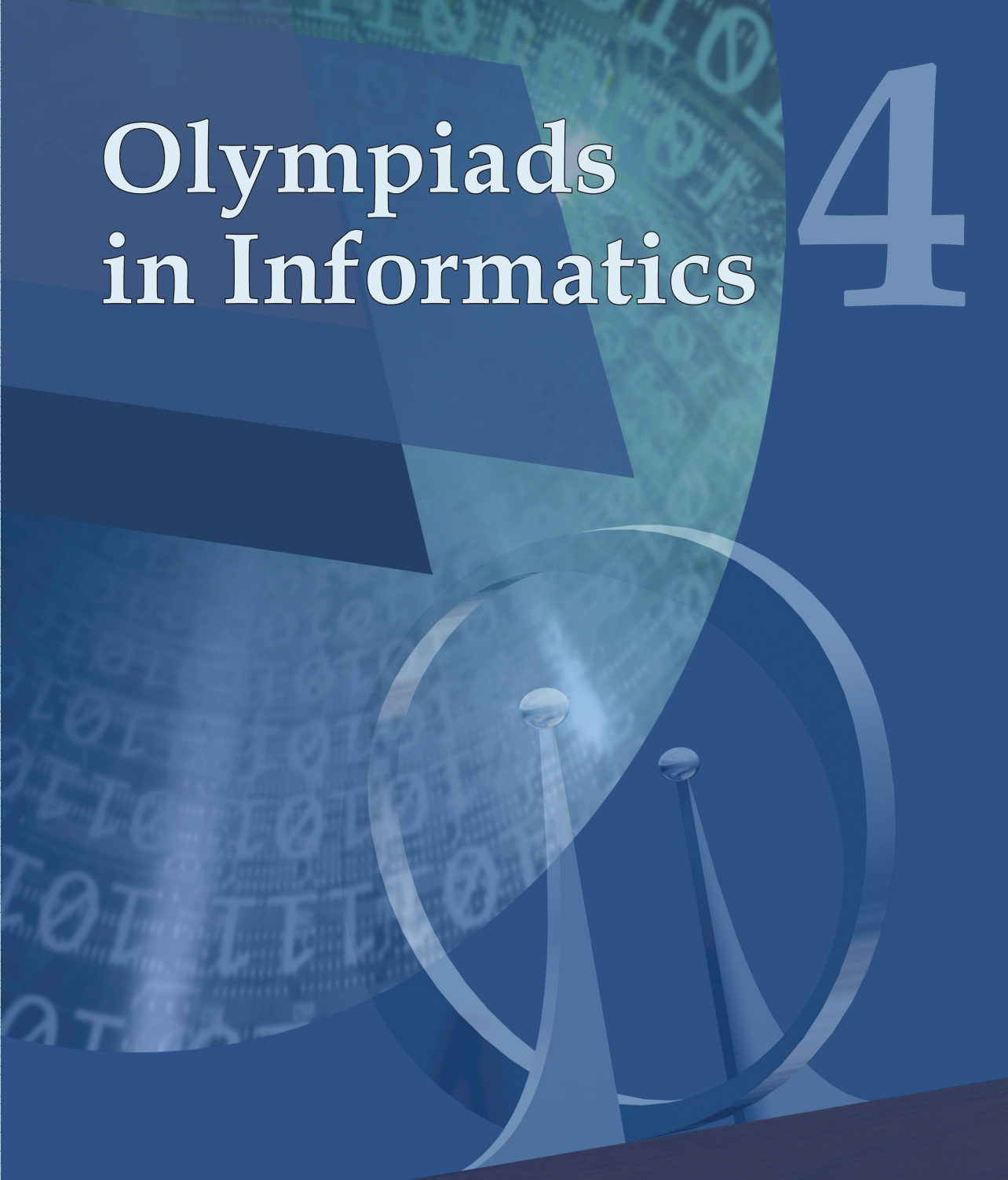
Olympiads in Informatics 4



IOI

International Olympiad in Informatics

ISSN 1822-7732

INTERNATIONAL OLYMPIAD IN INFORMATICS
INSTITUTE OF MATHEMATICS AND INFORMATICS
INTERNATIONAL FEDERATION FOR INFORMATION PROCESSING

OLYMPIADS IN INFORMATICS

Volume 4 2010

Selected papers of
the International Conference joint with
the XXII International Olympiad in Informatics
Waterloo, Canada, August 14–21, 2010

OLYMPIADS IN INFORMATICS

ISSN 1822-7732

Editor-in-Chief

Valentina Dagienė

Institute of Mathematics and Informatics, Lithuania, dagiene@ktl.mii.lt

Executive Editor

Richard Forster

British Informatics Olympiad, UK, forster@olympiad.org.uk

International Editorial Board

Gerald Futschek, Vienna University of Technology, Austria, futschek@ifs.tuwien.ac.at

Bruria Haberman, Holon Institute of Technology, Israel, Bruria.Haberman@weizmann.ac.il

Marcin Kubica, Warsaw University, Poland, kubica@mimuw.edu.pl

Ville Leppänen, University of Turku, Finland, villelep@cs.utu.fi

Krassimir Manev, Sofia University, Bulgaria, manev@fmi.uni-sofia.bg

Fredrik Niemelä, KTH University, Sweden, niemela@csc.kth.se

Rein Prank, University of Tartu, Estonia, rein.prank@ut.ee

Miguel A. Revila Ramos, University of Valladolid, Spain, revilla@mac.cie.uva.es

Peter Taylor, University of Canberra, Australia, pjt@olympiad.org

Troy Vasiga, University of Waterloo, Canada, tmjvasiga@cs.uwaterloo.ca

Peter Waker, International Qualification Alliance, Republic of South Africa,
waker@interware.co.za

http://www.mii.lt/olympiads_in_informatics

Publisher: Institute of Mathematics and Informatics
Akademijos 4, LT-08663 Vilnius, Lithuania

© Institute of Mathematics and Informatics, 2010
All rights reseserved

Olympiads in Informatics, 2010, Vol. 4
© 2010 *Institute of Mathematics and Informatics, Vilnius*

Foreword

Informatics and technology in general are curious endeavours in which to be involved. Working, researching or teaching within them, on a day to day basis, they seem to move quite slowly. Step back however and look a few years in the past and the pace of change can be staggering. And not just recent memory. It is a fair bet that this has been true for everyone reading this, whatever their age, country or time in their life when the statement might have been made.

Education also has its trends and its technologies. The internet, email and technology in general have brought new means of communication and access to information. The teaching of informatics has changed as new technologies are introduced, but so has the teaching of other subjects. Computers are no longer just for the specialist – they are basic tools.

The papers in this volume cover a range of topics. Some are specific to informatics but some are about education in general. Not just the way we are influencing it, but how it is influencing us. Government policies to educate all teachers and students, if not entire populaces, in what was not long ago not experienced by most. There are papers considering alternative ways of teaching informatics and getting students thinking about algorithms.

Papers on tasks and evaluation, looking at how (and indeed from where) we create quality tasks with a pedagogical benefit. Papers on communication and the sharing of resources. We are fortunate to have an enthusiastic, if small, community.

The recent IOI workshop is briefly reported in this volume and longer papers will follow in the future.

The editorial board has been expanded and by bringing in individuals from outside the IOI we intend to enlarge the scope of the journal. Small steps perhaps, but step back in a few years time...

As always thanks are due to all those who have assisted with the current volume – authors, reviewers and editors – and it would be remise not to thank those involved in arranging the workshop and encouraging research.

Particular thanks are due to the organisational committee for IOI'2010 in Canada without whose assistance we would be unable to hold the conference. Their assistance, during what is an already busy period, is gratefully received.

Editors

Olympiads in Informatics, 2010, Vol. 4, 3–14

© 2010 *Institute of Mathematics and Informatics, Vilnius*

Encouraging Algorithmic Thinking Without a Computer

Benjamin A. BURTON
School of Mathematics and Physics, The University of Queensland
Brisbane QLD 4072, Australia
e-mail: bab@maths.uq.edu.au

Abstract. At the secondary school level, traditional programming competitions remain inaccessible to the vast majority of students. We describe the Australian Informatics Competition (AIC), a pen-and-paper event that is accessible to a much broader audience but still retains a core focus on algorithms. In addition to multiple choice questions, a unique feature of the AIC is its three-stage tasks that invite algorithmic thinking by posing similar problems of increasing size. In this paper we describe the AIC, the design decisions behind it, and the types of problems that it contains.

Key words: multiple choice contests, algorithmic tasks, three-stage tasks.

1. Introduction

Competitions and enrichment programmes in computer science have enjoyed a sharp rise in popularity over recent decades. Consider for instance the International Olympiad in Informatics (IOI) – although much younger than many of the science olympiads, the IOI has grown to become the second largest of these international events.[1] Non-competitive programmes have also enjoyed an enthusiastic reception, such as Australia's National Computer Science School (`http://www.ncss.edu.au/`).

Despite this popularity, many national informatics olympiad programmes struggle to find students at the secondary school level (Anido and Menderico, 2007; Boersen and Phillipps, 2006; Choijoovanchig *et al.*, 2007; Pohl, 2007). In Australia the numbers are striking – the national entry-level mathematics competition enjoys several hundred thousand participants each year, whereas the entry-level programming competition for the informatics olympiad attracts just one or two hundred.

Several factors contribute to these extremely low rates of entry for national programming competitions:

(i) *Curriculum:* In many countries, computer programming and algorithm design receives very little attention in the secondary school curriculum (Verhoeff, 2009). In comparison, mathematics and other science olympiad disciplines (such as biology, chemistry and physics) are well-taught and widely studied. As a result, there

[1] Measured by the number of attending countries in 2009.

are far fewer students with the necessary skills for a programming competition; moreover, it is difficult for teachers to identify who these students are.

(ii) *Technology:* Programming contests require students to have dedicated access to a computer in exam conditions, often for several hours during the school day. This can make scheduling and supervision difficult for teachers, and (depending on school resources) can severely limit the number of entrants from any given school.

(iii) *Grading:* For programming contests that employ traditional computer-based grading, a student cannot score any points at all if they cannot produce a working program that reads from an input file, writes to an output file and does some work in between. This is a significant barrier to scoring, particularly given the poor curriculum support mentioned above. The result is often a large number of low or zero scores, discouraging inexperienced students from participating and disheartening those who do.

If we are to make informatics competitions accessible to a significantly broader audience, we should strive to address all three of these difficulties. A natural solution is a pen-and-paper contest, with multiple choice or short answer tasks that are quick to solve (whether correctly or incorrectly), highly approachable without any prior knowledge, and ranging in difficulty from challenging to very easy.

Several countries have adopted such solutions in recent years, For example, South Africa has offered a pen-and-paper contest since 2003 with high rates of participation (Merry *et al.*, 2008), and Australia introduced the pen-and-paper Australian Informatics Competition in 2005 (Clark, 2006). Lithuania introduced the Bebras (Beaver) contest in 2004 (Dagienė, 2006); this contest depends on computers but in a much more accessible way, and it has since spread to several European countries. Predating any of these events, Bulgaria has included informatics problems in their multiple choice mathematics contest Chernorizets Hrabar since 1992 (Tabov *et al.*, 2003).

The greatest difficulty with such a format is retaining the focus on algorithms and algorithmic thinking. Particularly with multiple choice, it is challenging to find questions that do not rely on pre-assumed knowledge (such as programming languages or pseudocode), but which nevertheless have more of an algorithmic flavour than traditional mathematics problems and puzzles.

In this paper we describe how Australia has responded to these challenges through the Australian Informatics Competition (AIC). Held annually since 2005, the AIC has a clear focus on algorithms, yet maintains an informal pen-and-paper setting with no required knowledge. The contest employs a mix of multiple choice and integer answers, and incorporates unique "three-stage tasks" that encourage students to develop informal algorithms by posing similar problems of increasing size.

In Section 2 we outline the structure and design of the AIC. Section 3 describes how the AIC maintains its algorithmic focus in this informal setting, and offers examples of different question types including the three-stage tasks mentioned above. The concluding notes in Section 4 include statistics that show how well the AIC has been received.

2. The Australian Informatics Competition

The AIC is offered in three divisions that span all of secondary school (years 7–12 in Australia). Each division consists of 15 questions: six multiple choice questions requiring a single letter from A to E, and nine integer answer questions requiring a single integer from 0 to 999. The integer answers are used for the three-stage tasks (described in the following section), and also serve to limit the points that students can achieve through guessing (as discussed below).

The contest is designed to minimise the burden on both teachers and students:

- For teachers, the contest is easy to administer. It runs for just one hour, and students can sit the contest at their desks with nothing more than pencil and paper (calculators are optional but unnecessary). This make it possible for a teacher to give the contest to an entire class during a single lesson.
- For students, the contest is designed to be accessible and engaging regardless of their academic background. Questions are posed as puzzles in a real-world or fantasy setting, which students can approach simply by scribbling and following their intuition. Although the questions aim to stimulate algorithmic thinking, they do not rely on any knowledge of programming or computing, and they do not involve code or pseudocode.
- The simple format of the contest (A–E and 0–999) further reduces the burden on students, and makes the contest easy to grade for a large number of participants. Students record their solutions by colouring circles in pencil on a specially designed "mark sense" answer sheet, and teachers simply post the papers back to the central contest office where they are graded by machine.

One disadvantage of multiple choice contests is the ease and effectiveness of guessing. In a series of studies, Clark and Pollard measure the impact of guessing (Clark and Pollard, 2004a; 2004b). The AIC has adopted some of their resulting suggestions, including integer answer questions and the effective use of distractors.

3. Creating Algorithmic Questions

In this section we show how the AIC maintains its algorithmic focus, despite the limitations of a puzzle-based setting, no assumed knowledge and a multiple choice / integer answer format. To do this, we walk through the different types of tasks that appear in the AIC and show how they stimulate thinking about algorithms in different ways. Section 3.1 begins with a discussion of the four major classes of multiple choice questions, and Section 3.2 introduces the AIC's distinctive three-stage integer answer tasks.

3.1. *Multiple Choice Tasks*

The multiple choice tasks in the AIC can be classified into four broad categories:

- *Algorithmic tasks*, which encourage students to develop an informal algorithm to solve a given puzzle;

- *Logic tasks*, which use non-algorithmic puzzles to encourage rigorous reasoning and case analysis;
- *Tracing tasks*, which are simple tasks that ask students to follow a well-defined set of instructions;
- *Analysis tasks*, where students probe the strengths and weaknesses of a given algorithm or problem.

This classification is of course rough – tasks may fit into more than one of these categories, and sometimes a task does not fit into any (such as the occasional pattern matching task).

Algorithmic Tasks

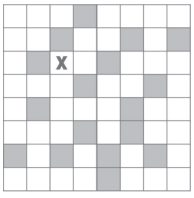
Algorithmic multiple choice tasks have the same aim as traditional informatics olympiad tasks: to encourage competitors to devise an efficient and correct algorithm to solve some problem.

In the multiple choice setting however, this aim is far less explicit. Tasks cannot ask for an algorithm directly – inexperienced students might not even know what an algorithm is. Instead, algorithmic tasks pose a puzzle where, if students are to solve it quickly and correctly, they must devise and follow some repeated systematic procedure – that is, an *algorithm*.

This is illustrated in the task *Dungeon* (Fig. 1), taken from the first ever AIC. Although this problem can be solved in time through guesswork or trial and error, it is faster and more reliable to work systematically outwards from the token, identifying all the squares that are one move away, then two moves away, and so on. In other words, a student who

Dungeon *(AIC 2005, Intermediate)*

A token (marked 'X' in the diagram) is in a maze. You may move the token around according to the following rule: in each move the token may travel any distance either horizontally or vertically, but it cannot pass over or stop on a shaded square.



For example, from its starting position the token could travel either one square right, one square down, two squares down or three squares down in a single move. To reach any other square would require more than one move.

What is the minimum number of moves that you need to ensure that the token can reach any white square from its starting position?

(A) 8 (B) 9 (C) 10 (D) 11 (E) 12

Fig. 1. The algorithmic task "Dungeon".

has never learned about algorithms or programming may find themselves conducting an informal breadth-first search.

It is important to choose the size of the problem carefully. In a multiple choice setting, the only way a student can communicate their algorithm is to run it by hand, and then select the final output from amongst the five choices offered in the question. This means that the problem must be small enough for the student to manually step through their algorithm in a short period of time. On the other hand, if the problem is too small then an algorithm becomes unnecessary, and students can simply solve the puzzle through exhaustive (or educated) trial and error.

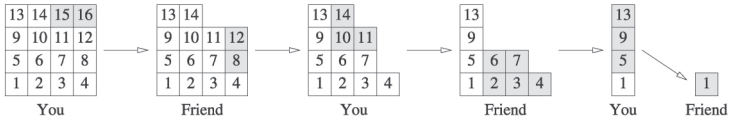
Another example of an algorithmic task is *Chocolate* (Fig. 2). Again students can approach this problem through trial and error, and eventually they will succeed. However, a fast and accurate solution is to identify all "board shapes" from which you can "win" in one move, then all board shapes from which your opponent must lose in two moves, and so on. With only 19 possible board shapes, it is certainly feasible to iterate this by hand.

Like an informatics olympiad problem, algorithmic multiple choice tasks reward both correctness and efficiency. Correctness is rewarded directly by giving points for the cor-

Chocolate *(AIC 2007, Intermediate & Senior)*

You and a friend are eating a block of chocolate by taking alternate bites – you take the first bite, your friend takes the second bite, you take the third bite, and so on.

Not all bites are the same size. To take a bite, you must choose a square and eat every square above it and/or to the right (including the square you chose). For example, consider the block illustrated below.



Here you take the first bite by choosing square 15 and eating everything above and to the right of it, which is just squares 15 and 16. Your friend then chooses square 8, thereby eating squares 8 and 12. You choose square 10, your friend chooses square 2, you choose square 5 and your friend finishes the block by choosing square 1.

As it happens, square 1 is a new experimental broccoli flavour that neither of you wants to eat. Suppose the chocolate has been reduced to the 3×3 block below, and it is your turn to take a bite. Only one of the following squares will allow you to force your friend to eventually take square 1 – which should you choose?

9	10	11
5	6	7
1	2	3

(A) Square 5 (B) Square 6 (C) Square 9

(D) Square 10 (E) Square 11

Fig. 2. The algorithmic task "Chocolate".

rect answer. Efficiency is rewarded indirectly by leaving the student more time to finish the rest of the exam.

Logic Tasks

Logic tasks do not require an algorithm per se; instead they are typically solved through rigorous reasoning and case analysis. These skills, however, are essential for algorithm design, and so the AIC explicitly asks questions of this type.

An example is the task *Palindromes* (Fig. 3), which is an exercise in rigorous case analysis. It is easy to show that the task is impossible using only one increment; with a little more work one can argue that it is impossible using two increments, and similarly for three. The task is finished by finding an explicit construction that solves it in four.

Another logic task is *Cities* (Fig. 4). Here the focus is on creative logic rather than case analysis. A promising chain of reasoning might be to observe that the closest pair of cities must be adjacent; from here we can work outwards to more distant cities until the locations of all the cities are identified.

Tracing Tasks

Tracing problems are simple tasks that ask students to step through a given procedure or follow a given set of well-specified instructions (which are presented in plain English, not code or pseudocode). This tests students' ability to understand and follow a given algorithm, which is an important precursor to developing their own algorithms.

Tracing tasks are typically the easiest tasks on an AIC paper. An example is *Leet Speek* (Fig. 5), which was the first question in the easiest AIC division in 2006.

Analysis Tasks

The final class of multiple choice tasks is analysis tasks, which ask students to study the behaviour of a given algorithm and/or its underlying problem. Such a task might ask for an approximate number of steps or a worst case scenario, thereby encouraging thinking about computational complexity and pathological cases. Other analysis tasks might ask

Palindromes *(AIC 2007, Intermediate & Senior)*

Given a sequence of digits, we can change it according to the following rules:

(i) If three consecutive digits are palindromic (i.e., the first is the same as the third), then all three digits can be removed. For instance, the sequence $16\underline{323}5$ can be changed to 165.

(ii) Any digit except for 9 may be increased by one. For instance, $1\underline{6}6725$ could be changed to 176725 (thus allowing the 767 in the middle to be removed).

What is the least number of times that rule (ii) must be used in order to remove all the digits from the sequence 294563011 ?

(A) 1 (B) 2 (C) 3 (D) 4 (E) 5

Fig. 3. The logic task "Palindromes".

Cities *(AIC 2007, Intermediate)*

The land of Pitopia is centred upon a large circular lake. Around this lake is a circular highway, with five cities placed along the highway. The distances between the cities are as follows:

Distance	City P	City Q	City R	City S	City T
City P		6 km	2 km	3 km	4 km
City Q	6 km		4 km	3 km	2 km
City R	2 km	4 km		5 km	6 km
City S	3 km	3 km	5 km		1 km
City T	4 km	2 km	6 km	1 km	

Note that there are always two different ways of travelling from one city to another (corresponding to the two different directions around the lake); the table above lists the shorter distance in each case.

You are travelling along the highway in a constant direction around the lake. In which order might you travel past the five cities?

(A) P, Q, S, T, R (B) P, R, S, T, Q (C) P, R, Q, T, S

(D) P, S, Q, T, R (E) P, T, S, Q, R

Fig. 4. The logic task "Cities".

Leet Speek *(AIC 2006, Junior)*

To translate an English phrase to leet speek, the following rules are used:

1. Replace any occurrences of `two`, `tu`, `too` or `to` with the digit `2`;
2. Replace any occurrences of `four`, `fore` or `for` with the digit `4`;
3. Replace any occurrences of `eight`, `eat` or `ate` with the digit `8`;
4. Remove any remaining vowels (letters `a`, `e`, `i`, `o` or `u`).

Note that rules 1–3 cannot cross word boundaries. For instance, you can turn `foretold` into `42ld`, but you cannot turn `sleigh time` into `sl8ime`.

Consider the sentence "`They ate two great fortune cookies for tea.`" When converted to leet speek, how many letters and digits does the final version contain?

(A) 16 (B) 17 (C) 18 (D) 20 (E) 24

Fig. 5. The tracing task "Leet Speek".

students to find the error in a sequence of instructions (highlighting "debugging" skills) or to complete an exhaustive set of input scenarios (highlighting "testing" skills).

The purpose of analysis tasks is to encourage skills that complement algorithm design – it is important for programmers to be able to analyse the correctness, efficiency and robustness of their own code.

An example of an analysis task is *Pizza Delivery* (Fig. 6). This is a simpler task – instead of an algorithm the user is presented with a straightforward sequence of steps, and the student must "debug" the sequence to locate the single error.

Pizza Delivery *(AIC 2008, Junior, Intermediate & Senior)*

A pizza delivery boy must deliver pizzas to 11 houses in a lane, with one pizza for each house. All houses are on the same side of the lane. His instructions read:

"Go to Mel's house, then go F3, B1, F6, B3, F1, B5, F1, F5, F1, B3",

where F3 means "go forward 3 houses", B1 means "go back 1 house", and so on.

Unfortunately, one of the instructions has been written down incorrectly. Where is the mistake?

(A) the 1st or 2nd instruction
(B) the 3rd or 4th instruction
(C) the 5th or 6th instruction
(D) the 7th or 8th instruction
(E) the 9th or 10th instruction

Fig. 6. The analysis task "Pizza Delivery".

The Bulgarian contest Chernorizets Hrabar (Tabov *et al.*, 2003) contains interesting analysis tasks of a different type. Given a low-level algorithm (expressed in code, pseudocode or as a flowchart), students are required to understand the algorithm and find out what high-level task it performs (typically some kind of numerical task, such as exponentiation, summation or root finding).

3.2. *Three-Stage Tasks*

Consider again the algorithmic tasks described in the previous section – tasks that encourage students to devise an algorithm that is both efficient and correct. Multiple choice algorithmic tasks suffer from two key difficulties:

- It is difficult to *encourage* students to find an algorithm. Students may simply attack a task with logic and/or educated guesswork and not realise that a more systematic procedure is necessary. Moreover, because tasks must be small enough to solve with pen and paper, logic and guesswork will often satisfy students that they have found the correct answer (regardless of whether this is true).
- It is difficult to *evaluate* whether students have found an algorithm. Because tasks are small, the right logic with some inspired guessing may well lead to the correct answer anyway, particularly when only five multiple choice options are available.

These difficulties stem from the facts that (i) multiple choice tasks are small, and (ii) they only contain a single "test case". To work around these difficulties, the AIC includes a series of *three-stage tasks*.

A three-stage task is a group of three related questions, each asking students to solve the same problem but with data sets of increasing size. Students will first be given the story and overall task description, followed by the three data sets that form the three questions. The answer for each data set is a single integer in the range 0–999. This is illustrated in the task *Spiders* (Fig. 7).

Spiders *(AIC 2006, Junior & Intermediate)*

As everybody knows, girl spiders are difficult to distinguish from boy spiders. You believe you can do this by using their size. For each spider colony you study, you declare that "all spiders larger than x millimetres are girls, and all spiders smaller than x millimetres are boys." Of course you will be wrong some of the time; your task is to choose a value for x so that you make as few errors as possible.

Each of the following scenarios describes a colony of spiders, giving the sizes of each boy and girl spider in millimetres. For each scenario, you must choose a value of x that gives you as few errors as possible (that is, the number of girl spiders of size $< x$ plus the number of boy spiders of size $> x$ is as small as possible). This value of x will be your answer to the question.

The size of every spider is even; each of your answers must be an *odd number*. There will never be more than one best possible answer.

1.

Sizes of boy spiders	12, 12, 14, 18, 22, 24
Sizes of girl spiders	16, 20, 26, 28, 28, 30, 30, 30, 32

2.

Sizes of boy spiders	14, 14, 14, 18, 18, 22, 26, 26
Sizes of girl spiders	16, 20, 20, 24, 24, 24, 28, 28, 28, 28

3.

Sizes of boy spiders	16, 18, 20, 22, 26, 28, 32, 34, 40, 42
Sizes of girl spiders	24, 26, 30, 36, 38, 42, 46, 48, 48, 50

Fig. 7. The three-stage task "Spiders".

Three-stage tasks are explicitly designed to entice students into formulating algorithms. The first data set is typically small, and can be solved using ad-hoc techniques. By the second data set the student should have a feel for the problem, and hopefully will be developing systematic techniques for manipulating the data. The third data set is larger again, and by this stage students should be able to apply their systematic techniques quickly and efficiently. The integer answers (0–999) serve to limit the value of guesswork, particularly for the larger data sets.

In summary, three-stage tasks aim to address the earlier difficulties as follows:

- By repeatedly asking students to solve similar tasks, they will be *encouraged* to develop algorithms;
- By successfully solving the larger data sets (which are less prone to ad-hoc techniques and guesswork), students can demonstrate that their algorithms work, allowing a clearer *evaluation* of algorithmic thinking.

The AIC includes three distinct three-stage tasks on every paper (questions 7–9, 10–12, and 13–15). Three-stage tasks need not be purely algorithmic; see for instance *Lost* (Fig. 8), a task that includes aspects of algorithms, analysis and debugging.

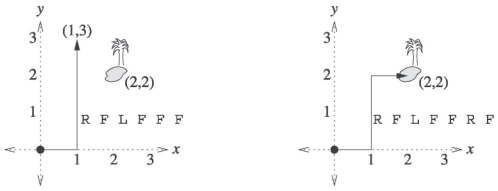
Lost *(AIC 2007, Intermediate & Senior)*

You are wandering through the desert with a map, which shows the desert as an (x, y) coordinate plane. You begin your journey at $(0, 0)$ facing north. In your hands are directions to an oasis, written as a sequence of letters. The possible letters are:

- `F`, indicating that you should walk forwards one kilometre in the direction you are currently facing;
- `L`, indicating that you should turn 90° to the left;
- `R`, indicating that you should turn 90° to the right.

Alas, the directions contain a critical mistake – one of the right hand turns has been deleted. Fortunately your map shows the coordinates of the oasis, and so you hope to use this information to work out where the missing right hand turn should be.

For example, suppose the directions are `R F L F F F` and the oasis is at $(2, 2)$. The first diagram illustrates this path, which ends at the incorrect location $(1, 3)$.



With some thought it can be seen that the directions should be `R F L F F R F`. That is, the missing right hand turn takes place just before the final walk forwards, as shown in the second diagram above.

Each scenario below lists a series of directions, followed by the location of the oasis. For each scenario, how many letters appear before the missing `R` must be inserted?

1. `RFFLFLFFFRFF` $\longrightarrow (3, 3)$

2. `RFFLFRFFLFFLFLFRFFR`
`FFLFLFRFRFFRFLFFLF` $\longrightarrow (5, 5)$

3. `RFFFLFFRFFFRFRFFRFFFF`
`FFLFFLFFFLFLFFFFFLFF` $\longrightarrow (8, 8)$

Fig. 8. The three-stage task “Lost”.

4. Conclusion

In this paper we describe the Australian Informatics Competition, a short pen-and-paper contest that aims to make problems about algorithms accessible to a much broader range of students than traditional programming contests. The contest is designed to be manageable for teachers and schools even when students sit the contest in large numbers (such as entire classes). The tasks are puzzle-based with a strong focus on algorithmic thinking, and the unique three-stage tasks are designed to both encourage and reward the development of algorithms in ways that typical multiple choice tasks cannot.

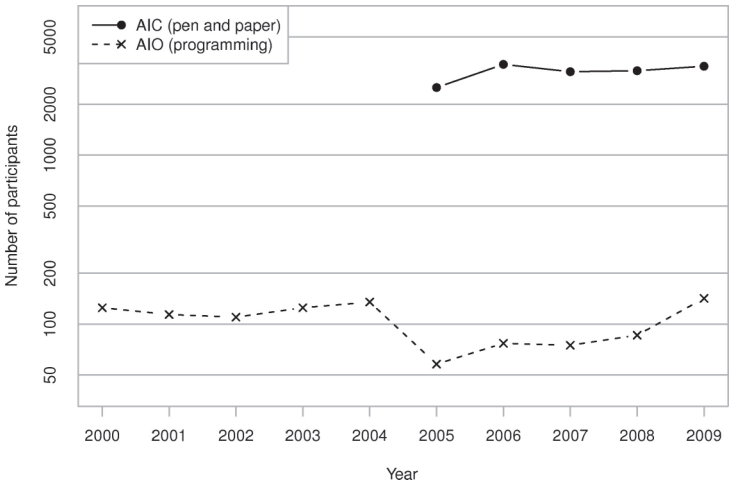


Fig. 9. Participation in entry-level informatics contests over the past decade.

The AIC has been extremely well received in Australia since its introduction in 2005. Fig. 9 plots the number of entrants over the last decade for the AIC in comparison to the Australian Informatics Olympiad (AIO), which is the entry-level programming contest for the national olympiad programme. The first ever AIC attracted 2511 students, compared with a maximum of 142 students for the programming contest over its entire 12-year history. Current AIC entries number well over 3000, and the contest now attracts international entrants from New Zealand and Singapore.

It is interesting to note the sharp dip in entrants for the programming contest in the year when the AIC was introduced (58 students compared with 135 the year before). That year also marked a significant drop in the number of zero scores, suggesting perhaps that inexperienced students were moving from the programming contest to the more accessible written contest (understandable, given the respective aims of each contest). Happily, numbers for the programming contest have since grown again to beyond their pre-2005 levels.

The AIC now enjoys an important place within the Australian informatics olympiad programme. The reader is referred to (Burton, 2008) for a wider view of the national programme as a whole.

As noted in the introduction, other communities have responded to similar challenges in different ways. The Beaver (Bebras) competition retains a dependence on computers, which the AIC explicitly aims to avoid; however, this allows Beaver to offer innovative problems such as interactive tasks that are impossible in a pen-and-paper setting. See (Dagienė and Futschek, 2008) for details, as well as a list of criteria that can help contest designers distinguish good tasks from bad.

Finally, readers are encouraged to try their hand at the sample tasks in this paper – solutions can be found in the table below.

Dungeon:	B	*Palindromes:*	D	*Leet Speek:*	A	*Spiders:*	25, 19, 35
Chocolate:	B	*Cities:*	C	*Pizza Delivery:*	D	*Lost:*	7, 20, 21

Acknowledgements

The author is grateful to the AIC problems committee for their hard work and enthusiasm over the last five years, and in particular to David Clark who chairs the committee and without whom the AIC might never have seen the light of day. The author is supported by the Australian Research Council's Discovery Projects funding scheme (project DP1094516).

References

Anido, R.d.O., Menderico, R.M. (2007). Brazilian Olympiad in Informatics. *Olympiads in Informatics*, **1**, 5–14.

Boersen, R., Phillipps, M. (2006). Programming contests: Two innovative models from New Zealand. Available from `http://www.bwinf.de/competition-workshop/papers.html`.

Burton, B.A. (2008). Informatics olympiads: Challenges in programming and algorithm design. In G. Dobbie and B. Mans (Eds.), *Thirty-First Australasian Computer Science Conference* (*ACSC 2008*), Vol. 74 of *CRPIT*. ACS, Wollongong, NSW, Australia, 9–13.

Choijoovanchig, L., Uyanga, S., Dashnyam, M. (2007). The Informatics Olympiad in Mongolia. *Olympiads in Informatics*, **1**, 31–36.

Clark, D. (2006). The 2005 Australian Informatics Competition. *The Australian Mathematics Teacher*, **62**(1), 30–35.

Clark, D., Pollard, G. (2004a). A measure of the effectiveness of multiple choice tests in the presence of guessing: Part 1, crisp knowledge. *Mathematics Competitions*, **17**(1), 17–33.

Clark, D., Pollard, G. (2004b). A measure of the effectiveness of multiple choice tests in the presence of guessing: Part 2, partial knowledge. *Mathematics Competitions*, **17**(1), 34–47.

Dagienė, V. (2006). Information technology contests – Introduction to computer science in an attractive way. *Informatics in Education*, **5**(1), 37–46.

Dagienė, V., Futschek, G. (2008). Bebras international contest on informatics and computer literacy: Criteria for good tasks. In R.T. Mittermeir and M.M. Sysło (Eds.), *Informatics Education – Supporting Computational Thinking*, Vol. 5090 of *Lecture Notes in Comput. Sci.*. Springer, Berlin, 19–30.

Merry, B., Gallotta, M., Hultquist, C. (2008). Challenges in running a computer olympiad in South Africa. *Olympiads in Informatics*, **2**, 105–114.

Pohl, W. (2007). Computer science contests in Germany. *Olympiads in Informatics*, **1**, 141–148.

Tabov, J., Kelevedzhiev, E., Lazarov, B. (2003). Multiple choice style informatics. *Mathematics Competitions*, **16**(2), 60–69.

Verhoeff, T. (2009). 20 years of IOI competition tasks. *Olympiads in Informatics*, **3**, 149–166.

B.A. Burton was director of training for the Australian informatics olympiad programme from 1999–2008, and now sits on the IOI Scientific Committee. He has been involved in the Australian Informatics Competition since its inception in 2005, and edited the AIC papers for the first three years. He is presently a QEII Research Fellow at the University of Queensland, where he works on computational geometry and topology in three and four dimensions.

Olympiads in Informatics, 2010, Vol. 4, 15–29

© 2010 *Institute of Mathematics and Informatics, Vilnius*

Mutual Influence of the National Educational Standard and Olympiad in Informatics Contents

Vladimir M. KIRYUKHIN
Department of Informatics and Control Systems, National Research Nuclear University "MEPhI"
Kashirskoe Shosse 31, 115409 Moscow, Russian Federation
e-mail: vkiryukhin@nmg.ru, msvm@lianet.ru

Abstract. The success of a contestant in the IOI certainly depends too much on the talent of the pupil, the qualification of her/his teachers and additional work with an individual coach. The success of a national team in the IOI however also substantially depends on the level of teaching of informatics at schools in the corresponding country. Strong basic education at regular schools could lead to programming competitions with more participants and allow the selecting of those pupils, who are able to storm the top ranked IOI positions. The goals and the content of the teaching in the Russian schools are defined by the State School Educational Standard (SSES). Taking it into account the teachers and the individual coaches should organize the preparation of students for both the National Olympiad in Informatics and the IOI. In the given article the influence of SSES on the Russian Olympiad in Informatics content is considered.

Key words: olympiads in informatics, IOI, secondary school education, state school educational standard, olympiads in informatics content, competition tasks content.

1. Introduction

Discussing the content of the Russian Olympiad in Informatics (RusOI) for secondary school students it is necessary to stress that it is an official annual event of the Ministry of Education and Science of Russian Federation. RusOI is included in the system of Unified State Examination (USE) of secondary school students. Winners from the final stage of RusOI are accepted in educational programs of universities in the domain of informatics (computer science) without entrance examinations. It means that all secondary school students of the country should have equal opportunity to participate in the olympiad and this opportunity must be guaranteed at the state level.

The goals and the content of the teaching in Russian state schools are defined by the State School Educational Standard (SSES). Organization of the USE also obeys this standard. This fact influences the content of the RusOI in two different ways. On one hand, the olympiad content has to meet all the requirements of the standard concerning the USE of secondary school students in informatics. On the other hand, the olympiad content should be accessible to all students. Each student has to have an opportunity to participate in the olympiad.

In the given article the mutual influence of SSES and RusOI content is considered in depth. In Section 2 the Educational Standard is presented. In Section 3 a comparison

is made between the content of education in informatics and information technologies at schools and the content of the Olympiad in Informatics. In Section 4 some conclusions are formulated. One appendix is included in the paper – a translation of the Russian SSES in Informatics and Information technologies for the high level education in Russian schools.

2. Characteristics of the Russian SSES in Informatics and IT

At the present time, Russian SSES is the basic document which defines the content of school education in informatics and information technologies (the part of the standard concerning the high level of secondary school is given in Appendix). It postulates the main aspects of the modernization of Russian schools, namely:

- introduction of specialized courses in informatics and IT for in-depth training of secondary school students;
- reflection of personal and professional interests of students in the content of secondary education;
- inclusion of creative activities in the learning process;
- formation of key competencies to ensure readiness of students for using the acquired knowledge and skills in real life.

The definition of such goals and tasks had a positive impact on the development of the RusOI. This is because Russian SSES provides, on the first place, the possibility for studying all important subjects during the basic school level (5–9th grades). Until 1998 informatics in Russia was taught only in the high school level (10–11th grades). Students could prepare for participation in informatics olympiads only by means of additional training at specialized educational centers or with individual coaches. Now, lessons in computer science/informatics and information technology are included in the basic level of the school (8–9th grades), and additional lessons in information technologies are also included in the subject "Technology", which starts in 5th grade. The number of classes dedicated for these subjects in each grade is shown in Table 1. So, each student has enough classes in informatics and information technologies in order to meet the requirements of the RusOI and to enter its preparation program.

Thanks to the possibility of all students studying informatics courses in basic school, now the olympiad community looks younger. Children start to express a big interest to information technologies even in the primary school (1–4th grades) where they have a

Table 1

Classes for studying Informatics and Information Technologies in the basic school (5–9th grades)

Subjects	Number of classes (per year)					Total
	5th grade	6th grade	7th grade	8th grade	9th grade	
Informatics and IT	0	0	0	35	70	105
Technology and IT	70	70	70	35	0	245

possibility of receiving initial algorithmic knowledge and get to work with computer programs. This is very important for the children because it eliminates the psychological barriers when they are sitting in front of computer, helps them to understand the actual capabilities of the computers, to develop algorithmic thinking, to initiate informational and instructional activity in educational process, and as a final result to generate an interest in serious informatics. In order to help the teaching of informatics in primary school many free resources were published on the Internet at the site "State Internet-collection of digital educational resources" (SIC-DER, 2007), part "Informatics and ICT", 2–6th grades, named *Set of Virtual Labs on Informatics "Problem Book 2–6"* (in Russian, *Система виртуальных лабораторий по информатике "Задачник 2–6"*).

The second important impact of Russian SSES is that its part (see the Appendix) concerning the high level of the schools (10–11th grades) includes aspects of professional-oriented training of the students. Entering high school each student has to select some professional-oriented courses that will form her/his individual path of development. Thanks to this, each student has the opportunity to implement her/his creative potential in informatics at school, not just at the special centers for training of talented pupils. Obviously, the quantity of such special centers is not enough, and they are only at universities. It is important that the professional-oriented training at schools, including country schools, considerably expands the scope of the students involved in the olympiad movement.

The quantity of classes devoted to professional-oriented courses in informatics is presented in Table 2. It is possible to assert, with a significant confidence, that this quantity of classes satisfies the requirements of talented pupils for deep studying of informatics in many schools in Russia.

The SSES emphasize that the professional-oriented courses in informatics and information technologies for high level of the school aim to achieve the following goals:

- developing and structuring the student's knowledge of mathematical objects in informatics; helping student to understand the means of modeling; teaching them to construct descriptions of the objects and the processes, allowing their computer modeling; helping them to understand information processes in biological, technological and social systems;
- mastering student's skills: to build mathematical objects in informatics, including logical expressions and programs written in formal language, satisfying some description; to create programs in programming language following some specification; to use standard tools and to adjust them for needs of the user;

Table 2

Number of classes for professional-oriented courses in Informatics in the high level (10–11th grade)

Professional-oriented course	Number of classes in Informatics for professional-oriented courses	
	Federal component	Additional component for special courses
Physic/mathematics	4 classes per week	From 1 to 5 classes per week
Information technologies	4 classes per week	From 1 to 7 classes per week

- developing student's algorithmic thinking, including elements of systematic thinking and abilities for formalization;
- forming student's sense of responsibility for the results of their work, positive social activity inside the information society and inadmissibility of actions that break legal and ethical norms when working with information;
- mastering student's skills: to work on projects; to create, edit, format, save, and transfer information objects of various type with the help of modern software; to construct computer models; to implement information projects in a team; to do information work in various spheres claimed from the labor market.

It is important to stress that the above mentioned purposes of the professional-oriented education in informatics are also typical for the RusOI (Kiryukhin, 2008). The experience gained by the olympiad community through the years, substantially promotes creation of new educational technologies for professional-oriented education in Informatics (Kiryukhin, 2009). In particular, the olympiad experience promoted creation of new special elective courses for students, which also include olympiad content.

For these purposes many books (Kiryukhin, 2007; 2008; 2009; Kiryukhin and Okulov, 2007), now very popular in Russia, were published, which help teachers to organize training of talented students and to prepare students for participation in the informatics olympiads. An electronic collection of competition tasks was published on the Internet portal of the RusOI (RusOI, 2009). A system of training contests was created and an Internet-driven contest is organized regularly. Remote on-line practical courses in informatics and information technologies are offered to students of Russian schools (OWPI, 2008). Thus, the professional-oriented component of the preparation of students, including methodical materials and technology of programming resources, are regularly developed and updated.

Anyway, many teachers are still inclined to believe that the content of the Olympiad in Informatics is not linked to the real school curricula. We will provide more detailed analysis in the next section of occurrence of content from the RusOI in the curriculum in informatics for the high level of Russian schools (10–11th grades). It is important to demonstrate that there is a real possibility for students to receive olympiad preparation directly at schools.

3. Relation between SSES and Olympiad in Informatics Contents in Russia

If we consider the SSES in informatics and information technologies for professional-oriented level of the Russian school we will see that the general skills in informatics, necessary for talented students to implement their ideas when solving problems with a computer, are completely covered by the professional-oriented courses. It is especially necessary to stress the wide representation in professional-oriented courses of material necessary for the creation of scientific and algorithmic thinking in students (see Table 3 and the Appendix).

Comparing the content of the RusOI with the topics from the Russian SSES in informatics for professional-oriented level, presented in Table 3, it is possible to conclude

Table 3

Topics of professional-oriented courses in Informatics and Olympiad in Informatics content

Topics from Olympiad in Informatics content	Topics from professional-oriented course in Informatics
Programming language (6 classes)	Rules for construction and estimation of the performance of algorithms. Splitting a task into subtasks. Examples of graphical and numerical algorithms written in programming language.
Computable functions (2 classes)	Functions computable by algorithms. Completeness of the formalization of the computability concept. Universal computable function. Diagonal proofs of non-existence. Inductive definitions of objects. Computable functions defined by systems of functional equations.
Deterministic games with perfect information (4 classes)	Trees. Winning strategy in a game. Game interpretation of logical formulas.
Correctness proofs (4 classes)	Correspondence between an algorithm and the task specification, invariants, inductive proofs.
Practice in construction of algorithms (4 classes)	Numeral system, arithmetic and logical operations; generation of pseudo-random sequences. Algorithms for solving Calculus problems (computation of approximate surface and the value of a function, represented by series, simulation of processes described by differential equations). Brute-force algorithms. Breadth-first and depth-first search.
Data types (4 classes)	Basic data types. Matrices (arrays). Operations with numbers, matrices, strings, lists, usage of pseudo-random numbers. User-defined (abstract) data types.
Complexity of the description of an object (2 classes)	Optimality of the formalization. Algorithmic definition of randomness.
Complexity of calculation (5 classes)	Examples of effective algorithms. Exhaustive search problem.
Events. Parallel processes (3 classes)	Interaction of parallel processes, interaction with the user.

that the curriculum for professional training in informatics in the high level of the Russian school (10–11th grades) to a large degree covers the content of the RusOI. But, as experience shows, for successful performance at the RusOI, especially at the final stages, mastering of material included in the curriculum is not enough. In addition some forms of individual work with talented students is also necessary, as well as actively to develop and introduce various professional-oriented special courses at school.

Topics of professional-oriented courses in Informatics and Olympiad in Informatics content

The Russian SSES for professional-oriented level admits studying of additional special (elective) courses. It is possible to say that the philosophy of professional-training allows each student to receive an education of high quality, following an individual path of development and a profession choice.

It happens that the creation by members of the olympiad community of some professional-oriented special courses, which are accessible to all schools, is possible. In such a way the skills of the teachers preparing contestants, the set of methodical materials

which they use in work with the pupils attracted to informatics, the possibility of cooperation with other students and coaches – direct or through the global network, internet resources, training competitions, and the regular school stages of the RusOI are available for all students. And these are the important necessary conditions, which will allow each school to realize the right of each secondary school student to take part in the olympiad movement in informatics in Russia.

In creating the RusOI curriculum it is necessary to consider also the requirements of the USE defined in Russian SSES. Requirements to participants in the RusOI should correspond to the requirements of the USE. As at the olympiad the same tasks are offered to all participants, irrespective of their grade, we will consider this correspondence both for the basic and the high level of the school.

Preparation in informatics at *the basic level of school* aims to give to students general computing competences (freely to work with the computer and its software) and the ability to put this competence into practice for solving various information tasks (see Table 4).

The analysis of Table 4 shows that the requirements of Russian SSES in informatics for the basic level of schools allows even the students from 8–9th grades to partic-

Table 4
Requirements to preparation of students in the basic level
(Topics from the Standard in Informatics for 8–9th grades)

Topic from the Standard in Informatics for the basic level (8–9th grades)	Requirements to preparation of students in the basic level
Presentation of the information. Information, information objects of various types. Language as a way of presentation of the information: natural and formal languages. Formalization of the description of real objects and processes, examples of modeling of objects and processes, particularly with computer. Information processes: storage, transmission and processing of information. Presentation of the information in a digital form. Units for measuring of information. Control and feedback. The main stages of development of resources of information technologies.	**To use the acquired knowledge and abilities in practical activities and a daily life for:** creation of elementary models of objects and processes in the form of images and drawings, dynamic (electronic) tables, programs (particularly in the form of flowcharts).
Information processing. Algorithm, properties of an algorithm. Ways for description of algorithms; flowcharts. Algorithmic constructions. Logical values, operations, expressions. Partitioning of the task into subtasks, auxiliary algorithm. Processed objects: strings of characters, numbers, lists, trees, graphs.	**To understand, to know**: the main properties of algorithms and kids of algorithmic constructions: following, branching, cycling; concept of auxiliary algorithm. **To know how**: to execute the basic operations with objects – strings of characters, numbers, lists, trees; to check properties of these objects; to build simple algorithms. **To use the acquired knowledge and abilities in practical activities for:** carrying out of computer experiments with usage of ready models of objects and processes.

ipate successfully in the RusOI. As a result, the number of pupils of the 8–9th grades, as a proportion of the participants of the RusOI, permanently increases. Besides a deep knowledge of the mathematical foundations of informatics the technological aspects of education of students at the basic level of school are important, namely, the use of operating systems, network tools for information processing, debugging tools in a programming environment, high-speed typing on the keyboard, etc. The technological skills of the students influence the speed of implementing solutions of competitive tasks and have an effect on the results of participants from the 8–9th grades.

Preparation for the RusOI continues with the professional training at high school. Comparing the content of last the ten years competitive tasks and the curriculum of professional-oriented course in informatics allows the emphasis of those topics of professional-oriented course and the competences of the students, which are valuable for the olympiad. It is necessary to remark, that all topics of the professional-oriented course in informatics are included in competitive tasks. In Tables 5–8 topics from the professional-oriented course in informatics are presented in parallel with the corresponding requirements toward the level of preparation of pupils. The comparative analysis of topics and requirements shows that achievement of each requirement could be evaluated with any competitive tasks. Moreover, each competitive task reveals the competence of the student not fragmentarily but within a set.

Table 5
Correspondence between topics of the standard and requirements at the level of preparation of pupils for the area "Information and informational processes" of the professional-oriented level of the school

Topics of the Standard from the area "Information and information processes" for professional-oriented level of the school	Requirements to the level of preparation of pupils
Kinds of information processes	To recognize the information aspects in the activity of a person and informational interaction in the elementary social, biological and technical systems.
Models of activity of the person	To recognize the information aspects in the activity of a person and informational interaction in the elementary social, biological and technical systems.
Logic and algorithms mathematical models	To know kinds and properties of information models of real objects and processes, as well as the methods and the resources for computer implementation of information models.
	To interpret the results received during simulation of real processes.
Logic and algorithms	To know the general structure of the activities for creation of computer models. To execute virtual experiments and independently to create the elementary models with the use of educational virtual laboratories and the modeling environments.
Basics of the algorithms theory	Properties of algorithms and the main algorithmic constructions; the thesis about completeness of the formalization of notion algorithm.
Programming language	The main constructions of the programming language.

Table 6
Correspondence between topics of the standard and requirements at the level of preparation of pupils for the area "Informational work of the person" of the professional-oriented level of the school

Topics of the Standard from the area "Informational work of the person" for professional-oriented level of the school	Requirements to the level of preparation of pupils
Informational ethics and rights, information security	To know norms of informational ethics and rights, information security, principles of assurance of information security.
Forms of a professional information work of the person	To know assignments and areas of usage of the information and communication technologies and the information resources. To estimate the numerical parameters of information objects and processes: required memory capacity for information storage; a transfer rate and information processing.

Table 7
Correspondence between topics of the standard and requirements at the level of preparation of pupils for the area "ICT Resources" of the professional-oriented level of the school

Topics of the Standard from the area "ICT Resources" for professional-oriented level of the school	Requirements to the level of preparation of pupils
Computer architecture and computer networks	Means and support of reliable functioning of ICT resources. Basic principles of the organization and functioning of computer networks. To fulfill requirements of safety techniques, hygiene, ergonomics and resource-saving by operation with information resources. Support of reliable functioning of ICT resources. To eliminate the elementary faults, to instruct users in basic principles of ICT usage.

Long-term experience in carrying out the RusOI shows that the participants are forming, in addition to all other, the following qualities:

- deep knowledge of the mathematical bases of informatics and the theory of algorithms;
- steady knowledge of information processes, information forms, ways of presentation and transfer of information;
- clear understanding of the principles of operation of computers and the role of software;
- steady practical skills of independent solving of practical tasks with computer programs;
- fluent possession of keyboard input in both Russian and English language;
- clear understanding the structure of computer, skills in operation with peripherals and various media;
- free skills in using: operating system, graphic interface, files system, systems for retrieval, archiving and converting resources, and other applications;
- steady skills in operating with shared and personal resources in a local computer network;

Table 8

Correspondence between topics of the standard and requirements at the level of preparation of pupils for the areas "Technologies of creation and processing of the text information", "Technology of creation and processing of the graphics and multimedia information", and "Processing of the numerical information" of the professional-oriented level of the school

Topics of the Standard from the areas "Technologies of creation and information processing" of the professional-oriented level of the school	Requirements to level of preparation of pupils
Usage of special software tools and digital equipment.	Presentation of information in the form of multimedia objects with a reference system (for example, for allocation in a network). Creations of own databases, digital archives, media libraries. Operating with informational objects, using available knowledge for the possibilities of information and communication technologies. Performing statistical data processing with computer.
Search technologies and information storage Telecommunication technologies Technology of control, planning and the activity organization	Operating with informational objects, using available knowledge for the possibilities of information and communication technologies including creation of data storage structures; to use help systems and other sources of supplemental information; to obey intellectual property rights on information.

- steady skills in operating with specialized software (programming environments, compilers, debuggers);
- free skills in operating with tools of a global computer network (registration, data transfer, information security);
- clear understanding of the norms for operating with information;
- developed sense of self-control and responsibility;
- skills in independent planning of goals;
- general cultural skills of the organization of a jobsite;
- general cultural skills of etiquette;
- strong will for achieving results.

From the above analysis it follows that these competences undoubtedly define a steady professional interest of a student in informatics and a desire for a progress in the professional sphere. Clearly, such a student is potentially active during teaching and will become a successful university student, irrespective of the educational institution selected by her/him. These competences will help to develop her/him and to be successful in any professional field of activity, which she/he will chose in the life.

4. Conclusion

The problems considered in this article play a very important role in the further development of the informatics olympiads in many countries, including the development of the IOI. A large difference between the requirements of the State School Education Standard in Informatics and the content of the informatics olympiads will make these olympiads

accessible only for small number of "professional in informatics" secondary school students. In such case the role of the informatics olympiads will only be to allocate medals among the participants. But this is not the goal. The informatics olympiads should be an example for further development of the State School Education Standards in Informatics and a stimulation for improvement of teachers' activity in their difficult work to find and educate the intelligent elite in each country.

The author hopes that this article will not be the last in the series of articles on the subject and that other members of the IOI community will also share their experience in solving above mentioned problems.

References

Kiryukhin, V.M. (2007). The modern contents of the Russian national olympiads in informatics. *Olympiads in Informatics*, 1, 90–104.

Kiryukhin, V. (2008). *Informatics. Russian Olympiads*. Issue 1, Prosveschenie, Moscow (in Russian, *Информатика. Всероссийские олимпиады*. Выпуск 1). http://www.prosv.ru/book.aspx?ob_no=209&d_no=13828&ltype=9577&subject=1467

Kiryukhin, V. (2009). *Informatics. Russian Olympiads*. Issue 2, Prosveschenie, Moscow (in Russian, *Информатика. Всероссийские олимпиады*. Выпуск 2). http://www.prosv.ru/book.aspx?ob_no=209&d_no=17575&ltype=9577&subject=1467

Kiryukhin, V. (2009). *Informatics. International Olympiads*. Issue 1, Prosveschenie, Moscow (in Russian, *Информатика. Международные олимпиады*. Выпуск 1). http://www.prosv.ru/book.aspx?ob_no=209&d_no=17310&ltype=9577&subject=1467

Kiryukhin, V., Okulov, S. (2007). *Methods of Problem Solving in Informatics. International Olympiads*. LBZ (BINOM. Knowledge Lab), Moscow (in Russian, *Методика решения задач по информатике. Международные олимпиады*).
http://www.lbz.ru/katalog/products/literatura-dlja-shkol/informatika/olimpijskie-vysoty/metodika-reshenija-zadach-po-informatike-699

OWPI (2008). *Open Web-Practice in Informatics* (in Russian).
http://webpractice.cm.ru

RusOI (2009). *Site of the Russian Olympiad in Informatics for Secondary School Students* (in Russian).
http://www.info.rosolymp.ru

SIC-DER (2007). *State Internet-Collection of Digital Educational Resources* (in Russian).
http://www.school-collection.edu.ru

Appendix. The Russian SSES in Informatics and ICT for High Level of the School

A1. Introduction

Studying of Informatics and Information and communication technologies at a professional-oriented level of the general secondary (full) school is directed to achievement of the following goals:

- developing and structuring the student's knowledge of mathematical objects in Informatics; helping student to understand the means of modeling; teaching them to construct descriptions of the objects and the processes, allowing their computer modeling; helping them to understand information processes in biological, technological and social systems;

- mastering student's skills: to build mathematical objects in Informatics, including logical expressions and programs written in formal language, satisfying some description; to create programs in programming language following some specification; to use standard tools and to adjust them for needs of the user;
- developing student's algorithmic thinking, including elements of systematic thinking and abilities for formalization;
- forming student's sense of responsibility for the results of their work, positive social activity inside the information society and inadmissibility of actions that break legal and ethical norms when working with information;
- mastering student's skills: to work on projects; to create, edit, format, save, and transfer information objects of various type with the help of modern software; to construct computer models; to implement information projects in a team; to do information work in various spheres claimed from the labor market.

A2. The Full Level of School Education Curriculum for Profile Course in Informatics and ICT. Basis Concepts of Informatics and Information Technologies

A2.1. Information and Information Processes

Kinds of information processes. Transfer of information. Signals, coding, decoding and distortion of information. Discrete (digital) representation of text, graphic, sound and video information. Speed of transfer of information. *Perception, storing and processing of information by human, limits of sensitivity and resolution of sense organs.* (N.B. Italics mark the material taught in class but not included in the Requirements to the level of preparation.)

Systems, components, state and interaction of components. Information interaction in a system, control and feedback.

Models in human activity. Descriptions (information models) of real objects and processes, conformity of a description to the object and the purposes. Descriptions by schemes, tables, graphs, formulas. Usage of descriptions (information models) in human interaction, practical activities, research.

Mathematical models: examples of logic and algorithmic languages, use of them for the description of objects and processes of the nature (alive and lifeless) and technology, including physical, biological, economic processes, and information processes in technical, biological and social systems. Use of simulation environments (virtual laboratories) for carrying out of computer experiments in educational activity.

Numeral systems. Logic and algorithms. Statements, logic operations, quantifiers, validity of a statement. Strings (finite sequences), trees, lists, graphs, matrixes (arrays), pseudo-random sequences. Inductive definition of objects. Computable functions, completeness of conception for computability, universal computable function, *diagonal proofs of non existence*. *Winning strategies. Complexity of computation. Exhaustive search. Expressing computable function by system of equations. Complexity of description.* Error-correcting codes. Sorting.

Elements of the algorithms theory. Formalization of the concept of algorithm, equivalence of formalizations. Computability. Construction of algorithms and practical computations.

Programming language. Data types. Basic constructions of a programming language. Programming environment. Basic stages in development of programs. Partitioning of a task into subtasks.

A2.2. Information Activity of a Human

Kinds of professional information work of a human and used tools (hardware and information resources). The professions connected to construction of mathematical and computer models, programming, support of information activities of individuals and organizations. Role of information in the modern society and its structures: economic, social, cultural, educational. Information resources and channels of the state, society, and organization; structure. Educational information resources.

Economy of information sphere. Cost of information activities.

Information ethics and legislation, information security. Laws concerning information, offences in information sphere, measures of their prevention.

A2.3. ICT Means

Architecture of computers and computer networks. Software and hardware of computers and computer systems. Kinds of software. Operational systems. Concept of system administration.

Safety, hygiene, ergonomics, technology requirements of a computer workplace. Typical malfunctions and difficulties in the use of ICT. Equipment of a computer workplace according to the purposes of its use.

Estimation of numerical parameters of information objects and processes, characteristic for the chosen sphere of activity.

Preventive maintenance of the equipment.

A2.4. Technologies of Creation and Processing of the Text Information

Concept of desktop publishing system. Creation of computer publications.

Use of ready and creation of own templates. Use of system for spelling and grammar. Thesauruses. Use of systems of bilingual translation and electronic dictionaries. Collective work with a text, including in a local computer network. Use of the digital equipment.

Use of the specialized tools for editing of mathematical texts and graphic representation of mathematical objects.

Use of systems for recognition of texts.

A2.5. Technology of Creation and Processing of Graphic and Multimedia Information

Concept of a system for computer aided design, environments for computer design and multimedia environments. Formats of graphic and sound objects. Input and processing of graphic objects. Input and processing of sound objects.

Use of the special software and digital equipment.

Creation of complex graphic objects for various domains: transformations, effects, design. Creation and transformation of sound and audio-visual objects. Creation of presentations, performance of educational creative and design works.

Skilled works in the field of cartography; use of geographic information systems in research of ecological and climatic processes, in city management and in agriculture.

A2.6. Processing of Numerical Information

Mathematical processing of statistical data, results of experiments, including the use of peripheral sensors. Use of dynamic (electronic) tables for performance of educational tasks from various domains: processing results of natural-science and mathematical experiments, economic and ecological observations, sociological investigations, individual parameters of educational activity. Examples of elementary problems of book keeping, planning and the account of resources.

Use of tools for solving statistical and settlement-graphic problems. Processing of numerical information by sample problems of accounting and planning.

A2.7. Technologies of Search and Storage of the Information

Concept of database management systems, retrieval systems in computer networks, library information systems. Computer archives of information: electronic catalogues, databases. Organization of databases. Examples of databases: in legislation, in libraries, in public health services, in taxes administration, in social activities, in human resources management. Use of DBMS for creation of school information system.

Use of tools of retrieval systems (formation of requests) for search in educational portals and electronic catalogues of libraries, museums, book publishing, mass-media within the framework of educational tasks from various subject domains. Rules for referring the used sources.

A2.8. Telecommunication Technologies

Concept of tools for telecommunication: e-mail, chat, teleconferences, forums, space bridges, an Internet-telephony. Special software tools for telecommunication technologies. Use of telecommunication in collective activity. Technologies and tools in the global and local computer networks for protection of information from destructions and non-authorized access. Rules for use of anti-virus programs and their adjustment for automatic check of the traffic.

Tools for creation of information objects in Internet. Methods and tools for creation and support of an Internet site.

A2.9. Technologies of Management, Planning and Organization of Activity

Technologies of automated management in the educational environment. Technologies of management, planning and organization of activity of a person. Creation of organizational diagrams and schedules; automation of the control of their performance.

Systems of automatic testing and control of knowledge. Use of testing systems in educational activities. Tools for creation of simple tests and accounting the results of testing.

A3. Requirements to the Level Preparations of Graduates

As a result of studying Informatics and ICT at a profile level the pupil should:

know/understand

- symbolic mathematics;
- the basic constructions of programming language;
- properties of algorithms and the basic algorithmic constructions; the thesis about completeness of formalization of concept of algorithm;
- kinds and properties of information models of real objects and processes, methods and tools of computer implementation of information models;
- the general structure of activity on creation of computer models;
- purpose and areas of use of the basic tools of information and communication technologies and information resources;
- kinds and properties of sources and receivers of the information, ways of coding and decoding, the reason of distortion of the information during the transfer; relation between bandwidth of the channel and the speed of the transfer;
- basic principles of the organization and functioning of computer networks;
- norms of information ethics and legislation, information security, principles of maintenance of information security;
- ways and tools for maintenance of reliable functioning of ICT environment.

be able

- to differentiate: information aspects in human activity; information interaction in the elementary social, biological and technical systems;
- to build information models of objects, systems and processes, using typical tools (programming languages, tables, graphs, diagrams, formulas, etc.);
- to calculate value of complex statement from values of elementary statements;
- to carry out statistical data processing with the help of a computer;
- to interpret the results received during modeling of real processes;
- to eliminate elementary malfunctions; to instruct users on principles of ICT use;
- to estimate numerical parameters of information objects and processes: a memory size necessary for storage of the information; speed of transfer and processing of the information;
- to operate with information objects, using available knowledge of opportunities of information and communication technologies including to create structures of data storage; to use help systems and other sources of help information; to obey rules for intellectual property of information;
- to carry out virtual experiments and independently to create elementary models in educational virtual laboratories and modeling environments;
- to obey requirements for safety, hygienic, ergonomic and recourse saving work with tools of information technologies; to maintain reliable functioning of ICT environment.

be able to use the knowledge and skills in practical activities and a daily life for:

- searching and selection of information, in particular, connected to personal cognitive interests, self-education and vocational counseling;
- presenting information as multimedia objects with system of references (for example, a network hypertext); creations of own databases, digital archives, media libraries;
- preparation and performing of presentations, participation in collective discussion, fixing its course and results;
- personal and collective dialogue with use of modern software and hardware for communications;
- obeying of requirements of information safety, ethics and the legislation.

V.M. Kiryukhin is professor of the Informatics and Control Systems Department at the National Research Nuclear University MEPhI. He is the chairman of the Federal Methodical Commission on Informatics which is responsible in Russia for carrying out the national olympiads in informatics. He is the author of many papers and books in Russia on development of olympiad movements in informatics and preparations for the olympiads in informatics. He is the exclusive representative who took part at all IOI from 1989 as a member of the IOI International Committee (1989–1992, 1999–2002) and the Russian team leader. He received the IOI Distinguished Service Award at IOI 2003, the IOI Distinguished Service Award at IOI 2008 as one of the founders of the IOI making his long term distinguished service to the IOI from 1989 to 2008 and the medal "20 Years since the First International Olympiad in Informatics" at the IOI 2009.

Olympiads in Informatics, 2010, Vol. 4, 30–51

© 2010 *Institute of Mathematics and Informatics, Vilnius*

Strategy for ICT Skills Teachers and Informatics Olympiad Coaches Development

Vladimir M. KIRYUKHIN
Department of Informaticss and Control Systems, National Research Nuclear University
Kashirskoe Shosse 31, Moscow 115409, Russian Federation
e-mail: vkiryukhin@nmg.ru , msvm@lianet.ru

Marina S. TSVETKOVA
Publishing House "BINOM. Knowledge Laboratory"
Proezd Aeroporta 3, Moscow 125167, Russian Federation
e-mail: tsvetkova@lbz.ru , msvm@lianet.ru

Abstract. The raising of interest in pupils to study informatics and become involved in informatics olympiads in many respects depends on the qualification of teachers and the coaches who are engaged in the search and preparation of pupils for informatics olympiads. The analysis of tendencies of development in the Russian and international informatics competition over the last 10 years, and programs of informatization of schools in Russia, has allowed three levels of informatics and informational communication technologies (ICT) to be defined for the preparation of school teachers in Russia. In the given article these levels of ICT preparation of school teachers in Russia and their link among themselves are described. Organization of the preparation of teachers, taking into account characteristic features for each level, will allow the essential quantity of participants of the informatics olympiads to increase and will raise the results of their involvement in the Russian and international informatics olympiads.

Key words: secondary school education, informatics and informational communication technologies skills of teachers, Olympiads in informatics, IOI, informatization of education system, state educational programs, ICT competence of teachers.

1. Introduction

The Russian Government has put education in the 2001 national policy priorities and has revitalized active realization of projects of Russian education informatization. The state concept of Russian education modernization (2000–2010) additionally became a basis for the development of informatics and informational communication technologies (ICT) in secondary schools and a social activity of youth peoples of Russia.

A very important result of the fulfillment of "The State concept of Russian education modernization (2000–2010)" was a statement in 2004 of the state school educational standard which is now (2009–2010) being updated. Before 2004 there was only a minimum literacy level for school subjects which was updated every 5 years. Now the state school educational standard includes complex curricula for all school educational fields

(math, natural sciences and informatics, philology and humanities sciences, technology and art), mandatory requirements for teaching pupils and conditions of teaching process (list of equipment for school), supplies and materials for school (Federal book list for school subject).

Now the state school educational standard has introduced to schools a number of new problems:

1. To develop the interest of students in new information technologies.
2. To guarantee the creation of ICT competence for students on the basis of an informatics course.
3. To reveal talented students in the area of informatics on the basis of an open school stage of the Russian Olympiad in Informatics.
4. To give the possibility for talented pupils to prepare for the final stages of the Russian Olympiad in Informatics.
5. Actively to co-operate between schools and to offer to the students the professional-oriented elective courses from coaches in different schools over the Internet.
6. Effectively to use the informational space of school.
7. Actively to use ICT competence of pupils in training various school subjects.

For solution of these problems it is necessary to generate teachers' ICT. Obviously problems 1–3 are realized by the informatics teacher at school. Their professional competences are completely defined by the state standard in the informatics course at school. A description of the curriculums of school informatics courses, at the basic and professional-oriented levels, are presented in the article (Kiryukhin, 2010). It is exactly these informatics teachers that help to reveal talented pupils in informatics by means of an open school stage of the Olympiad in Informatics in Russia. In this case the competences of comprehensive school informatics teachers are developed together with standard state update in informatics every 5 years.

For solving problems 4–5 it is especially necessary to select coaches from among informatics teachers. These are those teachers who prepare students for involvement in the final stages of the Russian Olympiad in Informatics. The solution of task 4 by schools requires the development of curricula for the special ICT preparations of informatics teachers-coachers. The curriculum of preparation of such teachers-coachers is described for the first time in this article on the basis of the analysis by the authors of the competition task content of the Russian and International Olympiads in Informatics for the last 10 years. This curriculum expands the standard topics on informatics for schools, and also includes topics on mathematics, logic, and programming. On the basis of additional preparation topics the teacher can form elective courses for students grades 9–11 as professional-oriented training at high school. Above all these topics in the curriculum automatically become topics in specialized courses for students – future participants of the Olympiad in Informatics.

Speaking about education informatization it is impossible to forget the teachers who are not teaching informatics at school. Such teachers are the majority at comprehensive school and they should be ICT competent to effectively use the informational environment of school in partnership with pupils. Such partnership with ICT competent teachers

in various school subjects will allow ICT active pupils to develop in other school subjects, with advanced ICT usage and help define the future profession. Finally, problems 6–7 require solving unification of common ICT competences for schools teachers.

From what is mentioned above it follows that the level of ICT preparation for school teachers in various subjects should be different. It causes difficulties for ICT, teaching them and the organization of pedagogical partnership of students and teachers in using ICT, especially for the students who have been carried away from informatics. The school should not only reveal the ICT talented students, but also support ICT with other school subjects and give students a chance in their future occupation to use high information technologies. The school's task is to help talented pupils be prepared for innovative activities, including those with usage high ICT.

2. Levels of ICT Preparation of School Teachers in Russia

The analysis of the state school educational standard and contents of the competition tasks of the Russian and International Olympiads in Informatics for the last 10 years has allowed the allocation of three levels of ICT preparation for school teachers in Russia:

1. The level of ICT preparation of each teacher in comprehensive school.
2. The level of ICT preparation of informatics teachers for improving their professional skills.
3. The level of preparation of informatics teachers and coaches who will prepare students for involvement in the final stages of the Russian Olympiad in Informatics.

The first level of ICT preparation concerns each comprehensive school teacher. This level is common for all teachers. This level sets common ICT competences for teachers of different school subjects. The common ICT competence of teachers shows a degree of partnership with teachers and pupils in the informational environment of a school. It is possible to name this level the informational culture of the teachers. This level of teacher preparation is very important so that pupils can apply ICT abilities to different subjects at school. The ICT abilities of pupils are formed by informatics teachers at school.

The second level of ICT preparation is connected with professional skills teachers of informatics in Russia. This level of preparation of informatics teachers is defined by the state school educational standard on informatics, which is refreshed every 5 years. The contents of the informatics course was developed for the Russian schools in 1998. After that, in 2004, the state school education standard including informatics and ICT was accepted (Kiryukhin and Tsvetkova, 2008). Currently the new standard is being prepared for acceptance in 2010 which will update the informatics and ICT program of preparation of teachers to include new achievements in informatics and information technologies.

The third level is the highest level of preparation for informatics teachers. It is the preparation among informatics teachers of those who will be training students for participation in final stages of the Russian Olympiad in Informatics or the IOI. This level of preparation is defined by the contents of modern competition tasks and international informatics olympiads. The curriculum, developed by the authors of this paper, is based on

analysis of the Russian Olympiads in informatics contents (Kiryukhin, 2007; Kiryukhin, 2008) and the contents of IOI competition tasks (Verhoeff *et al.*, 2006; Kiryukhin and Okulov, 2007) over the 10 last years and programs of informatization of schools in Russia.

It is necessary to notice, that all three levels of ICT preparation of informatics teachers in comprehensive school are connected among themselves (Tsvetkova, 2008). As a result it is reflected in pupils' successful lives in the informational world; it especially concerns talented pupil in the field of informatics and their state support in an education system. It also influences the ICT activity of pupil in study, creative development and professional orientation.

As mentioned above the second level of preparation of informatics teachers is defined by state school educational standards of informatics and ICT or by similar documents on which the basis the informatics or computer science at comprehensive schools is studied. It is abundantly clear, that in every country there are features of such preparation of informatics teachers and it is not necessary to speak in this case about any general approach. Therefore we will only consider the preparation of teachers at the first and third levels. As to the ICT preparation of informatics teachers in Russia, these questions are in detail described in paper (Kiryukhin, 2010).

3. Common ICT Competences (Skills) of Comprehensive School Teachers

Experience has shown it is possible to consider that the modern common ICT competence of comprehensive school teachers is *the closest development area to professional teachers' skills*. This closest development area demands mandatory equipment: a teacher workstation by the computer and such ICT resources (including new digital equipment and special software), as to form the tool making computer system (TCS) of professional teacher activity which is a part *of the informational space of education.* A very important role in TCS belongs to the Internet educational resources which are already broadly used by comprehensive school teachers in Russia. Some examples of such Internet educational resources are the following:

- Federal system of informational educational resources (FSIOR, 2008; WE, 2007);
- Informational system of state certification (EGE, 2009);
- Informational content training system for secondary schools "KM-School" (KMS, 2009);
- Informational content training system for primary schools (Nachalka, 2009);
- Distance training system "Teleschool" (Teleschool, 2009);
- On-line methodical support system of teachers in informatics (MSSTI, 2008);
- Open interactive educational Internet-video channel for teachers (I-NetVC, 2008);
- Social network for teachers (SNT, 2009);
- Social contents network for schools (SCNS, 2009);
- Children's social network "Bibigon" (Bibigon, 2009);
- Open school encyclopedia by Cyril & Methodius (CM, 2009);

- Open web-practice in informatics (OWPI, 2008).

Since the partnership of teachers and students taking place is the basis of ICT usage in various school subjects, it is necessary to provide a common ICT competence of teachers across school subjects. The contents of such common ICT preparations correspond to the following:

1. Presence of the general understanding of didactic abilities of ICT.
2. Presence of understanding of uniform information space of educational institution, purpose and functioning of computer, devices for input-output of the information, computer networks and the opportunities of using them in educational process.
3. Skills in the organization of personal information environment, the interface of operating system, performance of file operations, organization of the information-educational environment as file system, the basic techniques of input-output of the information including installation and removal of applications.
4. Skills in preparation of didactic materials and briefs according to a subject curriculum by means of office technologies (distributing materials, presentations, etc.):
   - 4.1. Text input with the keyboard and techniques for text formatting.
   - 4.2. Preparation of the distributing materials containing graphic elements, typical methods of work with vector graphics tools.
   - 4.3. Methods of working with tabulated data (making lists, information cards, simple calculations).
   - 4.4. Methods of creating graphics and diagrams.
   - 4.5. Technique of creation of pedagogically effective presentations (for a lesson, a statement on teachers' meeting, a report etc.).
5. Availability of base services and technologies of the Internet in a context of its use for educational activity:
   - 5.1. Methods of navigation and search of the educational information in the Internet, its reception and saving for subsequent using in pedagogical process.
   - 5.2. Methods of work with e-mail and teleconferences.
   - 5.3. Methods of work with file archives.
   - 5.4. Methods of work with ICQ (AOL, etc) and other communication technologies.
6. Skills in the technological basis of site creation for a teacher's portfolio:
   - 6.1. Availability of understanding of purpose, structure, tools of navigation and design of a site for support of educational activity;
   - 6.2. Availability of understanding of web-page structure;
   - 6.3. Skills in simple methods of site-making which allow the representation of educational information in the form of a site;
   - 6.4. Skills in methods of Intranet and Internet publications for a site for educational activity.
7. Knowledge of digital educational resources and trends in the market of electronic editions in the sector of general education, focused on subject-professional work, and the digital educational resources executed during realization of Federal education informatization programs.

8. Skills in basic usage techniques of digital educational resources in teaching and educational process.
9. Availability of understanding of technologies and resources for distance educational processes and of the possibilities of using them in pedagogical activity.

4. The Common Curriculum of Improvement Comprehensive School Teachers

The common curriculum for the improvement of comprehensive school teachers has been developed on the basis of the analysis of various variants of similar programs in the following projects: "Teacher to the future" (Intel, 2003), "Academy of teachers" of the international initiative Microsoft "Partners in learning" (SNT, 2009), "Informatization of education system" (Tsvetkova *et al.*, 2005) and also the programs of many ICT training centers in regions of Russia (MSSTI , 2009). All these programs have variations in their contents, taking into account own aspects of training.

It is important that in worked out curriculum it is summarized the experience of all programs for last 5 years. Some parts are revealed invariants for all teachers of comprehensive school, they help to allow the development of the common information culture of teachers, subject to their level of computer literacy (Tsvetkova, 2009; 2010).

The common curriculum of improvement for comprehensive school teachers consists of 6 typical modules for the preparation of school teachers by the common ICT competences described above. No module exceeds 18 hours of training. To each kind of competence there is a corresponding part of the program of training. It is important to notice, that each module is focused on ICT competence in educational activity. This feature of the curriculum allows a teacher to reach competence in the field of ICT, not on user level, and on a professionally oriented level in the sense of embedding ICT in pedagogical practice.

The common curriculum of improvement for comprehensive school teachers is represented in Table 1. It is oriented at 72 hours of studying. Of them 26 hours are connected with theory and 48 hours with practice. Modules 1, 2 and 3 make a course of the computer literacy and last 36 hours. Modules 4, 5 and 6 make a course of the common information culture of the teacher and it also lasts 36 hours. For groups of teachers possessing computer literacy, it is possible to use the curriculum on the basis of 36 hours of a common information culture course and 36 hours of the additional module 7 (updated modules 1–3). Topics which can be replaced with the additional module 7 are defined on the basis of primary testing of teachers. It is offered on those topics on which a teacher demonstrated competent after testing, to promote his trajectory of development in the ICT school environment with use of the additional module.

The content of each part of training is realized both though lectures and a practical training, in the form of computer practical work and in the form of workshops. Workshops are practical lessons in the form of discussions that do not demand the use of computers. Such lessons are recorded on video and participants can repeat them again and again for better understanding. The purpose of workshops is to search for optimum techniques for carrying out of lessons for various groups of teachers on the basis of using

Table 1

Typical modules of the curriculum "The common ICT competence of teachers"

Modules of the curriculum			Class periods		
			In total	Lectures	Seminars, practical employment (occupations)
1.	**Introduction in ICT**		**4**	**4**	
	1.1	ICT-preparation in the structure of pedagogical activity **(skill 1)**	2	2	
	1.2	Workshop of teachers "Concept of uniform information space of educational establishment, model of its construction, teacher's personal information space". **(skill 2)**	2	2	
2.	**The organization of a teacher workplace with use of computer (skill 3)**		**14**	**4**	**10**
	2.1	The organization of teacher's personal information space and computing work station	4	2	2
	2.2	Introduction in Microsoft Windows / Linux	4		4
	2.3	The general questions of a technique of introduction of digital equipment in teaching and educational process (interactive board, digital laboratory, computer models)	4	2	2
	2.4	Practical work. The tool making computer system (TCS) of professional teacher activity which is a part of the informational space of school (the school schedule, the electronic journal of lessons)	2		2
3.	**Methodical bases of preparation of evident and didactic materials means ICT (skill 4)**		**18**	**6**	**12**
	3.1	Methods of preparation of didactic materials (on example Microsoft Word / Open office)	4	2	2
	3.2	Methods of preparation of didactic materials (on example Microsoft Excel / Open office)	4	2	2
	3.3	Preparation of the distributing materials containing graphic elements	2		2
	3.4	Methods of preparation of evident means and educational-methodical materials (on example Microsoft PowerPoint / Open office)	4	2	2
	3.5	*Practical work "Creation of evident and educational-methodical materials by Microsoft Office/ Open office means "*	4		4
4.	**The Internet in educational activity (skill 5)**		**12**	**4**	**8**
	4.1	Bases of construction of the Internet network. Internet-channels, main principles of work of satellite segment, the uniform Educational Information Space. Educational opportunities of services of the Internet network	1	1	
	4.2	Information search systems in the Internet network in the activity of the teacher and pupils	2		2
	4.3	*Educational resources of the Internet (the review and thematic search)*	2		2
	4.4	Informational protection systems	2	1	1
	4.5	*e-mail of support of educational activity*	3	1	2
	4.6	Legal aspects of the use of the software and Internet – resources in education. License agreements. Free software	2	1	1

To be continued

Continuation of Table 1

Modules of the curriculum			Class periods		
			In total	Lectures	Seminars, practical employment (occupations)
5.	**The Internet for teacher's portfolio (skill 6)**		**12**	**4**	**8**
	5.1	Introduction in technology of creation of Web-sites of educational purpose with tools of site constructor and by the example of HTML language use	4	2	2
	5.2	Network educational communities and projects	2	2	
	5.3	Remote support of educational process. Network association of methodologists	2		2
	5.4	*Practical work "Creation of a breadboard model of a site – teacher's portfolio"*	4		4
6.	**Digital educational resources in pedagogical activity (skill 7–9)**		**12**	**4**	**8**
`	6.1	The review of the digital educational resources executed during realization of Federal target programs **(skill 7)**	4	2	2
	6.2	Workshop of teachers "Designing of a lesson with use of digital educational resources"**(skill 8)**	2	2	
	6.3	Activity of the teacher groups in the school distance training space with students studying at home **(skill 9)**	2		2
	6.4.	*Additional practical work on use digital educational resources (DER) in training the subjects of the curriculum (in groups) on example of the* DER *of the school media library*	4		4
In total:			**72**	**26**	**48**
7.	***Additional Variation modules (Updated skills 1–4)***		***36***	***14***	***22***
	7.1	Specificity of education cooperation the context of work teachers with students. School project teams. Learning ICT-project (in school subject) Workshop of teachers "School ICT center for home work. School Internet club/ICT club for family"	6	4	2
	7.2	Technology organization computer help service in the school as student's ICT-team	6	2	4
	7.3	Technology organization school publishing student's team	6	2	4
	7.4	Technology organization web-studio in school as student's web-team	6	2	4
	7.5	Technology organization school computer club with digital laboratory	6	2	4
	7.6	Technology organization the school video-studio as student's TV-team	6	2	4

ICT resources. Such workshops are significant for searching for new techniques for the preparation of each comprehensive school teacher on the basis of distant technologies and for development of socially useful school projects.

In Table 1 typical modules of the common curriculum of preparation of comprehensive school teachers are presented. Variation themes for the teachers on different school subject are indicated by italics.

The further development of the described curriculum can be realized by the addition of 18 hours of new modules, taking into account new spheres of ICT application to teaching practice. For example modules 1–2 and module 3 in the computer literacy course will soon pass into an area of self-reliant study by teachers if they become proficient in computer literacy.

The decade 2010–2020 becomes a decade of development of information culture of teachers. The major role during this period will be distant education technologies on a basis the Internet, TV-technologies and interactive media resources (I-NetVC, 2009), (MSSTI, 2009). The big developments will be in the computer workplace of the teacher and the digital lab ware. The common ICT culture of teachers will be the mastering of tools of lesson design and a management system for educational processes. The information space of school will be transformed to interschool information system.

All of this will demand comprehensive mastering by teachers of the common ICT competences and system development of their information culture in the uniform information space of education. In this case new modules of training should build-in a flexibly, for each teacher, in the trajectory of development of information culture.

5. The Complex Curriculum of Preparation of Informatics Teachers and Coaches in Olympiad Informatics

Preparation of informatics teachers and coaches in informatics olympiads on the one hand should be based on the state school educational standard on informatics and on the other hand should take into account the modern competition tasks contents of national (Kiryukhin, 2008; 2009) and international (Verhoeff *et al.*, 2006; Kiryukhin and Okulov, 2007) informatics olympiads. The analysis of the currently enforced state school educational standard on informatics in Russia (Kiryukhin, 2010) has suggested a complex curriculum of preparation of informatics teachers on based on their professional ICT competence.

Taking into account the approach described in the paper (Kiryukhin, 2007) defining the contents of the Olympiad on Informatics we will select the following key parts of the complex curriculum of preparation of informatics teachers and coaches in informatics olympiads:

1. Mathematical Fundamentals of Informatics.
2. Developing and Analyzing of Algorithms.
3. Programming Fundamentals.
4. Computer Literacy.
5. Operating Systems.
6. Basis of Programming Technology.
7. Fundamental Methods of Calculations and Modeling.
8. Introduction to Network Technologies.

Taking this into account, the complex curriculum presented below for the preparation of informatics teachers and coaches in informatics olympiads consists of eight sections,

each of which reveal sub-topics. Each topic in turn contains, in more detail, the didactic units revealing key knowledge and skills that allow for each student to draw up an individual trajectory of preparation for the Olympiad in Informatics.

Each didactic unit has a certain level of complexity to which readers will be given guidance. In particular, three levels of complexity are selected, marked as follows:

- didactic unit without "*" – means that it concerns initial level of complexity (the first level), and the knowledge of these didactic units allows pupils to take part in school and municipal stages of the Russian Olympiad in Informatics;
- didactic unit with "*" – means a level of preparation sufficient for successful performance at regional stages of the Russian Olympiad in Informatics (the second level of complexity), provides practical and conceptual level of requirements to the participant of informatics olympiad, allows intelligently to come to solution of competition tasks and provides with it possibility technologically to present the own ideas;
- didactic unit with "**" – means the third level of complexity; additional studying of these didactic units forms key abilities for students for solving competition tasks, opens before the olympiad participant the possibility to show the creative potential and to get solutions of enough complex competition tasks which are offered at a final stage of the Russian Olympiads in Informatics and the IOI, allows the filling of student portfolio achievements with diplomas of winners or prize-winners of the final stages of the Russia Olympiad in informatics and the IOI.

Taking into account the above, the mentioned complex curriculum of preparation of informatics teachers and coaches on informatics olympiads will look like the following.

1. Mathematical fundamentals of informatics
   - 1.1. Functions, relations and sets
     - 1.1.1. *Functions, inverse functions, composition*
     - 1.1.2. *Relations (reflexivity, symmetry, transitivity, equivalence, lexicographical order)*
     - 1.1.3. *Sets (Venn diagrams, complements, Cartesian products)*
     - 1.1.4. *Well-ordered sets **
     - 1.1.5. *Cardinability and countability ***
   - 1.2. Basic geometry
     - 1.2.1. *Point, line, segment, vector, angle*
     - 1.2.2. *Cartesian coordinate system in Euclidean space*
     - 1.2.3. *Euclidean distance*
     - 1.2.4. *Cross-product and scalar product on the plane*
     - 1.2.5. *Triangle, rectangle, polygon*
     - 1.2.6. *Convex polygons*
     - 1.2.7. *Trigonometric functions and formulae **
     - 1.2.8. *Voronoi diagram and Delaunay triangulation ***
   - 1.3. Basic logic
     - 1.3.1. *Logic variables, operations, expressions*
     - 1.3.2. *Truth tables*

1.3.3. *Boolean functions*
1.3.4. *Universal and existential quantification*
1.3.5. *Ways to describe Boolean functions*
1.3.6. *Transformation of logical expressions*
1.3.7. *Normal forms (conjunctive and disjunctive) **
1.3.8. *Minimizing Boolean functions **
1.3.9. *Axiomatic of propositional logic **
1.3.10. *Predicate logic **

1.4. Basics of counting

1.4.1. *Counting arguments:*

- *Sums and product rule*
- *Arithmetic and geometric progressions*
- *Fibonacci numbers*
- *Inclusion-exclusion principle **

1.4.2. *Recurrent relations*
1.4.3. *Matrices and matrix operations **
1.4.4. *Fast multiplication of numbers or matrices ***

1.5. Proof techniques

1.5.1. *Direct proofs*
1.5.2. *Drawer principle*
1.5.3. *Proof by counterexample*
1.5.4. *Proof by contraposition*
1.5.5. *Proof by contradiction*
1.5.6. *Mathematical induction*
1.5.7. *The structure of formal proofs **

1.6. Basics of theory of number

1.6.1. *Prime numbers, Fundamental Theorem of Arithmetic*
1.6.2. *Division with remainder*
1.6.3. *Greatest common divisor*
1.6.4. *Co-primes*
1.6.5. *Divisibility. Residue ring **
1.6.6. *Chinese remainder theorem **
1.6.7. *Primitive roots and discrete logarithms ***

1.7. Basics of algebra

1.7.1. *Polynomials and operations with them. Solving quadratic equations. Viet theorem*
1.7.2. *The general case of Viet theorem. Symmetric polynomials **
1.7.3. *The concept of group ***
1.7.4. *Groups properties ***
1.7.5. *Normal subgroups ***
1.7.6. *Theorems about homomorphism and isomorphism ***

1.7.7. *Using group theory to solve combinatorial problems* **

1.8. Basics of combinatorial calculus

1.8.1. *Permutations, arrangements and combinations:*

- *Basic definitions*
- *Pascal's rule*
- *Binomial theorem*

1.8.2. *Grey codes: subsets, combinations, permutations* *

1.8.3. *Inverse tables* *

1.8.4. *Sets partitioning. Stirling numbers* *

1.8.5. *Parentheses sequences* *

1.8.6. *Connection between parentheses sequences and other combinatorial objects (binary and rooted trees, triangulations etc.)* **

1.8.7. *Estimating numbers of combinatorial objects. Stirling's formula. Corollaries.* **

1.9. Graph theory

1.9.1. *Kinds of graphs*

1.9.2. *Paths and connectivity*

1.9.3. *Operations with graphs*

1.9.4. *Trees*

1.9.5. *Spanning trees*

1.9.6. *Graph coloring*

1.9.7. *Eulerian and Hamiltonian graphs*

1.9.8. *Graph covering and independent sets* *

1.9.9. *Planar graphs and graph layout* *

1.9.10. *Biconnectivity. Bridges, articulation points* *

1.9.11. *Connection between DAGs and order relations. Transitive closure* *

1.9.12. *Bipartite graphs* *

1.9.13. *Flows and networks* *

1.9.14. *Planning network graph* *

1.9.15. *k-connectivity* **

1.10. Basics of probability theory

1.10.1. *Probability and expectation. Axiomatic of probability theory* *

1.10.2. *Total probability lemma and Bayes formula. Conditional probability and expectation* **

1.10.3. *Generating random objects for testing* **

1.10.4. *Approximate optimizing methods* **

1.11. Basics of games theory

1.11.1. *Game, the result of the game*

1.11.2. *Basic games and strategies*

1.11.3. *Sprague-Grundy function* *

1.11.4. *Matrix games* **

1.12. Linear programming

1.12.1. *Linear programming problems. Geometrical interpretation* *

1.12.2. *Basic methods of solving linear programming problems: simplex method, table interpretation* **

1.12.3. *Duality* **

1.12.4. *Examples of linear programming problems: maximum flow, assignment problem, shortest path* **

1.12.5. *Discrete linear programming* **

1.13. Basics of mathematical analysis

1.13.1. *Derivative and integral. Evaluating the area* *

1.13.2. *Green formula* **

1.14. Automatons and grammars

1.14.1. *Finite automaton* **

1.14.2. *Connection between finite automatons and grammars* **

1.14.3. *Using finite automatons* **

1.14.4. *Normal forms* **

2. Developing and analyzing algorithms

2.1. Algorithms and their properties

2.1.1. *The concept of algorithm*

2.1.2. *Properties of algorithms*

2.1.3. *Non-formal notation of algorithms*

2.2. Data structures

2.2.1. *Basic data structures*

2.2.2. *Sets*

2.2.3. *Sequences*

2.2.4. *Lists*

2.2.5. *Non-directed graphs*

2.2.6. *Directed graphs*

2.2.7. *Trees*

2.2.8. *Heap and RMQ/RSQ tree**

2.2.9. *Fenwick trees and their n-dimensional implementation* *

2.2.10. *Balanced trees* *

2.2.11. *Hash tables and associative arrays* *

2.2.12. *Trie* **

2.2.13. *Suffix tree* **

2.3. Basic algorithmic analysis

2.3.1. *Big O notation*

2.3.2. *Complexity sets*

2.3.3. *Asymptotic analysis of upper and average complexity bounds*

2.3.4. *Balance between memory and run-time complexity* *

2.3.5. *Using recurrent relations to analyze recurrent algorithms* *

- 2.3.6. *NP-complexity* **
- 2.3.7. *Computability* **
- 2.3.8. *Universal algorithms and self-applicability problem* **
- 2.3.9. *Matroid theory* **

2.4. Algorithmic strategies

- 2.4.1. *Brute force*
- 2.4.2. *Greedy algorithms*
- 2.4.3. *Divide et impere* *
- 2.4.4. *Backtracking* *
- 2.4.5. *Heuristics* *
- 2.4.6. *Branch-and-bound method* **
- 2.4.7. *Simulated annealing* **
- 2.4.8. *Four Russians speed-up* **

2.5. Recursion

- 2.5.1. *The concept of recursion*
- 2.5.2. *Recursive mathematical functions*
- 2.5.3. *Simple recursive functions*
- 2.5.4. *Implementing recursion*
- 2.5.5. *Divide et impere* *
- 2.5.6. *Backtracking* *

2.6. Basic computational algorithms

- 2.6.1. *Simple numeric algorithms*
- 2.6.2. *Classical combinatorial algorithms*
- 2.6.3. *Subset algorithms: generating all, generating next and previous, obtaining number and obtaining by number*
- 2.6.4. *Combinations and permutations algorithms: generating all, generating next and previous, obtaining number and obtaining by number*
- 2.6.5. *Linear and binary search*
- 2.6.6. *Quadratic sorting algorithms (selection, insertion)*
- 2.6.7. *Counting sorting*
- 2.6.8. *O(N log N) sorting (Quicksort, heap sort, merge sort)* *
- 2.6.9. *Digital sort* *
- 2.6.10. *Obtaining word number in lexicographically ordered set of its letters permutations* *
- 2.6.11. *Arbitrary-size arithmetic* *

2.7. *Numeric algorithms*

- 2.7.1. *Integer factorization*
- 2.7.2. *Eratosthenes sieve*
- 2.7.3. *Euclid's algorithm*
- 2.7.4. *Extended Euclid's algorithm. Implementing without division**
- 2.7.5. *Solving linear modular equations using Euclid's algorithm* *
- 2.7.6. *Effective implementation of Eratosthenes sieve (O(n))* *

2.7.7. *Gaussian method and matrix inversions* **
2.7.8. *Fast power calculations. RSA* **
2.7.9. *Discrete logarithms* **
2.7.10. *Roots modulo n* **
2.7.11. *Effective primality test* **
2.7.12. *Fast factoring algorithms. Rho heuristics* **
2.7.13. *Berlekamp's algorithm* **

2.8. String processing

2.8.1. *Search for substring. Naïve method*
2.8.2. *Linear substring algorithms (Knuth–Morris–Pratt, Z-function)* *
2.8.3. *Cyclic strings* *
2.8.4. *Editorial distance and optimal alignment* *
2.8.5. *Boyer–Moore algorithm* **
2.8.6. *Aho–Corasick algorithm* **
2.8.7. *Building suffix trees* **
2.8.8. *Digital suffix sorting* **
2.8.9. *Prime strings, string factorization* **
2.8.10. *Building suffix automatons* **

2.9. Graph algorithms

2.9.1. *Breadth- and depth-first search*
2.9.2. *Implementation methods of BFS (with queue or without)*
2.9.3. *Connectivity test*
2.9.4. *Shortest path problem in weighed graphs*
2.9.5. *Topological sort, strong connectivity and order diagram* *
2.9.6. *Negative cycles – criterion, search algorithm* *
2.9.7. *Time synchronization and linear inequality system* *
2.9.8. *Eulerian cycle search (incl. lexicographically minimal)* *
2.9.9. *Transitive closure* *
2.9.10. *Weighted minimal spanning trees* *
2.9.11. *Search for 2-connectivity components, bridges and articulation points using DFS* *
2.9.12. *Bipartite matching and minimum vertex cover search* *
2.9.13. *Searching for maximal flow* **
2.9.14. *Searching for maximal flow of minimum cost* **
2.9.15. *Assignment problem – Hungarian method. Connection between Hungarian method, minimal-cost maximal-flow and Dijkstra algorithm* **
2.9.16. *Fast algorithms for maximal-flow problem:* $O(N^3)$ **

2.10. Dynamic programming

2.10.1. *The concept of dynamic programming. Recursive and linear implementation.*
2.10.2. *Monotone-direction problems*
2.10.3. *Backpack problem*

2.10.4. *Eliminating extra parameters **
2.10.5. *Building the way of solution using dynamic table **
2.10.6. *Common scheme of dynamic programming **
2.10.7. *Subset and profile dynamic programming **
2.10.8. *Broken-profile dynamic programming **

2.11. Game theory algorithms

2.11.1. *Dynamic programming and brute-force. DAG games **
2.11.2. *Retro-analysis. Effective implementation ***
2.11.3. *Fast Boolean evaluation. Using it in game analysis ***
2.11.4. *Position scoring. Alpha-beta pruning ***

2.12. Geometrical algorithms

2.12.1. *Test for coincidence of points, rays, lines and segments*
2.12.2. *Storing methods for points, lines and segments*
2.12.3. *Calculating distance between objects on the plane **
2.12.4. *Intersecting segments on the plane **
2.12.5. *Calculating polygon area using vertex coordinates. Pick's formula **
2.12.6. *Convex hull algorithms **
2.12.7. *Circles on the plane. Intersecting circles and other objects **
2.12.8. *Determining if the point is inside polygon **
2.12.9. *Scanning line method **
2.12.10. *Half-plane method ***
2.12.11. *Boundary detour method ***
2.12.12. *Effective algorithm of searching for two nearest points ***
2.12.13. *Effective algorithm for Voronoi diagram ***

3. Programming basics

3.1. Programming languages

3.1.1. *Types of languages*
3.1.2. *Procedure languages*
3.1.3. *Syntax and semantics of high-level languages*
3.1.4. *Formal syntax description methods: Backus–Naur form**
3.1.5. *Object oriented languages **

3.2. Programming constructions

3.2.1. *Variables, types, expressions and assignments*
3.2.2. *Input/output basics*
3.2.3. *Conditions and cycles*
3.2.4. *Functions and parameters*
3.2.5. *Structured decomposition **

3.3. Variables and data types

3.3.1. *Data types as set of values and operations*
3.3.2. *Declaration properties (linking, visibility area, blocks and lifetime)*
3.3.3. *Type matching*

3.4. Data structure types

3.4.1. *Primitives*
3.4.2. *Arrays*
3.4.3. *Record*
3.4.4. *Strategies of selection of appropriate data structure*
3.4.5. *Data storage in memory **
3.4.6. *Static, automatic and dynamic memory allocation **
3.4.7. *Pointers and references **
3.4.8. *Linked structures **
3.4.9. *Methods of implementing stacks, queues and hash tables **
3.4.10. *Methods of implementing graphs and trees **

3.5. Abstraction mechanisms

3.5.1. *Procedures, functions and iterators as abstraction mechanisms*
3.5.2. *Parameterization mechanisms (references and values)*
3.5.3. *Units in programming languages*
3.5.4. *Parameterized types ***

3.6. Fundamental programming species

3.6.1. *Task solving strategies*
3.6.2. *Role of algorithms in problem-solving process*
3.6.3. *Strategies of implementing algorithms*
3.6.4. *Implementing recursion*
3.6.5. *Testing strategies **

4. IT facilities

4.1. Digital logic

4.1.1. *Logical schemes*
4.1.2. *Scales of notation*
4.1.3. *Computer arithmetic*

4.2. Data representation

4.2.1. *Bits, bytes and words*
4.2.2. *Numeric representation **
4.2.3. *Fixed and floating point **
4.2.4. *Signed numbers representation **
4.2.5. *Non-numeric data representation, character codes, graphic data **
4.2.6. *Arrays and records **

4.3. Computer engineering principles

4.3.1. *Von Neumann principle*
4.3.2. *Control block: decoding and executing instructions*
4.3.3. *Instruction set and types (data manipulation, control, input/output)*
4.3.4. *Instruction formats **
4.3.5. *Addressing methods **
4.3.6. *Procedure calls and return **

4.3.7. *Input/output and interrupts **

4.4. Memory

4.4.1. *Operations with memory*
4.4.2. *Memory hierarchy*
4.4.3. *Data encoding, compression and integrity **
4.4.4. *Cache **

4.5. Communications

4.5.1. *I/O basics*
4.5.2. *External memory, external devices*
4.5.3. *Network technologies*
4.5.4. *DMA **

5. Operating systems

5.1. Operating systems basics

5.1.1. *Role and tasks of operating systems*
5.1.2. *Typical operating system functions*
5.1.3. *Directories: contents and structure*
5.1.4. *Naming, search, access, backup*

5.2. Basic functions of operating systems

5.2.1. *Abstractions, processes and resources*
5.2.2. *Device organization*
5.2.3. *Protection, access and authentication*

5.3. Memory control

5.3.1. *Physical memory and memory-controlling hardware*
5.3.2. *Segment and paged memory model. Flat model **
5.3.3. *Caching **

6. Basics of programming technologies

6.1. Programming facilities and space

6.1.1. *Programming space*
6.1.2. *Testing facilities **

6.2. Software quality assurance

6.2.1. *Basics of testing, testing plan **
6.2.2. *White box and black box methods **
6.2.3. *Integrated testing, element testing, system testing, quality assurance **
6.2.4. *Stress-testing **

7. Calculation methods and modeling

7.1. Basics of computational mathematics

7.1.1. *Basic methods of computational mathematics*

- *Finding roots and values of function **
- *Finding perimeter, area and volume **

7.1.2. *Grid and step methods **
7.1.3. *Floating point arithmetic ***
7.1.4. *Precision loss, stability, convergence ***

7.2. Modeling basics

7.2.1. *Concepts of model and modeling*
7.2.2. *Basic model types*
7.2.3. *Components of computer model and description methods: input and output variables, state variables, entry/exit functions, time shift*
7.2.4. *Stages and specials of building computer model*
7.2.5. *Basic stages of using computer models during solving problems*

8. Network technologies

8.1. Networks and communications

8.1.1. *Network cards and network devices*
8.1.2. *Data transfer media*
8.1.3. *Network architecture*
8.1.4. *Passwords and access control mechanisms*
8.1.5. *Service quality: performances, restoring after fault**

8.2. Wireless networks

8.2.1. *Specific problems of wireless and mobile computing*
8.2.2. *Installing programs on wireless and mobile computers*
8.2.3. *Wireless networks and links*

It is necessary to notice, that for the preparation of informatics teachers and coaches for informatics olympiads when using the described curriculum it is necessary to consider, that studying is presented in topics and didactic units should be going from simple to difficult. In particular, at first it is necessary to master the first level of complexity, and then to go further. Moreover, if teachers or coaches do not put before themselves the task of preparation of students for a successful performance at final stage of the Russian Olympiad in Informatics or the IOI it is enough for such teachers or coaches to master only the first level of complexity. It is important as such informatics teachers or coaches should have more knowledge even if they only teach to the first level, but not all of them can be in a condition to master higher level of complexity.

Only the most prepared teachers and coaches can be promoted to the third level of complexity. Here it is important to remember what evaluation of level of any informatics teacher-coaches preparation can be done by means of their pupils' achievements at informatics olympiads. If pupils get medals at the IOI then their informatics teacher-coaches will make gold fund of coaches of the country which will be capable to prepare students for victories on the national and international informatics olympiads.

6. Conclusion

Preparation of teachers according to the allocated levels of ICT competence plays the important role in development of informatization of each country. The curriculums of

preparation of teachers offered in this paper are applicable in other countries where the course of informatics and ICT (computer science) is included in the national education program on informatics and the national informatics olympiads are organized. Russian experience in this area can also be useful for the systematization of programs of school teachers preparation on informatics and ICT in any country. The problem of improving of ICT competence of teachers should be solved in a complex that will guarantee not only high level of studying of informatics at comprehensive schools, but also successes of students at the IOI.

References

Bibigon (2009). *Children's Social Network "Bibigon"* (in Russian).
http://www.bibigon.ru

CM (2009). *Open School Encyclopedia by Cyril & Methodius* (in Russian).
http://megabook.ru

EGE (2009). *Informational System of State Certification* (in Russian).
http://www.ege.ru

FSIOR (2008). *Federal System of Informational Educational Resources* (in Russian).
http://www.fsior.edu.ru

I-NetVC (2009). *Open Interactive Educational Internet-Video Channel for Teachers* (in Russian).
http://www.binom.vidicor.ru

Intel (2003). *The Common Project of the Companies Intel and Microsoft "Teachers to the Future"* (in Russian).
http://www.intel.com/cd/corporate/education/emea/rus/elem_sec/programs/teach

Kiryukhin, V.M. (2007). The modern contents of the Russian national olympiads in informatics. *Olympiads in Informatics*, 1, 90–104.

Kiryukhin, V., Okulov, S. (2007). *Методика решения задач по информатике. Международные олимпиады* (Methods of Problem Solving in Informatics. International Olympiads). LBZ (BINOM. Knowledge Lab.), Moscow (in Russian).
http://www.lbz.ru/katalog/products/literatura-dlja-shkol/informatika/olimpijskie-vysoty/metodika-reshenija-zadach-po-informatike-699

Kiryukhin, V. (2008). *Информатика. Всероссийские олимпиады.* Выпуск 1 (Informatics. Russian Olympiads. Issue 1). Prosveschenie, Moscow (in Russian). http://www.prosv.ru/book.aspx?ob_no=209&d_no=13828&ltype=9577&subject=1467

Kiryukhin, V., Tsvetkova, M. (2008). Роль государственного стандарта в определении содержания олимпиад по информатике (State standard role in definition of the olympiads in informatics contents). All-Russian olympiads of secondary school students. *Information-Methodical Magazine*, 3 (in Russian).

Kiryukhin, V. (2009a). *Информатика. Международные олимпиады.* Выпуск 1 (Informatics. International Olympiads. Issue 1). Prosveschenie, Moscow (in Russian). http://www.prosv.ru/book.aspx?ob_no=209&d_no=17310&ltype=9577&subject=1467

Kiryukhin, V. (2009b). *Информатика. Всероссийские олимпиады.* Выпуск 2 (Informatics. Russian Olympiads. Issue 2). Prosveschenie, Moscow (in Russian). http://www.prosv.ru/book.aspx?ob_no=209&d_no=17575&ltype=9577&subject=1467

Kiryukhin, V. (2010). Influence of the state school education standard on results of performance of the Russian pupils at the IOI. *Olympiads in Informatics*, 4, 15–29.

KMS (2009). *Informational Content Training System for Secondary Schools "KM-School"* (in Russian).
http://www.km-school.ru

MSSTI (2009). *On-Line Methodical Support System of Teachers in Informatics* (in Russian).
http://www.metodist.LBZ.ru

Nachalka (2009). *Informational Content Training System for Primary Schools "Nachalka"* (in Russian).
http://nachalka.info

OWPI (2008). *Open Web-Practice in Informatics* (in Russian).
http://webpractice.cm.ru

RusOI (2009). *Site of the All Russian Secondary School Students Olympiads in Informatics* (in Russian).
http://www.info.rusolymp.ru

SCNS (2009). *Social Contents Network for Schools* (in Russian).
http://www.school-club.ru

SNT (2009). *Social Network for Teachers* (in Russian).
http://www.it-n.ru

Teleschool (2009). *Distance Training System "Teleschool"* (in Russian)
http://www.internet-school.ru

Tsvetkova, M., *et al.* (2005). *Межшкольные методические центры. Организационная деятельность* (Interschool methodical centers. Organizational activity). World Bank, National training foundation, Moscow (in Russian).

Tsvetkova, M.S. (2008). Методические основы наставничества в олимпиадной информатике в России (Methodical bases of tutorship in Olympiad informatics in Russia). All-Russian olympiads of secondary school students. *Information-Methodical Magazine*, 2 (in Russian).

Tsvetkova, M., Kuris, G. (2008). *Виртуальные лаборатории по информатике в начальной школе.* Серия: "ИКТ в работе учителя" (Virtual laboratories on informatics in a primary school. Series "ICT in Teacher Activity") LBZ (BINOM. Knowledge Lab.), Moscow (in Russian).
http://www.lbz.ru/katalog/products/literatura-dlja-shkol/informatizacija-obrazovanija/virtualnye-laboratorii-po-informatike-vnachalnoj

Tsvetkova, M. (2009). *Модели непрерывного информационного образования.* Серия: "Информатизация образования" (Models of continual information education. Series "Informatization of education"). LBZ (BINOM. Knowledge Lab.), Moscow (in Russian). http://www.lbz.ru/ katalog/products/literatura-dlja-shkol/informatizacija-obrazovanija/modeli-nepreryvnogo-informacionnogo-obrazovanija

Tsvetkova, M. (2010a). *Информационная активность педагогов* (Information activity of teachers). LBZ (BINOM. Knowledge Lab.), Moscow (in Russian).

Tsvetkova, M. (2010b). The olympiad in informatics as a part of the state program of school informatization in Russia. *Olympiads in Informatics*, 4, 120–133.

Verhoeff, T., Horváth, G., Diks, K., Cormack, G. (2006). A proposal for an IOI Syllabus. *Teaching Mathematics and Computer Science*, IV (1), 193–216.
http://www.win.tue.nl/wstomv/publications/ioi-syllabus-proposal.pdf

WE (2007). *Digital Educational Resources* (in Russian).
http://www.windows.edu.ru

V.M. Kiryukhin is professor of the Informatics and Control Systems Department at the National Research Nuclear University MEPhI. He is the chairman of the Federal Methodical Commission on Informatics which is responsible in Russia for carrying out the national olympiads in informatics. He is the author of many papers and books in Russia on development of olympiad movements in informatics and preparations for the olympiads in informatics. He is the exclusive representative who took part at all IOI from 1989 as a member of the IOI International Committee (1989–1992, 1999–2002) and the Russian team leader. He received the IOI Distinguished Service Award at IOI 2003, the IOI Distinguished Service Award at IOI 2008 as one of the founders of the IOI making his long term distinguished service to the IOI from 1989 to 2008 and the medal "20 Years since the First International Olympiad in Informatics" at the IOI 2009.

M.S. Tsvetkova, PhD in pedagogic science, associate professor of Academy of Improvement of Professional Skill of Educationalists, prize-winner of competition "The Teacher of Year of Moscow" (1998), main expert of state projects of school education informatization in the Ministry of Education of the Russian Federation (2001–2005), the expert of the World bank project "Informatization of Education System". Since 2002 she is a member of the Central Methodical Commission of the Russian Olympiad inInformatics, the pedagogic coach of the Russian team on the IOI.

Olympiads in Informatics, 2010, Vol. 4, 52–66

© 2010 *Institute of Mathematics and Informatics, Vilnius*

Algorithms without Programming

Marcin KUBICA, Jakub RADOSZEWSKI
Institute of Informatics, University of Warsaw
Banacha 2, 02-097 Warsaw, Poland
e-mail: {kubica,jrad}@mimuw.edu.pl

Abstract. Programming contests are generally intended to popularize computer science, particularly algorithmic thinking and programming skills. However, such contests are addressed to a limited group of pupils – those that can write programs and know at least some programming techniques. But how can we attract those pupils that know nothing about programming or algorithms? We argue that one of the ways to do it is to present tasks that require some algorithmic thinking, but no programming. The paper contains several such example tasks together with their analysis.

Key words: algorithmic tasks, programming contests.

1. Introduction

One of the main goals of various programming contests is popularization of programming, or more generally, computer science, among youngsters. However, many such contests are addressed to pupils that are already familiar with programming (Verhoeff *et al.*, 2006; Diks *et al.*, 2007). Usually the focus is on correctness and efficiency. One could say that they promote knowledge of programming techniques and algorithmics. But how can we bring the attention of young pupils to programming in the first place? The contests that we mentioned may be too much for the first step. Before introducing a programming language, we could bring students' attention to very simple algorithms, without formulating them as programming tasks.

It is not surprising that the best candidates for such contests are pupils interested in math, since computer science is a younger sibling of mathematics. In many computer science problems one can find interesting mathematical sub-problems. Also, many mathematical problems can be solved with the aid of computers. There is a wide spectrum of problems with purely mathematical problems on one end, and purely algorithmic problems on the other end. Many examples of intermediate tasks can be found in such contests as: the Beaver competition (`www.bebras.org`; Dagienė, 2006; 2010), the Ugāle team competition (Opmanis, 2009), and Project Euler (`projecteuler.net`). The problem how to choose good tasks for a contest has brought attention of many researchers, e.g. (Dagienė and Futschek, 2008; Diks *et al.*, 2008; Verhoeff *et al.*, 2006; Forišek, 2006). In our opinion, offering tasks or puzzles requiring various levels of algorithmic thinking is a good way to popularize learning programming among young pupils. This opinion is based on two years of cooperation with a monthly journal *Delta*

(`www.mimuw.edu.pl/delta`) addressed to secondary school pupils and dedicated to mathematics, computer science, physics and astronomy.

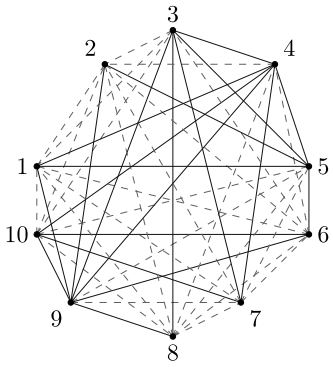
In the following sections we present a couple of simple problems formulated in a purely mathematical manner. Each problem is followed by a hint how to solve it, a solution and some methodological comments. All the problems require elements of algorithmic thinking. However one does not have to write programs, or even formally define an algorithm. These problems have already been presented (Radoszewski, 2009) in an article addressed to secondary-school pupils. The key properties of these tasks that make them suitable for initial education of algorithmic skills are summarized in Conclusions.

The authors would like to thank anonymous referees for many helpful comments.

2. Uni-Color Triangles

2.1. *Problem*

You are given 10 points in the plane, such that no 3 of them are collinear. Each pair of points is connected by a green (dashed) or blue (continuous) line – see the following figure.



How many uni-color triangles, with vertices at the given points, are present in the figure?

Hint: You will find the following table useful: the cell located in the ith row and jth column denotes the color (b – blue, g – green) of the line connecting points i and j.

	1	2	3	4	5	6	7	8	9	10
1		g	g	b	b	g	g	g	b	g
2	g		g	g	b	g	g	g	b	g
3	g	g		b	b	g	b	b	b	g
4	b	g	b		b	g	b	g	b	b
5	b	b	b	b		b	g	g	g	g
6	g	g	g	g	b		g	g	b	b
7	g	g	b	b	g	g		g	g	b
8	g	g	b	g	g	g	g		b	g
9	b	b	b	b	g	b	g	b		b
10	g	g	g	b	g	b	b	g	b	

2.2. *Solution*

The total number of triangles with vertices it the given 10 points is:

$$\binom{10}{3} = \frac{10 \cdot 9 \cdot 8}{1 \cdot 2 \cdot 3} = 120.$$

Instead of counting uni-color triangles, we can count multi-color triangles (i.e., with exactly two sides of equal color), and subtract the number of such triangles from 120.

Each multi-color triangle has exactly **two** vertices in which two sides of different colors meet. For each of the 10 points, let us count the number of green and blue edges incident to it – the results are shown in the following table:

Point	1	2	3	4	5	6	7	8	9	10
Blue edges	3	2	5	6	5	3	3	2	7	4
Green edges	6	7	4	3	4	6	6	7	2	5

Now, we can multiply the numbers of green and blue edges incident to the given points, and sum the products:

$$\begin{aligned} 3 \cdot 6 + 2 \cdot 7 + 5 \cdot 4 + 6 \cdot 3 + 5 \cdot 4 + 3 \cdot 6 + 3 \cdot 6 + 2 \cdot 7 + 7 \cdot 2 + 4 \cdot 5 = \\ = 18 + 14 + 20 + 18 + 20 + 18 + 18 + 14 + 14 + 20 = 174. \end{aligned}$$

This way each multi-color triangle is counted twice. Hence, the number of uni-color triangles in the figure equals:

$$120 - \frac{174}{2} = 33.$$

2.3. *Methodological Comments*

Let us denote the number of vertices by n, here we have $n = 10$. A naive solution requires checking $\Theta(n^3) = \binom{n}{3}$ triangles, that is 120 triangles for $n = 10$. This is quite a large value – it suggests that it could be worthwhile to look for a more efficient solution. The combinatorial observation above reduces the problem to counting $\Theta(n^2) = \binom{n}{2}$

edges, plus processing n vertices. It is the optimal solution, since the size of the input is $\Theta(n^2)$. Although there is nothing about time complexity in the problem statement, the difference between computations that can be easily performed by hand and computations too complex to do manually corresponds to the difference between the optimal and inefficient solutions. This problem was presented at the 4th Polish Olympiad in Informatics, 1996/1997 (Guzicki, 1997).

3. Continuous Subsequences Divisible by 13

3.1. *Problem*

Does the following sequence of numbers:

$$(1,\ 1,\ 9,\ 7,\ 12,\ 4,\ 12,\ 5,\ 8,\ 2,\ 7,\ 2,\ 10,\ 2,\ 3)$$

contain a non-empty, continuous subsequence, whose sum is divisible by 13? If so, what is the number of such subsequences?

Hint: First, count the length of the given sequence. Then, count the sums of elements in consecutive prefixes (leading fragments) of the given sequence.

3.2. *Solution*

Observe that the sum of elements in any continuous subsequence equals the sum of elements in the prefix ending at the end of the subsequence, minus the sum of elements in the prefix ending one position before the beginning of the subsequence, see Fig. 1. Therefore, the first step should be to compute the sums of elements for all the prefixes of the given sequence (including zero for the empty prefix).

If for some continuous subsequence, the sum S of elements in it is divisible by 13, then the sums S_1 and S_2 of elements in the corresponding prefixes give the same remainders when divided by 13. Conversely, if there are two different prefixes, for which the sums of their elements give the same remainders modulo 13, then the sum of elements in the corresponding continuous subsequence is divisible by 13. The given sequence contains 15 elements, consequently, by the Dirichlet's principle (also known as the *pigeonhole principle*), there must exist two such prefixes, that the sums of their elements give the same remainder modulo 13. Hence, the answer to the first question is yes.

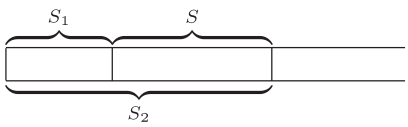


Fig. 1. Sum of elements in the continuous subsequence expressed using sums of elements in the corresponding prefixes.

Now, let us count the number of such continuous subsequences. Let us group the prefixes of the given sequence by the remainders (modulo 13) of the sums of their elements. Assume that for some remainder we have k prefixes in one group. How many continuous subsequences, with the sums of elements divisible by 13, do they generate? Exactly the number of (unordered) pairs of different such prefixes, that is:

$$\binom{k}{2} = \frac{k(k-1)}{2}.$$

The sums of elements of the prefixes are as follows:

Given sequence		1	1	9	7	12	4	12
Prefix length	0	1	2	3	4	5	6	7
Sum of elements	0	1	2	11	18	30	34	46
Remainder modulo 13	0	1	2	11	5	4	8	7

Given sequence	5	8	2	7	2	10	2	3
Prefix length	8	9	10	11	12	13	14	15
Sum of elements	51	59	61	68	70	80	82	85
Remainder modulo 13	12	7	9	3	5	2	4	7

If we group equal remainders, we obtain the following results:

Remainder	0	1	2	3	4	5	6	7	8	9	10	11	12
Number of prefixes	1	1	2	1	2	2	0	3	1	1	0	1	1
Generated cont. subseq.	0	0	1	0	1	1	0	3	0	0	0	0	0

Hence, the total number of continuous subsequences, for which the sum of their elements is divisible by 13, equals 6.

3.3. *Methodological Comments*

This problem illustrates some elements of dynamic programming. First, the original problem is reduced to a (parameterized) problem of counting the number of prefixes for which the sum of their elements, modulo 13, equals a given remainder. This problem, in turn, is reduced to calculating the sums of elements for all the prefixes of the given sequence.

It also informally illustrates the issue of time complexity. Let us denote by n the length of the given sequence, here $n = 15$. A naive solution requires checking $\Theta(n^2)$ continuous subsequences, and processing a subsequence takes $\Theta(n)$ time in average – this gives $\Theta(n^3)$ total time complexity. The first observation reduces the time needed to calculate the sum of elements in a continuous subsequence to $O(1)$, and hence the whole solution requires $\Theta(n^2)$ time. Finally, the presented solution requires $\Theta(n)$ time and is easy to perform by hand.

4. Subsequences Divisible by 5

4.1. *Problem*

What is the number of subsequences of the following sequence:

$$(3,\ 2,\ 3,\ 4,\ 1,\ 1,\ 2,\ 3)$$

such that the sum of their elements is divisible by 5? Subsequences that are equal, but formed from elements at different positions, should be counted separately, e.g. the sequence $2, 1, 2$ should be counted twice, since there are 2 different ways to obtain it.

Hint: For each prefix of the given sequence, count the number of its subsequences, for which the sum of elements modulo 5 equals, respectively, 0, 1, 2, 3 and 4.

4.2. *Solution*

The hint tells us what should be calculated, but it does not say how. First, let us consider a couple of the shortest prefixes of the given sequence. We start with the empty prefix. Of course, it has only one subsequence, the empty one, and the sum of its elements equals zero. Then there is a single element prefix (3). We can either take this element or not. Hence, there is one subsequence with the sum of elements equal 0 (the empty one), and one with the sum of elements equal 3. Now, let us consider the prefix $(3, 2)$. There are four subsequences of such a prefix: the empty one, (3), (2), and $(3, 2)$. The first two have already been considered, as subsequences of (3). The other two contain the last element of the prefix, as their last element.

The above observation can be generalized. All the subsequences of a given nonempty prefix can be divided into two groups: these containing the last element of the prefix, and these not containing it. Let us denote the last element of the prefix by x. Then, the number of subsequences of the prefix, for which the sum of elements modulo 5 equals k, is equal to the sum of:

- the number of subsequences of a prefix shorter by one position, for which the sum of elements modulo 5 equals k, and
- the number of subsequences of a prefix shorter by one position, for which the sum of elements modulo 5 equals $(k - x) \bmod 5$.

Using this rule, one can calculate the number of respective subsequences for all the prefixes. The following table shows the results for the given sequence:

Given sequence		3	2	3	4	1	1	2	3
Prefix length	0	1	2	3	4	5	6	7	8
$\equiv 0 \pmod 5$	1	1	2	3	4	7	13	26	51
$\equiv 1 \pmod 5$	0	0	0	1	2	6	13	26	52
$\equiv 2 \pmod 5$	0	0	1	1	4	6	12	25	50
$\equiv 3 \pmod 5$	0	1	1	3	3	7	13	26	52
$\equiv 4 \pmod 5$	0	0	0	0	3	6	13	25	51

Finally, we have 51 subsequences (including the empty one) of the given sequence, for which the sum of elements is divisible by 5. But what are they?

Enumerating all the 51 subsequences would be too tedious. Nevertheless, let us deal with this problem for a shorter prefix: $(3, 2, 3, 4)$. From the table we know that there are four such sequences. To find all of them, we should trace back, how the number 4 has been obtained. It was obtained as the sum $4 = 3 + 1$, where 3 and 1 are the numbers of subsequences of $(3, 2, 3)$ giving remainders, respectively, 0 and 1. Recursively, we can trace back how the numbers 3 and 1 have been obtained. The latter one corresponds to the sequence $(3, 3, 4)$. The other three are subsequences of $(3, 2, 3)$. Two of them are subsequences of $(3, 2)$ – clearly, they are $()$ and $(3, 2)$. The last one is a subsequence of $(3, 2)$, giving the remainder 2, extended by 3 – that is $(2, 3)$. Hence, all the four subsequences are: $()$, $(3, 2)$, $(2, 3)$ and $(3, 3, 4)$.

4.3. *Methodological Comments*

This problem is a clear illustration of dynamic programming. First, the hint shows how the original problem can be reduced to a number of simpler sub-problems. Then, knowing the recursive dependencies between these sub-problems, one can calculate the table with their results. Finally, it also illustrates a more general property of dynamic programming: knowing the number of solutions is half way to knowing the actual solution.

Without the dynamic programming, it is not feasible to do this task "by hand". The number of all possible subsequences that have to be considered is exponential. However, dynamic programming reduces the time complexity to $\Theta(n)$, where n is the length of the given sequence.

5. Coins

5.1. *Problem*

You are given 11 coins of the following values:

$$7,\ 300,\ 35,\ 83,\ 1,\ 17,\ 2,\ 1,\ 17,\ 170,\ 5.$$

What is the smallest (positive integer) amount of money, that cannot be paid using the coins?

Hint: Let us order the coins according to their value:

$$1,\ 1,\ 2,\ 5,\ 7,\ 17,\ 17,\ 35,\ 83,\ 170,\ 300.$$

Now, consider the sets of amounts of money that can be paid using consecutive prefixes of the ordered sequence of coins.

5.2. *Solution*

Let us follow the hint. Using just the first two coins, it is possible to pay any amount from 0 to 2. If we include the third coin, 2, the range of amounts that can be paid extends to from 0 to 4. What if we include coin 5? Can we pay any amount of money from 0 to $4 + 5 = 9$? Yes, for any amount from 0 to 4, we do not have to use coin 5, and for any amount x from 5 to 9, we use coin 5 and the amount $x-5$ can be paid using the remaining coins.

In general, if the range of amounts that can be paid using the first k coins is from 0 to r, and the following coin has value x, then either:

- $x \leqslant r+1$ and it is possible to pay any amount from 0 to $r + x$ using the first $k+1$ coins, or
- $x > r + 1$ and it is not possible to pay the amount of $r + 1$, since x and all the following coins have greater values.

Using this observation, we obtain the following sequence of included coins and ranges of amounts that can be paid:

Coins	1	1	2	5	7	17	17	35	83
Maximum amount	1	2	4	9	16	33	50	85	168

And finally, $170 > 168 + 1 = 169$, so the smallest amount that cannot be paid using the given coins is 169.

Of course, if we do not find such a coin x, that $x > r + 1$, then the smallest amount that cannot be paid is simply the sum of all coins' values plus one.

5.3. *Methodological Comments*

This problem resembles the classical problem: how to pay a given amount of money using the coins from the given set. Of course, as a side effect we obtain information about the amounts that cannot be paid. This would be an application of dynamic programming (again). The running time of such a solution is $\Theta(n \cdot S)$, where n is the number of coins and S is the sum of their values. It is simple to perform, but tedious.

On the other hand, the solution presented here is in a way greedy. Moreover, it is much faster – it runs in $\Theta(n)$ time.

6. Encyclopedia

6.1. *Problem*

Your bookshelf contains 12 volumes of encyclopedia. Their order has become quite random:

$$11,\ 1,\ 10,\ 4,\ 3,\ 2,\ 8,\ 7,\ 12,\ 6,\ 9,\ 5.$$

In one move you can take out one volume and insert it anywhere in the row of volumes (shifting some volumes if needed). What is the minimal number of moves necessary to order the volumes from 1 to 12, left to right?

Hint: What is the largest set of volumes that may be left and *not moved at all*?

6.2. *Solution*

Any set of volumes that may be left and not moved at all must obviously form an increasing subsequence of the given sequence of volumes. On the other hand, if we choose any increasing subsequence of the given sequence, and decide not to move these volumes, we can insert all the remaining volumes at their correct positions, so that all the volumes are ordered. Hence, the requested minimal number of moves equals the length of the given sequence, minus the length of its longest increasing subsequence.

It remains to show how to find the length of the longest increasing subsequence of the given sequence of volumes:

$$(11,\ 1,\ 10,\ 4,\ 3,\ 2,\ 8,\ 7,\ 12,\ 6,\ 9,\ 5).$$

The main idea of the solution is to calculate, for each element of the sequence, the length of the longest increasing subsequence ending at that element. Then it suffices to take the maximum of such lengths.

For the first two volumes the requested result is 1. The third volume has number 10, so it can be added at the end of a single-element increasing subsequence that ends at volume number 1, resulting in a subsequence of length 2. The same holds for the following 3 volumes: 4, 3 and 2. After that there is the volume 8, which can be added at the end of any of the last three 2-element subsequences formed, what results in a subsequence of length 3. Generally, for each element x, either:

- x is a single-element increasing sequence, or
- x can be appended at the end of the longest increasing subsequence ending at some y, provided that y precedes x in the given sequence and $y < x$.

Basing on this observation, we can calculate the following table of intermediate results:

Volume	11	1	10	4	3	2	8	7	12	6	9	5
Length of subsequence	1	1	2	2	2	2	3	3	4	3	4	3

Therefore, the longest increasing subsequence of the given sequence has 4 elements. Examples of such sequences are: $(1, 3, 7, 12)$ and $(1, 2, 6, 9)$. Hence, the smallest number of moved volumes equals $12 - 4 = 8$.

6.3. *Methodological Comments*

After reducing the main problem to the problem of finding the longest increasing subsequence, we deal with dynamic programming again – instead of one problem of the longest increasing subsequence, we have to consider a whole table of such problems. However,

the dependencies between increasing subsequences ending in different elements allow us to calculate their lengths more easily.

Trivial implementation of the dynamic programming algorithm runs in $\Theta(n^2)$ time, where n is the number of volumes. Without dynamic programming, one would have to consider an exponential number of subsequences of the given sequence. Note that using any of the classical $\Theta(n \log n)$ time algorithms for the longest increasing subsequence problem would be impractical here.

7. Heavy Encyclopedia

7.1. *Problem*

Let us modify the previous problem. The encyclopedia volumes turned out to be so heavy, that you can hardly move them from one position to another, or shift the remaining volumes to make space. You have changed your mind and decided to order the volumes only by swapping adjacent volumes on the shelf. What is the minimal number of such moves needed to order the volumes from 1 to 12, again left to right?

Hint: First perform such a sequence of moves, after which the first volume is located at the leftmost position on the shelf. Reduce the problem eliminating the first volume from further considerations, and repeat this procedure for the volumes $2, \ldots, 12$.

7.2. *Solution*

First, let us check, whether the hint is correct, i.e. generates the minimal number of moves. Let us consider some optimal solution and begin by focusing on volume 1. Let i be its initial position. The hint suggests that we should first move it to the left $i - 1$ times. If in the considered solution volume 1 is moved only to the left, then clearly there are $i - 1$ such moves. Moreover, without changing the total number of moves, we can make them first. This way we obtain an optimal solution that starts as suggested in the hint.

What if volume 1 is not only moved to the left? Let us assume that there is a volume x, which is swapped with volume 1, jumping over it to the left, that is: $\ldots, 1, x, \ldots \rightarrow \ldots, x, 1, \ldots$ In such a case, they must be swapped again at some point. Observe that we can save two moves, by "pushing" volume 1 in front of x and not swapping them at all. Hence, such a situation cannot occur in the optimal solution.

Now, assume that we first move volume 1 to the very left. Clearly, if it is swapped with any other volume later on, such a solution cannot be optimal. Hence, we can reduce the problem to ordering a smaller number of volumes. So, the hint is really sound.

Let us simulate the ordering described in the hint. The following table shows the sequence of volumes, after putting consecutive volumes in place:

Step	Moves	Volumes
1	1	11, 1, 10, 4, 3, 2, 8, 7, 12, 6, 9, 5
2	4	1, 11, 10, 4, 3, 2, 8, 7, 12, 6, 9, 5
3	3	1, 2, 11, 10, 4, 3, 8, 7, 12, 6, 9, 5
4	2	1, 2, 3, 11, 10, 4, 8, 7, 12, 6, 9, 5
5	7	1, 2, 3, 4, 11, 10, 8, 7, 12, 6, 9, 5
6	5	1, 2, 3, 4, 5, 11, 10, 8, 7, 12, 6, 9
7	3	1, 2, 3, 4, 5, 6, 11, 10, 8, 7, 12, 9
8	2	1, 2, 3, 4, 5, 6, 7, 11, 10, 8, 12, 9
9	3	1, 2, 3, 4, 5, 6, 7, 8, 11, 10, 12, 9
10	1	1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 10, 12
11	0	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
12	0	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
13		1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

The total number of moves is 31.

This result can also be obtained without simulating intermediate sequences of volumes. Note that the number of moves needed to position volume x equals the number of such volumes y, that $y > x$ and y is to the left of x in the original problem. Simply, these are the volumes that have to be swapped with volume x at some point, and after this, volume x is in its final position. Thus, the numbers of moves can be calculated just by analysing the original sequence of volumes.

Volumes	11	1	10	4	3	2	8	7	12	6	9	5
Moves	0	1	1	2	3	4	2	3	0	5	3	7

7.3. *Methodological Comments*

This problem illustrates two programming techniques: greedy programming and "divide and conquer" rule. The greedy algorithm sorts the volumes by placing consecutive volumes in place. The "divide and conquer" rule, after putting the first volume in place, allows us to reduce the problem to a smaller one.

When designing a greedy algorithm, usually, we have to prove two properties:

- the greedy property – that is, that there is an optimal solution starting as suggested in the hint,
- sub-problem property – that is, that the greedy step and an optimal solution of the smaller problem give the optimal solution of the original problem.

The greedy property is proved in the first two paragraphs of the previous section. The sub-problem property is rather trivial here, since we optimize the total number of moves.

It is worth noting that the number of moves is, in fact, the number of inversions in the given sequence. Clearly, for any two volumes x and y, such that $x < y$ and volume x is to the right of volume y, these volumes have to swapped at some moment in time. Hence, the number of moves is at least the number of inversions. Moreover, the final sequence, in which all the volumes are in place, contains no inversions, and every move

of the described solution reduces the number of inversions by one. Hence, the number of moves equals the number of inversions. As a consequence, every algorithm that reduces the number of inversions within each move, produces an optimal solution.

The time complexity of each of the presented solutions (the one based on simulation and the one analyzing inversions) is proportional to the number of required moves, that is $\Theta(n^2)$. On the other hand, calculating the total number of inversions can be done in $\Theta(n \log n)$ time, using appropriate data structures or algorithms – however, such a complicated solution would be impractical here.

8. Anti-Binary Sets

8.1. *Problem*

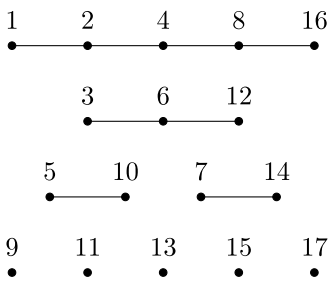
Let $Z = \{1, 2, \ldots, 17\}$. Let us consider such subsets $A \subseteq Z$, that for every $x \in A$ we have $2x \notin A$. Such subsets of Z are called *anti-binary*.

What is the largest anti-binary subset of Z? How many anti-binary subsets of Z are there?

Hint: Make a graph with vertices labeled from 1 to 17, and edges connecting vertices m and $2m$ (for $m = 1, 2, \ldots, 8$).

8.2. *Solution*

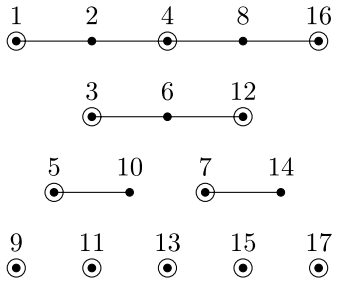
Let us construct the graph described in the hint:



It consists of a number of "paths": one containing 5 vertices, one containing 3 vertices, 2 containing 2 vertices each, and 5 single vertices. The definition of the anti-binary set means that such a set cannot contain elements connected by an edge. We can consider each path separately, since they are not connected with each other. First, we can select all 5 single vertices. From each path of size 2 we can select just one vertex. From the path of size 3 we can select 2 vertices – its ends. Finally, we can select 3 vertices from the path of size 5 – its ends plus the middle vertex. So, the maximum number of vertices that can form an anti-binary set is 12. An example maximum anti-binary set:

$$A = \{1, 3, 4, 5, 7, 9, 11, 12, 13, 15, 16, 17\}$$

is shown in the following figure:



Now, let us consider the number of anti-binary sets. This number also can be deduced from the structure of the constructed graph. Denote by A_n the number of anti-binary subsets of a path of size n. Then, the number of all anti-binary subsets of Z equals:

$$A_1^5 \cdot A_2^2 \cdot A_3 \cdot A_5.$$

One can easily check that $A_1 = 2$ and $A_2 = 3$. For longer paths it gets more and more complicated. Let us split all the anti-binary sets of a path of size n into two groups: containing the last vertex, and not containing it. The number of latter ones is simply A_{n-1}. If an anti-binary set contains the last vertex, then it does not contain its predecessor. Hence, the number of such sets equals A_{n-2}. From this, we obtain:

$$A_n = A_{n-1} + A_{n-2}.$$

Does it look familiar? Sure! It is a definition of the Fibonacci numbers! More precisely, $A_n = \mathit{Fib}_{n+2}$. In particular, $A_3 = 5$, $A_4 = 8$ and $A_5 = 13$. Hence, the total number of anti-binary subsets of Z equals:

$$2^5 \cdot 3^2 \cdot 5 \cdot 13 = 18\,720.$$

8.3. *Methodological Comments*

The first important step of the solution is to view the problem as a graph problem – we are looking for a maximum size independent set in the constructed graph. In general, this problem is NP-hard. However, since we consider graphs consisting of separate paths, it is much easier. Then the problem can be reduced even further, to graphs consisting of single paths. However, still the number of anti-binary sets is exponential, and the observation to use Fibonacci numbers is needed to compute it in linear time.

The solution contains elements of greedy programming, when constructing the maximum size anti-binary subset. On the other hand, in counting anti-binary subsets and computing Fibonacci numbers, there are elements of dynamic programming.

9. Conclusions

In this paper we have discussed several problems of algorithmic nature that we claim can be used to popularize algorithmic thinking among pupils not familiar with programming. Let us note that each of these problems is a small instance of a regular programming task, accompanied with a particular solution of the task that we would like to force. One could ask what distinguishes the tasks we have chosen in this paper that makes them good no-programming problems. Let us state a few conditions that we consider important.

The desired solution should be the one of the solutions to the problem that minimizes the total time of inventing it and "execution" by hand. In particular, both the time complexity and the constant factor are important. Moreover, the solution should technically be simple, contain just a few different patterns of steps to be performed. For all this to be possible, one should carefully examine possible simple solutions of the problem with worse time complexities and heuristics that could work well for the chosen instance of the problem.

Additionally, it is better to exclude problems that require the knowledge of classical algorithms and advanced techniques. This is for the problems to be solvable by pupils without large algorithmic background and for the problems to be interesting to pupils that have some knowledge of algorithms. Finally, the best situation is when there exists a very simple but slow solution for the problem that should be obvious for anyone solving the problem – this makes the task more understandable.

References

Beaver Competition. `www.bebras.org`.

Dagienė, V. (2006). Information technology contests – Introduction to computer science in an attractive way. *Informatics in Education*, 5(1), 37–46.

Dagienė, V. (2010). Sustaining informatics education by contests. In: *4th International Conference on Informatics in Secondary Schools – Evolution and Perspectives, ISSEP*, LNCS 5941, 1–12.

Dagienė, V., Futschek, G. (2008). Bebras international contest on informatics and computer literacy: Criteria for good tasks. In: *3rd International Conference on Informatics in Secondary Schools – Evolution and Perspectives, ISSEP*, LNCS 5090, 19–30.

Delta. `www.mimuw.edu.pl/delta`.

Diks, K., Kubica, M., Radoszewski, J., Stencel, K. (2008). A proposal for a task preparation process. *Olympiads in Informatics*, 2, 64–74.

Diks, K., Kubica, M., Stencel, K. (2007). Polish olympiads in informatics – 14 years of experience. *Olympiads in Informatics*, 1, 50–56.

Forišek, M. (2006). On the suitability of programming tasks for automated evaluation. *Informatics in Education*, 5(1), 63–76.

Guzicki, W. (1997). Uni-color triangles (Trójkąty jednobarwne). In: *IV Polish Olympiad in Informatics 1996/1997*, 119–120 (in Polish).

Opmanis, M. (2009). Team competition in mathematics and informatics "Ugāle" – finding new task types. *Olympiads in Informatics*, 3, 80–100.

Project Euler. `http://projecteuler.net`.

Radoszewski, J. (2009). Non-informatics tasks (Zadanka (nie)informatyczne). *Delta*, 8, 1, 12–14 (in Polish).

Verhoeff, T., Horváth, G., Diks, K., Cormack, G. (2006). A proposal for an IOI Syllabus. *Teaching Mathematics and Computer Science*, 4(1), 193–216.

M. Kubica (1971), PhD in computer science, assistant professor at Institute of Informatics, Faculty of Mathematics, Informatics and Mechanics, University of Warsaw, scientific secretary of Polish Olympiad in Informatics, IOI-ISC member and former chairman of Scientific Committees of IOI'2005 in Nowy Sącz, CEOI'2004 in Rzeszów, BOI'2001 in Sopot, and BOI'2008 in Gdynia, Poland. His research interests focus on combinatorial and text algorithms.

J. Radoszewski (1984), PhD student at Faculty of Mathematics, Informatics and Mechanics of University of Warsaw, chairman of the jury of Polish Olympiad in Informatics, former member of Host Scientific Committees of IOI'2005 in Nowy Sącz, CEOI'2004 in Rzeszów, and BOI'2008 in Gdynia, Poland. His research interests focus on text algorithms and combinatorics.

Olympiads in Informatics, 2010, Vol. 4, 67–75

© 2010 *Institute of Mathematics and Informatics, Vilnius*

Indonesian Olympiad in Informatics

Ilham W. KURNIA
Computer Science Faculty, University of Kaiserslautern, TU Kaiserslautern
Fachbereich Informatik, Gebäude 34, Postfach, 30 49, D-67653 Kaiserslautern
e-mail: ilham@cs.uni-kl.de

Brian MARSHAL
School of Computer Engineering, Nanyang Technological University
Nanyang Avenue, Block N4, Singapore 639798
e-mail: brian.marshal@pmail.ntu.edu.sg

Abstract. This article describes the experience of Indonesian Olympiad in Informatics (abbreviated as OKI in Indonesian) in introducing informatics to young generation in Indonesia through a form of competition. Organizing an informatics olympiad in a large developing country requires dealing with a myriad of challenges, such as unbalanced distribution of infrastructure development and poor human resource. In Indonesia, these challenges are dealt by staging a multi-tiered selection, concentrating on developing online training system and creating a strong cooperation with top universities. We analyze briefly how these efforts impact the spread of participation of OKI.

Key words: informatics, olympiad, training.

1. Introduction

Indonesian Olympiad in Informatics (abbreviated as OKI in Indonesian) is an informatics contest started in 1995 as an initiative from individuals such as Joko Saputro. The main goal of OKI is to introduce young generation in Indonesia to informatics through a form of competition, as the formal curriculum of pre-university education in Indonesia does not include any informatics education. In addition, OKI organizers coordinate the selection and training process of students to take part in IOI.

In the past 15 years, the number of participants of OKI itself has grown from 1 in 1995 to 1495[1] in 2009 with the primary sponsorship of the Ministry of National Education. Indonesia has participated in IOI every year since 1995 (with exception of 2003 due to visa problems) has collected in 2 golds, 11 silvers and 16 bronzes.

Despite the increase in the participant count, it is still relatively difficult to train the participants to reach their maximum potential. In this paper we look into factors that contribute to this difficulty, in particular with regards to poor infrastructure and human resource conditions, and what ways we have taken to resolve it.

[1] 1495 is the number of participants at the provincial selection stage in 2009. The number of participants at municipal selection stage is difficult to estimate since not every municipal keeps track of this properly, and some municipals even have extra selection stages.

This paper is organized as follows. Section 2 documents the challenges that are faced in organizing OKI. Section 3 discusses how they are tackled. In Section 4 we give a brief evaluation of OKI in terms of the participants. We close with our future plan.

2. Challenges

OKI continually faces multifaceted challenges and evolves accordingly. Here we focus on two main dimensions: infrastructure condition of Indonesia and human resource.

2.1. *Infrastructure Condition*

Indonesia is an archipelago of more than 17000 islands, with some 900 islands are inhabited, divided into 33 provinces[2]. It extends more than 5000 kilometers from east to west and more than 1500 kilometers from south to north, such that a hypothetical direct flight from Banda Aceh (west end) to Jayapura (east end) would take around 6 hours. 60% of 213 million citizens live in the Java island (Statistics Indonesia, 2005), where the capital of Indonesia, Jakarta, is, and most infrastructure development is done in this island.

Since 2003, a cooperation of three ministries in Indonesia execute the One School One Computer Laboratory program to "provide computer laboratories in all schools in Indonesia" (Salahuddin, 2005). However, in 2007 from 200000 schools, only a quarter have an unspecified kind of internet access, and in 2009 the ratio of computers to students in public school is 1:3200 (Business Monitor International, 2009). One participant said that he had to walk more than 50 kilometers to get computer access.

From these factors, we find that it is quite costly to gather so many potential students to have proper training. One alternative is to use the internet as a medium for knowledge transfer, but other means are also needed to reach more students.

2.2. *Human Resource*

There are three main factors of the human resource dimension that affect the organization of OKI, namely the secondary school curriculum, the tertiary education related to informatics, and the number of active organizers of OKI.

The official secondary school curriculum in Indonesia does not include informatics, as mentioned in Section 1. Consequently, there is a lack of secondary school teachers having the skills to teach informatics. At the few schools which offer informatics as an extracurricular subject, this subject is taught with an inclination towards using applications, such as word and image processors. The core informatics or even programming teaching is lacking, except in very select few, mostly at schools which have prior strong exposure to OKI. This means that developing the skills of potential students needs to be

[2] A general map of Indonesia with names of the biggest islands is available at `http://en.wikipedia.org/wiki/File:Indonesia_map.png`. A non-annotated map can also be seen in Fig. 2.

done in a more direct way. However, to be more effective in the long run, educating the teachers is a priority as the dependency of teachers by the students at school in Indonesia is relatively high.

Just as the infrastructure development centers in Java, skilled human resource is concentrated also in Java. Only 145 institutes in all other islands, compared to 256 institutes in Java, provide university-level informatics education (Directorate of Higher Education, 2008), and the quality differs significantly compared to the top universities in Java.

One factor that has a significant impact on the operational of OKI is the active organizers. Currently there are less than 10 active individuals in the main OKI organization which consist of university lecturers and OKI alumni. Moreover, they are involved in OKI as part-time (voluntary) organizers, which causes the implementation of many ideas to be brought up slowly. While the number of OKI alumni increases each year, this does not immediately translate in an increase of available human resource on site. A significant portion of the top alumni went to universities abroad (mostly Singapore), and the others who are absorbed in Indonesian universities pursue other interests.

3. Solutions (So Far)

To overcome the aforementioned challenges, we describe several aspects of how OKI is organized. First we describe the structure of the competition itself, followed by the online training aspect and the language used to introduce programming to newcomers. We then explain how the cooperation with top universities is built through the formation of OKI Bureaus. We also sketch other aspects of the organization of OKI.

3.1. *OKI Structure*

OKI is a one-year multi-tiered competition whose stages of the olympiad can be seen in Fig. 1. Each stage prunes down the number of participants to provide more intensive attention to each remaining participant. Since an edition of OKI lasts across two academic years[3], participants who are already in the last grade of schooling are not allowed to enter the pre-training camp selections.

The multi-tiered structure serves a couple of purposes. One purpose is to encourage more participation which increases the exposure of informatics in secondary school. Being a representative of their school/municipal/province may give the needed motivation to participants from less developed areas. Another purpose is to lessen the influence of luck in the selection process, especially towards the IOI team formation.

In the following subsections, we detail the two main parts of the OKI structure: the pre-training camp competition and the training camps.

3.1.1. *Pre-Training Camp Competition*

The first three stages of OKI focus on identifying participants who have the potential to be skillful in informatics. We try to assess this potential by employing two types of

[3]The academic year in Indonesia starts from July and ends in June.

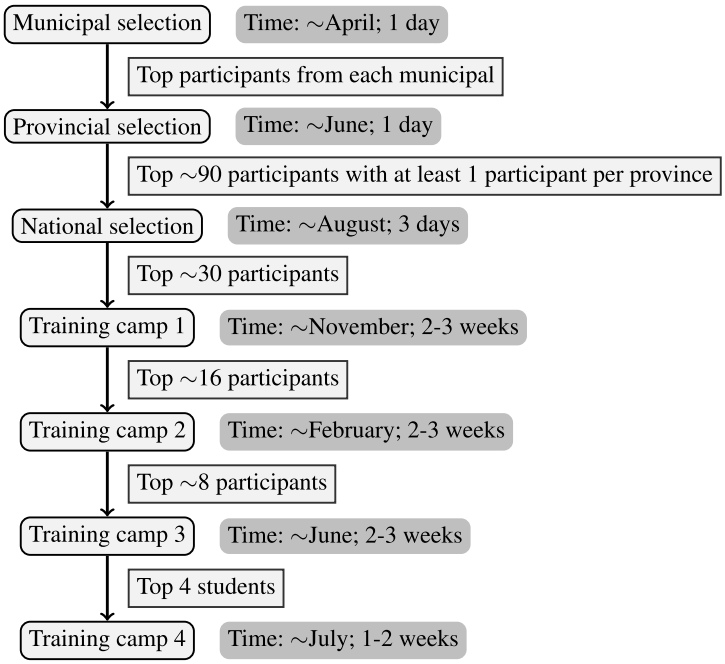


Fig. 1. Stages of OKI.

exams: analytical and algorithmic exams. The analytical exam is similar to the logic category exam used in the Brazilian Olympiad (Anido and Menderico, 2007), with extra arithmetic puzzles. The algorithmic exam consists of fundamental questions of programming techniques, such as control flow, variable assignment, logical operators, loop and recursion, using Pseudopascal (see Section 3.3). Although these two exam types persist, the exam format itself is still evolving.

What differentiates a stage from the other is the difficulty of the exam, the variation of questions and the proportion of aggregate marks between the two exam types. In the municipal and provincial stages, the analytical exam carries the same weight as the algorithmic exam, and both pen-and-paper exams consist of multiple choice questions. The programming exam at provincial stage is harder than at municipal stage.

Unlike all other stages, in the provincial stage, we take at least 1 participant per province to the national stage regardless of his or her result. This main objective of this decision is to give opportunity to all provinces to present their best participants in the national stage.

In the national stage, the analytical and algorithmic exams still exist, but now they have less weight. More emphasis is given to solving programming tasks. To see how far the participants are able to use their analytical skills to solve problems and program, the programming exam consists of a handful of simple implementation tasks (e.g., printing out the triangle number sequence and dealing cards) and harder tasks involving recursion, sorting, and searching.

3.1.2. *Training Camp*

After the national stage, we have a series of training camps. The training camps, which are round-robinly held at OKI Bureaus, are geared towards preparing the students for IOI. In the first two training camps, the participants are given lectures by university lecturers and OKI alumni consisting of material from the IOI syllabus (Verhoeff *et al.*, 2008). After each lecture, participants are given several basic programming tasks related to the lecture. Participants of the third and fourth training camps are drilled with typical IOI problems. The third training camp also serves to finish any parts of the syllabus that are not covered yet in the previous training camps. To rank the participants, scores from the lecture exercises and IOI-like simulations are aggregated with more weight given to the simulations.

Training camps give us opportunities to monitor closely the progress of each participants, especially at later stages, and tailor the direction of the camp to focus on their weak points and build their self-confidence. From our experience the participants get a boost of motivation when they correctly solve a problem. Therefore, we choose problems such that on each day, each participant should be able to solve a problem, while at the same time serving up at least one difficult problem to keep them thinking.

The small number of participants also allow us to experiment with different activities, such as programming on paper, emulating code-breaking session (Burton, 2008), and individual new problem composition and cross-solving them. While these activities spark the enthusiasm of the participants, the most important part of the training camp remains the discussion sessions. We find it crucial for an effective discussion session the presence of a moderator who knows the problems and ideas of the solutions, or at least directions to find the solutions. Thus, the ideal moderator is usually the problem setter.

3.2. *Online Training*

For participants who want to prepare themselves for OKI, we provide online training, which at the moment are opened only before each competition stage to allow intensive supervision. Pre-training camp stage up to the second training camp stage online training centers around topics which are examined on that stage. For the advanced stages, the online training is focused on building the problem solving and coding experience of the participants. To enhance the competitive spirit, the training material is given in levels akin to USACO Training (Kolstad and Piele, 2007), where participants may only move up a level after they finish all the work, and participants can view real-time how the others are faring.

Online training brings a number of benefits. First of all, it allows participants to improve themselves independently and systematically, even without the presence of a teacher. Secondly, the supervision in the form of comments on submitted works, hints, and clarification request replies allows the training to go more dynamically. This, in turn, allows us to recruit OKI alumni who are more interested to contribute this way. This also causes the student-teacher barrier to be much less due to the insignificant age difference.

Being an online training, the participants (and also the supervisors) have the flexibility to administer their own training session between a fairly large time period. This flexibil-

ity gives the participants time to actively participate in school life, which would not be possible if they were to study in a weeks-length training camp. However, as this online training is quite new, we need to optimize its usage and enrich the material.

3.3. *Pseudopascal*

Pascal is a language designed with pedagogical aspects in mind (ISO 7185). This advantage is utilized in OKI as the base language for the algorithmic test. Using pure Pascal language, in the past, have discouraged several potential participants who already know other programming language to take part. Therefore, in OKI we use a subset of Pascal called Pseudopascal (Setiawan, 2006) which includes only the following features:

- standard data types, except sets, pointers and (text) files,
- all Pascal control flow commands, except `goto`,
- all predefined statements and expression constructs remain as in Standard Pascal, except `with`,
- procedures and functions, without directives (`forward`, i.e., without mutual recursion),
- and predefined procedures related to reading from/writing to standard I/O.

Additionally, we can embed natural language into the program to allow some flexibility when describing an algorithm or a task.

In addition to simplifying the language, the participants are still able to program in Pseudopascal (minus the pseudocode) and have it compiled using a Pascal compiler, such as Free Pascal used in IOI. Thus, it integrates well with the online training application, and also previously developed learning material can still be used.

3.4. *OKI Bureaus*

In attempt to bring more manpower on board and spread the load of organizing OKI, we established OKI Bureaus. The bureaus function not only as a possible place to hold training camps and centers of information distribution, but also as regional contact point for arranging local training. For the universities, the training camps are good opportunities to attract potential students by showing directly what facilities they have. In return, they provide the participants with a head start in making their choice for tertiary education.

There are at the moment 5 OKI-bureaus as a result from our cooperation with University of Indonesia, Bandung Institute of Technology, Bogor Agricultural University, Gadjah Mada University, and Sepuluh November Technology University.

3.5. *Other Steps*

Starting from 2002, the Ministry of National Education organizes an annual national science olympiad (OSN in Indonesian) which centralizes the national selection stage of all different science olympiads. We take advantage of OSN to distribute offline materials, such as OKI live CD with grading system and eBooks, for the participants and their

teachers (supply permitting, participants of other olympiads) to bring home and further distribute especially to future participants.

We also hold teacher conferences during OSN to gain feedback about the condition at their places and inform them more about OKI. Additionally, OSN also ranks each province according to the performance of its participants (in terms of medals). This motivates provincial government to provide funds to train potential students. When possible, these seminars are also held at provincial selection when a member of OKI organizers is invited to oversee the selection process. These opportunities, however, are more sporadic than the annual OSN.

4. Results

Table 1 shows that participants from Java and Bali dominate training camps. This domination is being challenged by other regions, with increasing number of participants from Sumatera who pass the national selection. In the past 4 years, participants from Borneo and Sulawesi finally managed to break through. We attribute this growth to the decision of taking at least 1 participant per province for the national training and the online training which becomes more robust in the last few years.

Although the proportion of non-Java/Bali training camp participants shows an increasing trend, it still proves to be quite difficult for those participants to become an IOI team member. As shown in Fig. 2, except 1 participant from North Sumatera, so far only participants from Java and Bali manage to enter the IOI team. The reason for this is the stronger knowledge base and more local programming contests available in Java and Bali.

One pleasing tendency from the data we have is that participants who manage to enter the training camp stage usually bring a positive impact to their schools in terms of the success rate of getting more participants into the training camp. At the very least, participants from the same area become more competitive at national stage. We hope to see this snowball effect continue and create more areas where this effect can begin, especially in the eastern part of Indonesia.

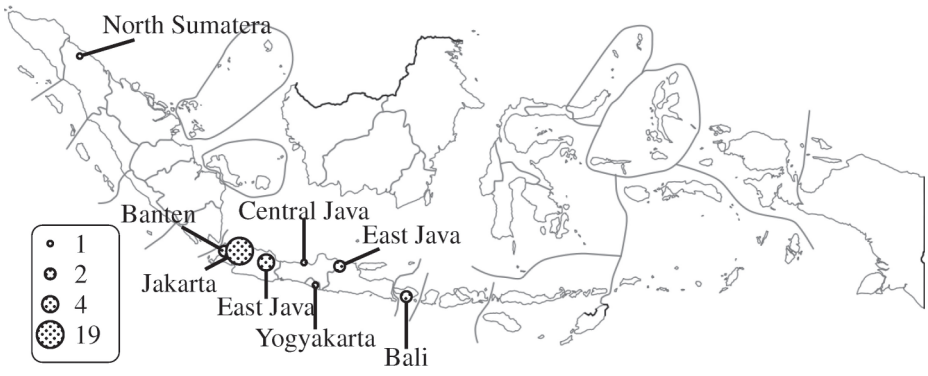


Fig. 2. 2002–2009 IOI team member count aggregation map.

Table 1
Participant province origin at the first training camp leading to IOI 2002–2009

Province	2002	2003	2004	2005	2006	2007	2008	2009	Total
Java									
Jakarta	10	9	4	5	8	8	6	7	57
Central Java	2	3	9	6	3	5	4	3	35
West Java	4	7	4	6	4	3	1	2	32
East Java	2	6	2	4	4	1	1	3	23
Yogyakarta		1	1	4	2	3	3	2	16
Banten		2	1	1	1	1	3		9
Total for Java	18	28	21	26	22	21	18	17	172
Bali	1	3	3	2	2	1	1	1	14
Sumatera									
Jambi			2	2	4	3	2	2	15
Riau	1	1					1	1	4
South Sumatera		1				1		3	5
West Sumatera								1	1
North Sumatera				5	6	1	2	2	16
Total for Sumatera	1	2	2	7	10	4	5	8	41
Borneo									
West Borneo					3	2		2	7
Central Borneo					1				1
Total for Borneo					4	2		2	8
West Nusa Tenggara	1	1	1	1					4
Sulawesi									
North Sulawesi					1				1
South Sulawesi							1		1
Gorontalo								1	1
Total for Sulawesi					1		1	1	3

5. Conclusion and Future Work

We presented the challenges of organizing an informatics olympiad in Indonesia and the implemented solutions which deal with these challenges. While the measure is not there yet to judge the progress so far with respect to the goal of OKI, we see that these solutions produce some improved results in terms of an increase in participants quality from areas without prior history of success in OKI.

Clearly, there is a room for improvement even with limited manpower. In the short-term future, we are concentrating on enriching the online training material, offering online contests and writing more informatics books for both the students and the teachers. We also start to keep track of the alumni to see the impact our alumni have.

Acknowledgment

We would like to acknowledge Suryana Setiawan, the chair of OKI and Indonesian IOI team leader, for his continuous and relentless effort to drive OKI from 1996. Without his involvement, both in the technical part, by making an online judge himself, and organizational part, OKI would not have developed this far. We also would like to thank Ardian K. Poernomo for reviewing previous drafts of this paper.

References

Anido, R.O., Menderico, R.M. (2007). Brazilian Olympiad in Informatics. *Olympiads in Informatics*, 1, 5–14.

BPS-Statistics Indonesia (2005). Population Survey (in Indonesian). `http://www.datastatistik-indonesia.com/`, last accessed: 15 January 2010.

Burton, B. (2008). Breaking the routine: Events to complement informatics olympiad training. *Olympiads in Informatics*, 2, 5–15.

Business Monitor International (2009). Indonesia Information Technology Report Q4 2009.

Directorate of Higher Education (2008). *Higher Education Institution Profile (in Indonesian)*. `http://evaluasi.or.id`, last accessed: 11 February 2010.

International Organization for Standardization (1990). American National Standard for information technology: Programming language PASCAL: ANSI/ISO 7185-1990: Revision and redesignation of ANSI/IEEE 770.X3.97-1983 (R1990).

Kolstad, R., Piele, D. (2007). USA Computing Olympiad (USACO). *Olympiads in Informatics*, 1, 105–111.

Salahuddin, M.R. (2005). *One School One Laboratory Program to Address Digital Divide in Indonesia*. `http://bit.ly/coD1RH`, last accessed: 11 February 2010.

Setiawan, S. (2006). *Pseudopascal (in Indonesian)*. `http://www.toki.or.id/toki2006/pseudopascal.pdf`, last accessed: 11 February 2010.

Verhoeff, T., Horváth, G., Diks, K., Cormack, G., Forišek, M. (2008). *The International Olympiad in Informatics Syllabus*. `http://people.ksp.sk/ misof//ioi-syllabus/`, last accessed: 11 February 2010.

I.W. Kurnia is an OKI alumnus and a coach since 2002. He participated in OKI 1999–2002 and IOI 2002. During the years at University of Indonesia studying for his bachelor, he gained hands-on experience in organizing national selections and training camps. He is a Ph.D. student at University of Kaiserslautern working on object-oriented program verification.

B. Marshal is an OKI alumnus, who participated in OKI 2005–2007 and IOI 2007. Since his high school graduation in 2007, he has participated in OKI activities as an organizer and coach whose main contribution is in organizing OKI alumni resources. Apart from his activities in OKI, he is currently a penultimate year computer science student at Nanyang Technological University.

Olympiads in Informatics, 2010, Vol. 4, 76–86

© 2010 *Institute of Mathematics and Informatics, Vilnius*

Testing of Programs with Random Generated Test Cases

Krassimir MANEV
Department of Mathematics and Informatics, Sofia University
J. Bourchier blvd. 5, 1164 Sofia, Bulgaria
Institute of Mathematics and Informatics, Bulgarian Academy of Sciences
G. Bontchev str. 8, 1113 Sofia, Bulgaria
e-mail: manev@fmi.uni-sofia.bg

Biserka YOVCHEVA, Milko YANKOV, Peter PETROV
Department of Mathematics and Informatics, Shumen University
Universitetska str. 115, 9712 Shumen, Bulgaria
e-mail: bissy_y@yahoo.com, m.yankov@f5bg.net, peshoto_bg@yahoo.com

Abstract. Testing of computer programs is an essential part of the evaluation process of a programming contest. It is a mix of functional and non functional testing and a specific case of the "black box" testing well known from the domain of Software engineering. The paper discusses one of the possible forms of creating test cases for a program that implement an unknown algorithm – the random generation of test cases – and the problems that could arise when random generated test cases are used for evaluation of programs submitted by participants in programming contests. Rooted trees were chosen for the start of the research because of their simplicity. More deep problems and more interesting results could be expected for general graphs and other combinatorial objects.

Key words: program testing, functional and non functional testing, "black box" testing, graphs, rooted trees, random generation, height, width and branching statistics of rooted trees

1. Introduction

In the late 70s' of the past century the intuitive *debugging* of computer programs was replaced by more systematic, mathematically reasoned and well planned, *software testing* (Mayers, 1979). With the development of the software industry and the growing of the number of proposed on the market software products testing of software became one of the essential stages of the software products' life cycle. Nowadays significant amount of the resources of the software developers are dedicated to testing of products under development. Something more, as a result of the efforts of many programmers and researchers *Software testing* became a scientific domain – sub-domain of *Software engineering* (Kaner *et al.*, 1999) – and corresponding discipline was included in the universities' curricula. One of the approaches of Software testing is the *random generation* of test cases.

Evaluation of the programs, submitted by the contestants during programming contests, is a specific kind of software testing. Random generation of test cases is a part of evaluation of competitive programs too. It was assumed for long years that the authors of the tasks have the best knowledge of the formulated by themselves problems and that they are able to prepare the best test cases. Some examples of Bulgarian national programming contests, and even from international contests, suggest that the author sometime underestimate the careful preparation of test cases.

During an IOI (about 10 years ago) a task on a graph $G(V, E)$ with a weight function c: $V \rightarrow N$ on the vertices was proposed. Some other function f: $V \rightarrow N$ has to be calculated and a vertex v to be found such that $f(v) = \max f(v_i)$, $v_i \in V$. Being in shortage of time one of Bulgarian contestants submitted as a solution a program that prints the vertex w, such that $c(w) = \max c(v_i)$, $v_i \in V$, and obtained enormously big amount of points. We could guess that some of the test cases were random generated.

Some years ago were popular optimization tasks (reduced versions of which are NP-complete), with "relative" evaluation. The programs that give a result nearest to the searched optimum received 100% of the points and the other programs received an amount of points proportional to the distance between their result and the best one. We proposed such task in one of Bulgarian national contests. After the grading we decided to change the random generated test cases with other random generated and the results of the contestants changed dramatically.

Examples show that, even inevitable, the random generation of test cases is full of risks. Random generation of test cases is one of the first approaches used for so called *automated test data generation* (Bird and Munoz, 1983; Duran and Ntafos, 1984) which is still in use in software industry. The advantages and negatives of the approach are widely discussed from the point of view of Software engineering. Here we will discuss some negatives of the approach when it is used for evaluation of solutions of the participants in programming contests.

The discussion is based on the example of *rooted trees*. Tasks on graph structures are among the most used for competitions in programming (Manev, 2008) and generating of test cases for such tasks is not trivial. Trees and rooted trees are simplest kind of graph structures and it is natural to start discussion with them. Because rooted trees have such interesting for random generating characteristics as height, width and branching, which are not intrinsic for not rooted trees, we prefer to start with the former for our considerations.

In Section 2 some notions from Software testing theory are given and the objectives of testing competitive programs are stressed. The necessary definitions from the Theory of graph structures are given in Section 3 as well as an example of a task on a rooted tree and some negatives following from not correct random generated test cases. All experiments in this and following sections were made with the standard function rand() of GNU C/C++ programming environment. Section 4 is defining two experiments aimed to estimated the ability of rand() to generate random rooted trees. Some results of the experiments are summarized and commented in Section 5. Section 6 contains ideas for future research.

2. Testing of Programs – Some Notions and Objectives

In general, two major sub-domains are considered in the theory and the practice of software testing – *functional* and *non functional testing*. We will use the following notation and assumptions for functional testing of a program module.

Each program module M is designed and implemented in a way to satisfy some *functional specification* F: $D \rightarrow R$, where D is the set of possible inputs and R – a set containing possible results (outputs). The set D is called *domain* of F.

Because M is implementing F, it could be considered as a function M: $D \rightarrow R$ too. We have to stress that the function M usually is a *partial* (not total), i.e., there are elements in D for which M is not defined – starting on an input δ, M crashes or enter an infinite loop. The set $D_M \subseteq D$ of all elements $\delta \in D$ at which $M(\delta)$ is defined is called a *domain* of M. We will suppose that R is large enough and when M stops it gives always a result ρ from R.

The ideal goal (the super goal) of the *functional testing* is to prove the *functional correctness* of M *toward* F, i.e., for given F and M to check whether $D_M = D$, and if it is true, to check whether $M(\delta) = F(\delta)$, for each $\delta \in D$. It is obvious that achieving the super goal of the functional testing in general case is very difficult and **rather impossible**. First, because it is impossible to find the domain D_M in general case – Stop-problem is undecidable for usual computational models. And second, because the size of D is usually very large.

In practice we take a reasonably small $D_T \subseteq D$ called *test set* or *test data*. Elements of D_T, called *test cases*, have to be chosen in such way that if $M(\tau) = F(\tau)$ for each $\tau \in D_T$ we could **consider** M (more or less) **functional correct** toward F. It is obvious that with such testing we are not able to check at all first condition of the functional testing – $D_M = D_F$ – because of the possibility that M enters an infinite loop for some $\tau \in D_T$. For the purpose additional efforts are necessary, as mentioned below.

The other form of testing, called *non functional*, is also interesting. Different qualities of the program module could be a goal of such testing when a commercial programming product is tested – usability, reliability (when the module provide some service), etc. One quality of specific interest for evaluation of programming contests is the CPU time elapsed by the program. For checking this quality, together with the test set D_T, a *time limit* constant L is chosen. The execution of the program on the test case is stopped if the elapsed time goes over L and the program is considered not correct for the corresponding test case.

Testing with time limit has a double purpose. First, it compensate the principle impossibility to obtain $D_M(\tau)$ in case when M crashes or enters in an infinite loop working on τ. But more important, especially for programming contests evaluation, is the possibility to estimate in such way the CPU time of the algorithm used (for definition of the notion *time complexity* see, for example, Cormen *et al.*, 2001). That is why the test set has to contain test cases of different input size, including very large test cases and the time limit to be chosen in such way that the possible algorithms of different time complexity to be identified. Random generation is the natural way for construction of the large tests.

We will finish discussion of the notions of the domain with another classification of the testing approaches. Two different cases could be defined depending of whether the source code of the module is used in the testing process or not. If the source code could be used for generation of the test set then the testing is called *transparent box* testing. If the source code could not be used – the testing is called *black box* testing.

During the evaluation of programming contest the codes of the tested programs are available. But by different reasons the usage of transparent box testing is not possible. The main of these reasons is that with the methods of transparent box testing different (by the number and the content of the test cases) test sets will be generated for the different programs (depending of the programs themselves), which is not acceptable for evaluation of programming contests. That is why in programming contests **the black box testing was always used**.

3. Testing with Random Generated Rooted Trees

Here we will give only the necessary for the rest of the paper definitions, concerning rooted trees. For the other used notions see, for example (Manev, 2008). By the classic definition, the graph $T(V, E)$ is a *tree* if it is connected and has no cycles. For the purposes of algorithmics the notion *rooted tree* is more helpful. One of the possible inductive definitions of rooted tree is:

DEFINITION. (i) The graph $T(\{r\}, \varnothing)$ is a rooted tree. r is a *root* and a *leaf* of T; (ii) Let $T(V, E)$ be a rooted tree with root r and leaves $L = \{v_1, v_2, \ldots, v_k\}$, $v \in V$ and $w \notin V$; (iii) Then T' ($V' = V \cup \{w\}$, $E' = E \cup \{(v, w)\}$) is also a rooted tree. r is a root of T' and leaves of T' are $(L - \{v\}) \cup \{w\}$.

By the DEFINITION rooted trees are undirected graphs but an *implicit orientation* on the edges of the rooted tree exists. Following DEFINITION we will say that v is a *parent* of w and that w is a *child* of v. Obviously each rooted tree is a tree and each tree could be rebuild as rooted when we choose one of the vertices for a root.

For each vertex v of the tree $T(V, E)$, rooted in the vertex r, we define the *height* $h(v)$ of v as the length of the path from r to v and the *height* $h(T)$ of T, $h(T) = \max\{h(v)| v \in V\}$. For each vertex v of the rooted tree $T(V, E)$ we define the *branching* $b(v)$ of v as the number of children of v and the *branching* $b(T)$ of T, $b(T) = \max\{b(v)|v \in V\}$. Let we define the ith *level* of the rooted tree $T(V, E)$, $i = 0, 1, \ldots, n-1$ as $L_i = \{v|v \in V, h(v) = i\}$, were $n = |V|$. Then the *width* $w(T)$ of T is defined as $w(T) = \max_{i=0,1,\ldots,n-1}\{|L_i|\}$.

One of the simplest ways to represent the tree $T(V, E)$ with $V = \{1, 2, \ldots, n\}$, rooted in the vertex r, is the *list of parents* – a vector $(p_1, p_2, \ldots, p_n)$, where p_i is the parent of $i \neq r$ and $p_r = 0$. The list of parents could be implemented in an array p[] of integers and is very convenient when it is necessary to build a rooted tree.

Suppose that for a programming contest the following task is prepared:

Task. A rooted tree T with n vertices, labeled from 1 to n, is given with its list of parents, $3 \leqslant n \leqslant 1000$. Let the root of the tree be the vertex labeled with 1. Write a program to find a vertex of T with maximal height.

Following DEFINITION we built a very simple generator of random rooted trees without any additional conditions on the structure of the generated objects. The essential part of the source is given below:

```
...
int n, p[MAXN], i;
p[1] = 0; p[2] = 1; // vertex 1 is the root
for(i = 3; i <= n; i++)
   p[i] = rand() % (i - 1) + 1;
...
```

For evaluation of contestants' program we decide to make 100 tests – of $10, 20, \ldots,$ 1000 vertices, respectively and for each test to assign 1 point. What will happen if a contestant has submitted a program, which for each test case print n – the maximal label of a vertex? We generated 100000 random test sets with 100 test cases each. The average result of the contestant is 5.29 points which is too much for the invested efforts. But even worst is that among the 100000 random generated there is a test set for which the contestant will obtain 16 points.

The obvious defect of the used generator is that the vertex with label n is always a leaf of the tree and the chance to be a leaf with a maximal height is big enough. To eliminate such defect we could append at the end:

```
for(i = 2; i <= n; i++)
    p[i] = perm(p[i]);
```

where the function `perm()` calculate some random permutation of the numbers $2, 3, \ldots, n$. And then to eliminate the test cases in which the vertex n is a leaf with maximal height. In such way the trivial program, which print he value of n will not obtain points. For the experiments described below we are using this ameliorated version of our simple generator.

4. Experiments on the Randomness of Generated Rooted Trees

In the following our goal is to check the "randomness" of the obtained by our generator rooted trees. For the purpose let us first stress that the upper bound 1000 for the parameter n is too small. Quite reasonable rooted trees could have 100000 vertices because the input data for such rooted tree will be less than 700 KB and reading such input will not increase dramatically the time necessary for execution of the chosen algorithm. Something more, suppose that the author has in mind two different algorithms – a naïve one of time complexity $O(n \log n)$ and sophisticated one of time complexity $O(n)$. With a black box testing it will be difficult to identify which of the algorithms is used by the contestant

if the maximal number of the vertices is about 10000, for example. Especially in the case when the second algorithm has relatively big multiplicative constant. The identification of the used algorithm when the rooted tree is with 100000 vertices is trivial.

In order to obtain relatively *reliable test set*, i.e., a test set that is able to convince us in the functional correctness of the tested program we have to answer some important questions that arise on this stage:

- How many tests cases are necessary?
- Which will be the size of the rooted tree in each test case?
- Is it necessary to incorporate some specific structure in the generated rooted trees?

In Bulgarian national contest each task is evaluated with 100 points. That is why the practice is the number of test cases in the test set to be some divisor of 100 grater then or equal 10 – 10, 20, 25, 50 or 100 – and for each test case, which successfully passed the testing 10, 5, 4, 2 or 1 point to be assigned, respectively. It is obvious that such practice is not related at all with the specific task and the expected algorithms. Two major negatives trivially follow from such approach. First, it is quite possible that the chosen number of test cases is not enough even for an average confidence in the correctness of the tested algorithm and its implementation. And second, it is possible that all generated in such way test cases to be, in some sense, equivalent and to happen that the test set punish some kinds of mistakes and tolerate other kinds.

Having in mind the last reason some authors are limiting themselves to generating by hand only test cases with a small number of vertices in order to incorporate in them some structure corresponding to the particularities of the task. Such practice is also unacceptable because it could lead to receiving huge amount of points for trivial (as time complexity) and even wrong solutions. The only way to escape it is *the automatic generation of the "big" test cases*.

There are two ways of generating test case with a program. First of them is the random generation of many test cases with one program having as a single parameter the size of the generated object. The second is to create a specific program for each test case that is incorporating some structure in the generating object or to use many parameters in the generator, in order to escape equivalent test cases. The second possibility is obviously consuming much more of the author's time and is rarely used. Because of frequent using of random generation of test cases a question arises: *how reliable are the test cases generated randomly without incorporating any structure elements in the generated objects or tuning of various parameters*?

Below we are describing some experiments with small random generated rooted trees in order to observe some trends and to establish some directions for future research of the problems.

For generating of full set of rooted trees with n vertices, without excluding the isomorphic cases, a very simple program was built too. It generates all subsets of $n-1$ edges and eliminates each of the subsets that forms disconnected graph or graph with a cycle. The obtained trees are transformed to rooted trees choosing for a root the vertex 1.

Two kinds of experiments were organized in order to obtain some statistics about the randomly generated rooted trees. The main characteristics mentioned above – the height, the branching, and the width of the rooted trees – were observed.

The goal of the first experiment was to check the ability of the standard function rand() to build really random rooted trees. For the purpose, first, all $RT(n)$ rooted trees with at most 8 vertices were generated, without eliminating the isomorphic cases (we postpone eliminating of the isomorphic cases for the moment). For each generated rooted tree with n vertices the three characteristics were calculated and the corresponding distributions $(h_1, \ldots, h_{n-1})$, $(b_1, \ldots, b_{n-1})$ and $(w_1, \ldots, w_{n-1})$ were obtained, where h_i, b_i and w_i are the number of rooted trees of n vertices with height, branching and width equal to i, respectively.

Then the same number $RT(n)$ of random rooted trees with n vertices was generated and the corresponding distributions $(H_1, \ldots, H_{n-1})$, $(B_1, \ldots, B_{n-1})$ and $(W_1, \ldots, W_{n-1})$ were compared with the distributions $(h_1, \ldots, h_{n-1})$, $(b_1, \ldots, b_{n-1})$ and $(w_1, \ldots, w_{n-1})$. Our preliminary expectations were that the function rand(), dedicated to generate random numbers in a given linear interval, will not be able to generate really random rooted trees – objects that have nonlinear structure. The obtained results are given in the next section.

The goal of the second experiment was to check is it possible with a small number of random generated rooted trees to obtain statistics for the observed parameters which are close enough to the statistics of the full set.

For the second experiment only the rooted trees of 7 vertices was used. Random subsets of 20, 30, 40 and 50 such trees were generated. For each subset the relative distribution (in %) of the observed characteristics was calculated and compared with the corresponding, also relative, distribution for the full set. Our preliminary expectations were that small subsets of random generated rooted trees will not be able to "cover" statistically the full set. The obtained results of this experiment are also given in the next section.

5. Experimental Results and Comments

5.1. *Statistics of Random Generated Rooted Trees with up to 8 Vertices*

Because there is a single rooted tree (with a root in the vertex with label 1) when the number of the vertices n is 1 or 2, the smallest interesting case is $n = 3$. There are three different rooted trees in this case (two of them are really isomorphic but we are considering isomorphic trees with different labeling of the vertices as different). The set of random generated 3 rooted threes coincided with the set of all trees.

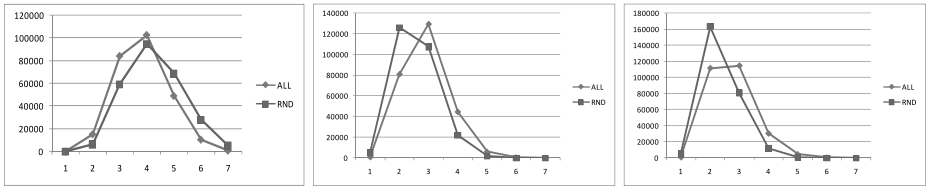
When $n \geqslant 4$, as it was expected, the distributions of the trees and random trees become different (see the results for $n = 8$ in Table 1 and on Fig. 1).

It is possible to classify the deviation of the random generated distribution from the real distribution (for all three observed characteristics) as "displacement of the peak". For

Table 1

Rooted trees with 8 vertices

k	Trees of height k		Trees of width k		Trees of branching k	
	ALL	RND	ALL	RND	ALL	RND
1	19	1	857	5040	857	5040
2	15166	6321	80976	126000	111394	163170
3	84067	59472	129403	107520	114421	80850
4	102563	94710	44581	21875	30459	12005
5	49016	68880	5897	1659	4583	1029
6	10456	27720	411	49	411	49
7	857	5040	19	1	19	1



a. Height b. Width c. Branching

Fig. 1. Rooted trees with 8 vertices.

the heights – the random generated values leftmost of the peak of the distribution curve are less then the real values and the random generated values rightmost of the peak are grate than corresponding real values. We will call such deviation *right displacement* of the peak. For width and branching the displacement is in the opposite direction – let us call it *left displacement*. Statistically, as big the height of the rooted tree is as small is its width (or branching), which explain the opposite directions of the displacements.

We have not an explication of the observed displacement but the trend is very clear. That is why we could suggest to incorporate minor correction in the generator when random tests are generated without any additional constrains on the structure of the rooted trees – to lift a bit the average height (or to decrease width/branching) of the generated objects.

5.2. *Statistics for Small Subsets of Random Generated Rooted Trees with 7 Vertices*

The results of the second experiment for the distribution by height and width are summarized in Table 2 and Table 3. In order to compare the obtained distributions by height they are transformed in percentages. As a criterion for *similarity* of two distributions was used the traditional χ^2 criterion – the sum of the squares of the differences between real percentage and those obtained by the random generation, divided to the real percentage (last element in each of the columns from 9 to 13).

Table 2

Random generated rooted trees with 7 vertices – height distributions

h	ALL	RND	R20	R30	R40	R50	%ALL	%RND	%R20	% R30	%R40	%R50
1	1	13	0	0	0	0	0.01%	0.08%	0.00%	0.00%	0.00%	0.00%
2	1056	1829	2	5	5	2	6.28%	10.88%	10.00%	16.67%	12.50%	4.00%
3	5550	6819	6	13	12	14	33.02%	40.57%	30.00%	43.33%	30.00%	28.00%
4	6240	6063	9	8	20	27	37.13%	36.07%	45.00%	26.67%	50.00%	54.00%
5	3240	1872	3	4	3	6	19.28%	11.14%	45.00%	13.33%	7.50%	12.00%
6	720	211	0	0	0	1	4.28%	1.26%	0.00%	0.00%	0.00%	2.00%
	16807	16807	20	30	40	50	0%	$\chi^2 = 12$	$\chi^2 = 43$	$\chi^2 = 29$	$\chi^2 = 22$	$\chi^2 = 13$

Table 3

Random generated rooted trees with 7 vertices – width distributions

W	ALL	RND	R20	R30	R40	R50	%ALL	%RND	%R20	% R30	%R40	%R50
1	720	211	0	0	0	1	4.28%	1.26%	0.00%	0.00%	0.00%	2.00%
2	9720	7539	9	14	23	33	57.83%	44.86%	45.00%	46.67%	57.50%	66.00%
3	5580	7233	10	13	12	12	33.20%	43.04%	50.00%	43.33%	30.00%	24.00%
4	750	1662	1	3	5	4	4.46%	9.89%	5.00%	10.00%	12.50%	8.00%
5	36	149	0	0	0	0	0.21%	0.89%	0.00%	0.00%	0.00%	0.00%
6	1	13	0	0	0	0	0.01%	0.08%	0.00%	0.00%	0.00%	0.00%
	16807	16807	20	30	40	50	0%	$\chi^2 = 18$	$\chi^2 = 16$	$\chi^2 = 17$	$\chi^2 = 19$	$\chi^2 = 8$

The experiment shows the expected amelioration of the similarity when the number of random generated objects is growing from 20 to 50 – values of 43, 29, 22 and 13. Applying the same χ^2 criterion on the generated in the previous experiment set of 16807 random trees of 7 vertices we could see that the significant increasing of number of generated trees do not lead to crucial amelioration of the distribution – the value $\chi^2 = 12$ in this case is practically the same as the value for 50 random generated trees.

As it is also expected the single tree with $h = 1$ had no real chance to be generated randomly. On the other side of the scale – a rooted tree with the maximal $h = 6$ was for the first time generated randomly only in the case of 50 generated trees. That means – if the task is such that the solution has to be tested on the marginal trees of height 1 and 6 then such tests have to be generated by hand.

From the other two experiments only the obtained distributions for the width of random generated subsets of 20, 30, 40 and 50 rooted trees is presented (see Table 3), because the results for distributions for branching, as it was mentioned above, are very similar.

Measuring the similarity with χ^2 criterion we could see that in the case of width distribution the sets with 20, 30 and 40 rooted trees approximate almost identically the real distribution (similarity of 16, 17 and 19, respectively) and the set of 50 random generated trees approximate real distribution much better – similarity 8. But the similarity

of the width distribution of generated for the first experiment set of 16807 random rooted trees happens to be 18 – not better than in the case of 20 and 30 random generated trees. So, our conclusion that generating of too much test cases is not necessary is confirmed by the width distribution too.

6. Conclusions and Ideas for Further Research

Generating of random test cases for evaluation the programs of the participants in programming contests has no alternative. But with not controlled random test generation we could obtain test sets that do not permit a fair evaluation process – with to many similar test cases or with missing of some important test cases. When a black box testing is performed it seems important that the test set contains test cases, which cover the whole variety of values of the intrinsic characteristics of the generated objects. Our experiments, even very simple, confirm this conjecture.

Discussion in this paper has some particularities that have to be stressed. First, we are considering all (i.e., labeled) rooted trees *without excluding isomorphic cases*. Much more reliable results could be obtained if we consider only one tree out of each set of isomorphic trees – i.e., not labeled rooted trees. For the purpose corresponding software could be created for eliminating isomorphic cases.

Second, our experiments are made on trees with very small number of vertices. As it was mentioned above, the interesting for evaluation of competitive programs are trees with hundred thousand of vertices. It is obvious that *the same experiments are impossible for such large trees* because of impossibility to generate all of them in real time and to collect the necessary statistics. For the purpose some combinatorial and statistical results for the distribution of the observed characteristics have to be attracted.

And third, *each generation* in the described experiments *was performed once*. From statistical point of view it will be better if each of these generations was repeated as many times as possible in order to eliminate the statistical mistake. The reason to do the former was to simulate the process of generation of real test cases – the author of random test cases usually performs the generation once. And we would like to see what will happen in such case.

The presented experiments and corresponding discussions was made on the example of rooted trees. But the problems will be the similar with *generation of any object* that could be used as a test case for a task proposed for programming contests – general graph structures, sequences of numbers, strings, set of geometric objects, etc.

Eliminating the mentioned above limitation of the presented experiments as well as investigation of many other aspects of the problem could be subject of a future research. Better knowledge of the statistical distribution of important characteristic of included in competitive tasks discrete objects will permit us to generate better test sets even when a random generation is used. It is worth to make the necessary efforts in this direction in order to obtain more adequate evaluation of contestants' program.

References

Bird, D., Munoz, C. (1983). Automatic generation of random self-checking test cases. *IBM System Journal*, 22(2), 229–245.

Cormen, T.H., Leiserson, C.E., Rivest, R.L. (2001). *Introduction to Algorithms*. MIT Press.

Duran, J., Ntafos, S. (1984). An evaluation of random testing. *IEEE Trans. on Software Engineering*, SE-10, 438–444.

Kaner, C., Falk, J., Nguyen, H.Q. (1999). *Testing Computer Software*, 2nd Ed. John Wiley and Sons, Inc.

Manev, K. (2008), Tasks in graphs. *Olympiads in Informatics*, 2, 90–104.

Myers, G.J. (1979). *The Art of Software Testing*. John Wiley and Sons.

K. Manev is a professor of discrete math and algorithms in Sofa University, Bulgaria, PhD in computer science. He was a member of NC for OI since 1982 and president of NC from 1998 to 2002. He was member of the organizing team of First and Second IOI, leader of Bulgarian team for IOI in 1998, 1999, 2000 and 2005. From 2000 to 2003 he was an elected member of IC of IOI. Since 2005 he represents the Host country of IOI'2009 in IC of IOI.

B. Yovcheva is assistant professor of informatics in Shumen University, PhD in education in informatics. She is creator of the private school A&B of Math and Informatics in Shumen and teacher of some of the most successfull Bulgarian participants in International and Balkan Olympiad in Informatics.

M. Yankov is an assistant professor of informatics in Shumen University. He is a coach of the students from Shumen University for competitions in programming.

P. Petrov is an young teacher in informatics in A&B School of Shumen. He is coach of the regional team of informatics for 15.5 years of Shumen. He is an author/co-author of a book on algorithms and algorithmic problems. Since 2009 he is an assistant professor of informatics in Shumen University and coach of the students from Shumen University for competitions in programming.

Olympiads in Informatics, 2010, Vol. 4, 87–94

© 2010 *Institute of Mathematics and Informatics, Vilnius*

Performance Analysis of Sandboxes for Reactive Tasks

Bruce MERRY
ARM Ltd
110 Fulbourn Road, Cambridge, CB1 9NJ, United Kingdom
e-mail: bmerry@gmail.com

Abstract. Security mechanisms for programming contests introduce some overhead into time measurements. When large numbers of system calls are made, as is common in reactive tasks with processes communicating over pipes, this may significantly distort timing results. We compared the performance and consistency of two sandboxes based on different security mechanisms. We found that in-kernel security has negligible effect on measured run-times, while ptrace-based security can add overhead of around 75%. We also found that ptrace-based security on a dual-core CPU adds far greater overhead as well as producing highly variable results unless CPU affinity is used.

Key words: sandbox, security, timing.

1. Introduction

In programming contests supporting native languages (such as C), it is common practice to execute the resulting code in some form of secured environment, or *sandbox*. This makes the contest environment robust against malicious or defective solutions that may attempt to interfere with the evaluation process. This is achieved by some form of monitoring of the calls made by the application to the operating system (so-called *system calls*).

The performance impact of such monitoring will of course depend on the number of system calls that are made. Batch tasks are tasks in which the solution inputs data from a file, processes it, and outputs results to another file. Since I/O libraries generally buffer data, this usually results in relatively few system calls, even if there are a large number of individual input and output values[1].

Reactive tasks are tasks where a solution will both send and receive data in an interleaved sequence. Typical reactive tasks are query processing systems (where the solution will receive information, then answer queries about the information, potentially interleaved with updates) and games (where the solution will output moves in the game and receive input about the state of the game). For reasons of security, this is commonly implemented as a communication between processes using a pipe; since each output must

[1]It should be noted, however, that a common mistake of C++ programmers is to end lines using `std::endl`, which also flushes the output and hence produces a system call.

be produced before the response can be received, each communication results in a system call. Thus, reactive tasks can require hundreds of thousands of system calls, making them particularly sensitive to the performance of the sandbox.

2. Experimental Design

2.1. *Sandboxes*

We have used two sandboxes. The first is based on the interception of system calls from userspace (using the `ptrace` system call in Linux), which is used in the USA Computer Olympiad (USACO; Kolstad,2009).

The second sandbox is based on in-kernel security checks and implemented as a Linux Security Module (Merry, 2009). This is the security module used in the South African Computer Olympiad (SACO), but updated to run with Linux 2.6.30. Note that this version of Linux no longer allows such security modules to be loaded at run time, so the "module" is in fact a kernel patch and is part of the binary kernel image.

2.2. *Tasks*

To keep the number of variables manageable, we considered only a single task: *Regions* from the International Olympiad in Informatics 2009 (Fan and Peng, 2009). We also only used test case number 32 (the last and largest). This task is a query processing task, and for this test case there are 200,000 queries. This yields slightly over 400,000 system calls (one to receive each query, one for each reply, and a small number during startup and shutdown) — this was confirmed using strace (`http://linux.die.net/man/1/strace`).

The source code for the actual reactive grader was not available, so we used a dummy grader, which passes on queries from a file and accepts responses, but does not discards the responses rather than checking them.

We also ran each test in a "batch" mode. This used the same solution code (in particular, including a flush after each write), but directed input and output from and to files rather than an inter-process pipe. This has half the number of system calls (since output is flushed but input is buffered), but allows us to observe the effects of a large number of system calls in the absence of a reactive grader.

2.3. *System Setup*

Tests were run on a MacBook Pro with a 2.16 GHz processor and 1GB of RAM, running Linux 2.6.30. Some steps were taken to reduce the effect of random factors on the runtimes so that small differences between sandboxes would not be lost in the noise:

- All the important files (sandbox userspace, program to test, reactive grader, input and output files) were held on an in-memory filesystem (tmpfs), to eliminate the effects of disk caching and disk access time.

- To make the above more effective, all programs were statically linked, so that library code was also stored in RAM.
- Since the machine is a laptop, the CPUs are normally run at lower frequency when idle. For these tests, the frequency was forced to the maximum.

The same kernel was used for all tests, which included the patches necessary for the SACO sandbox. However, the boot-time kernel argument to enable this sandbox was only used where it was actively being tested. When the kernel argument is not passed, the patches may slightly increase the memory footprint of the kernel, but its security hooks are never called and so it should not degrade performance.

The USACO sandbox is written to deal with input and output files in a particular way. For our testing, it was modified to allow the sandboxed program to communicate directly with standard input and standard output. However, the security aspects of the sandbox were not modified.

The USACO sandbox also relies on the system calling conventions in the i386 architecture (i.e., 32-bit code). Although the kernel and operating system are 64-bit, the solution was compiled as a 32-bit binary (using the `-m32` option to GCC).

2.4. *CPU Affinity*

As well as comparing sandbox methods, we have investigated the effects of the number and configuration of CPU cores in the system. This was necessarily limited to two CPU cores on a single CPU package, since that was what was available in the test system. Four configurations were tested:

nosmp The `nosmp` command-line argument was passed to the kernel during boot, which limits the system to a single CPU, and also affects some code that needs to take additional steps in multi-processor systems.

none No special steps are taken. Processes run on both CPU cores, and are free to migrate.

separate Both the solution and the reactive grader (the process on the other end of the pipes) use CPU affinity masks to restrict them to a single core. The solution is locked to one core, the reactive grader to the other.

same Similar to the above, but both processes are locked to the same core.

3. Results and Analysis

Fig. 1 shows box plots for all combinations of sandbox and CPU affinity[2]. Times reported are the time measured for the solution process alone, computed as user plus system time. A number of observations can be made:

[2]The thick horizontal black bar indicates the mean, the boxes show the range covered by the central 50% of observations, and circles show outliers.

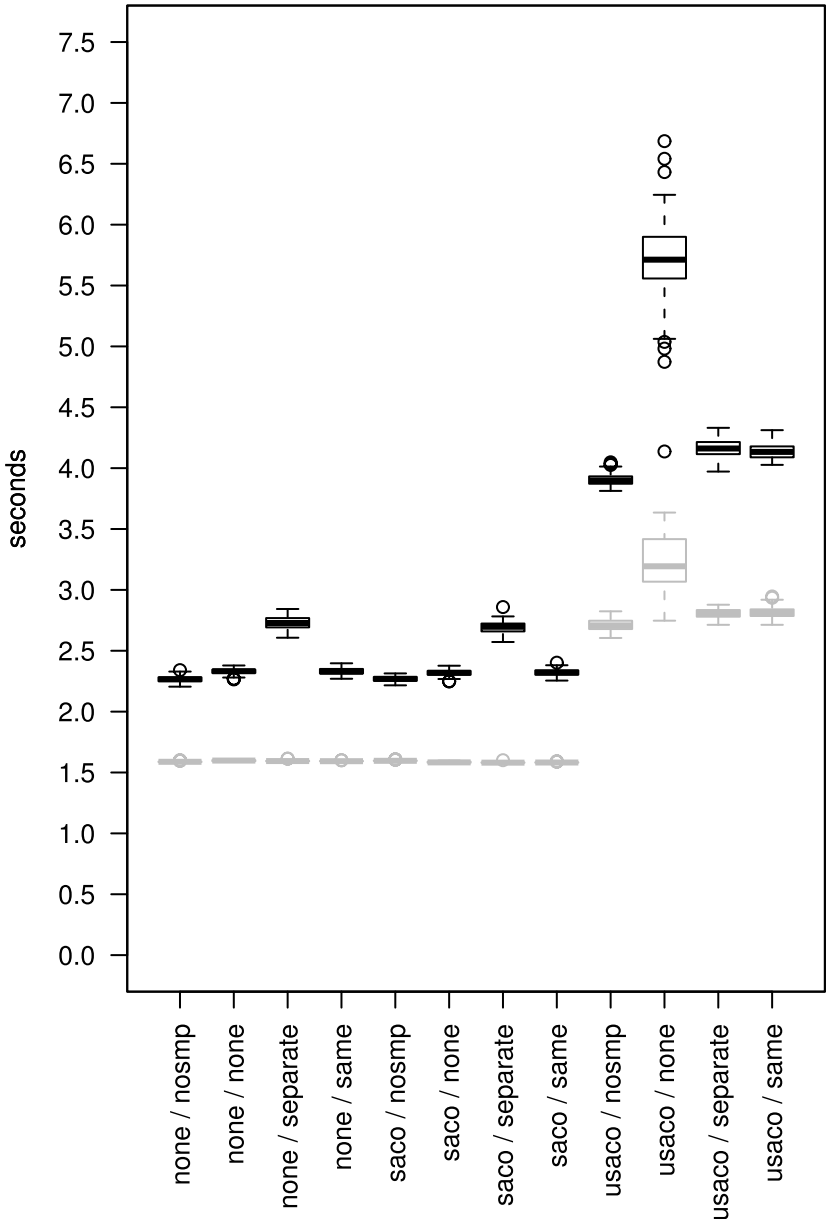


Fig. 1. Observed CPU times over 100 runs. Labels are of the form *sandbox* / *CPU affinity*. Black boxes show reactive grading, gray boxes show batch grading.

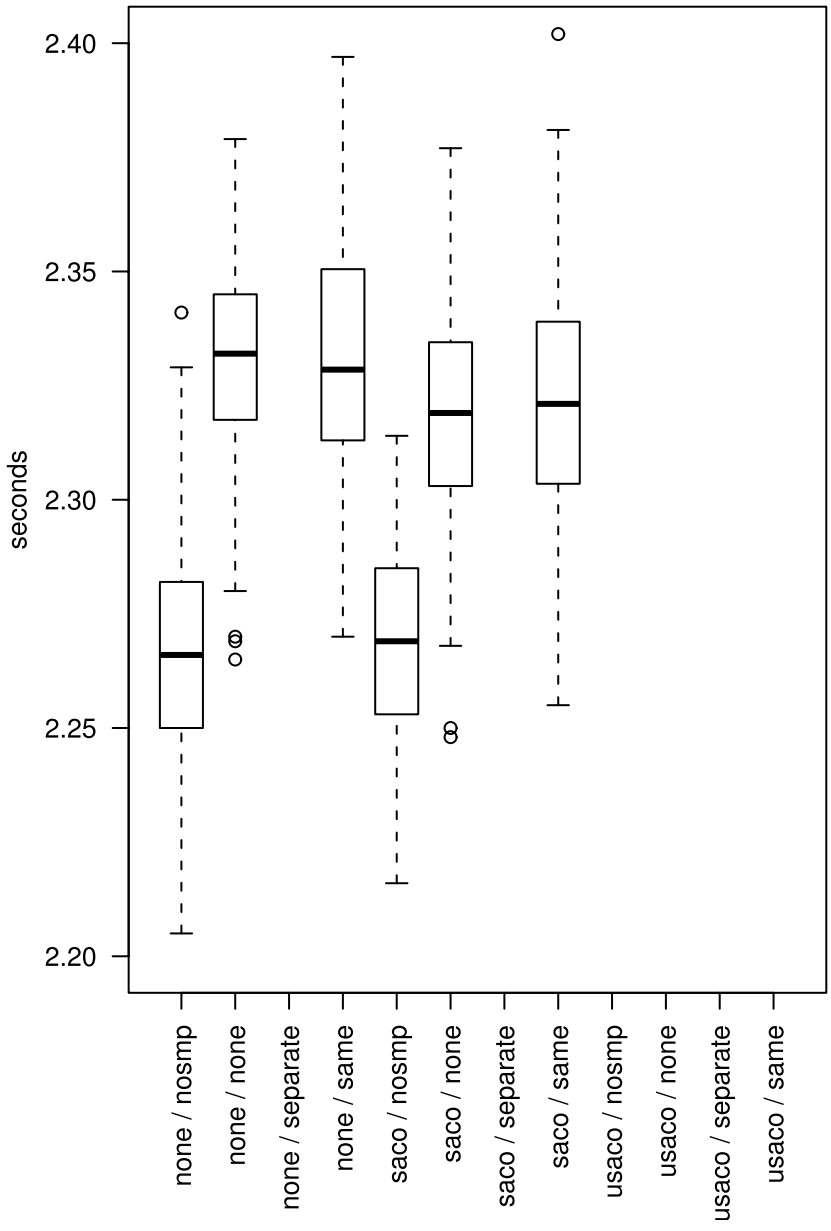


Fig. 2. Observed CPU times over 100 runs, close-up view. Refer to Fig. 1 for details.

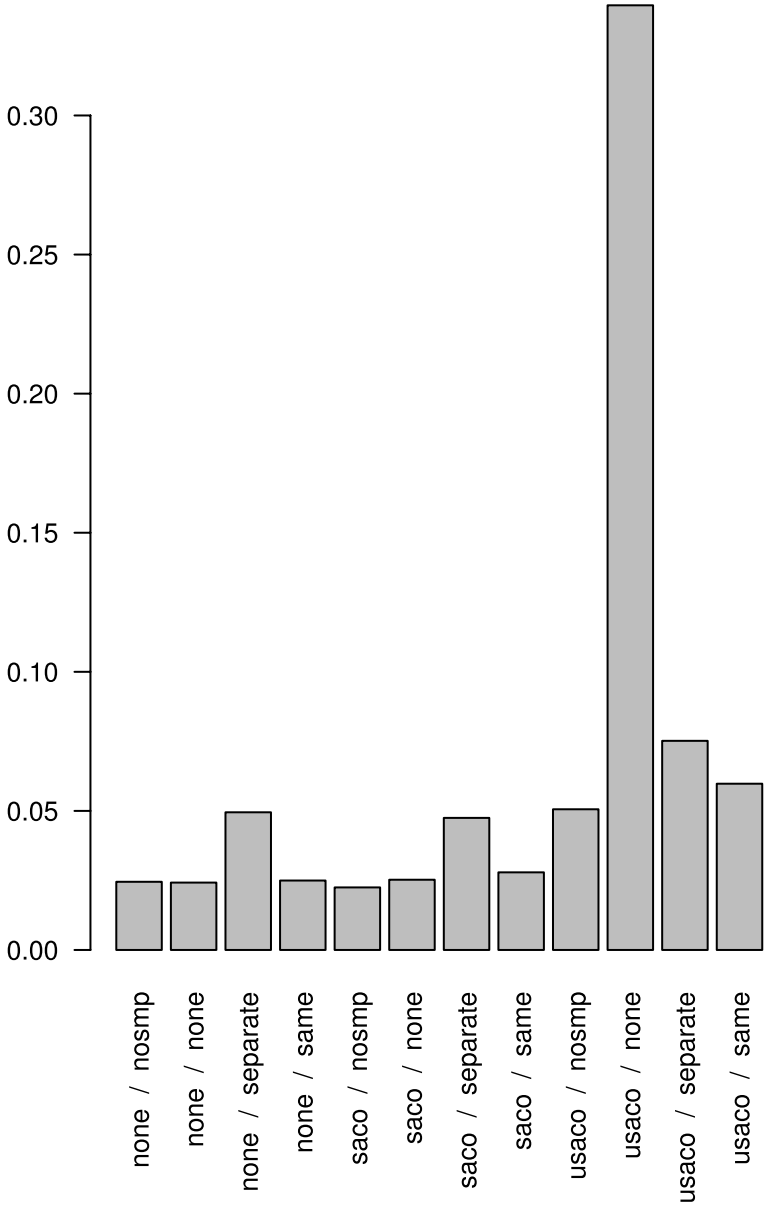


Fig. 3. Standard deviations of the test runs, for reactive mode only.

- The most striking result is that the USACO sandbox suffers both extremely high overhead and high variability when run on two CPUs without affinity masks, even in batch mode. While we have not performed a detailed investigation, a possible explanation is that the sandbox runs as a user-space process, and must examine memory in the solution process to determine whether the system call is safe. If the processes happen to be running on separate processors, then the memory may not be in the cache of the processor which needs to access it.
- For all choices of affinity and mode, the USACO sandbox introduces significant overhead.
- In reactive mode, forcing the processes to run on separate cores generally reduces performance. This may be due to the overheads of inter-processor communication, which may trigger cache writebacks or invalidations to ensure that the processors have a coherent view of the data.
- The batch times are generally lower, due to having half the number of system calls. Excluding the USACO grader, the differences between groups are statistically significant (other than between **same** and **separate**, which are equivalent without a reactive grader), but are clearly very small.

Fig. 2 shows a close-up view of some of the box plots, for reactive mode only. It is clear that when running with the SACO sandbox or without a sandbox, a single CPU yields lower times than two. These graphs also show that the SACO sandbox adds no statistically significant overhead (confirmed by t-test).

Fig. 3 shows the standard deviation associated with each test. It confirms that the USACO sandbox without affinity control suffers from much greater variability, while the SACO sandbox has no obvious effect. It also indicates that forcing the solution and the reactive grader to run on separate cores increases variation.

4. Conclusions and Future Work

It should be noted that the results show the measured run-times of the solution process. Lower run-times do not necessarily equate to faster evaluation, and indeed the **nosmp** tests may take longer to complete due to only a single core being available. From the contestants' point of view, timing should satisfy two requirements:

1. Times should have low variability, to ensure that evaluation is fair. If two contestants have functionally equivalent solutions, it should not be the case that one scores higher than the other due to random fluctuations.
2. The security system should not significantly alter times. While contestants ought to test their solutions on the evaluation system (this facility being commonly available in contests with on-line submission), it is nevertheless more convenient to be able to test locally with some expectation that similar timing will be seen during evaluation.

It is clear that ptrace-based security does not satisfy the second requirement where large numbers of system calls are involved. The first requirement can be met to some

extent with ptrace-based security, but only if CPU affinity masks are correctly configured on multi-core systems.

It should also be noted that this is a limited experiment, involving only a single CPU design, task, solution and test case, and so conclusions might not generalise. CPU cache and memory architecture in particular may significantly influence the effects of CPU affinity; and the pattern of timing in system calls may also have an effect.

Acknowledgements

Rob Kolstad kindly provided access to the USACO sandbox, and also suggested investigating CPU affinity.

References

Strace – Trace System Calls and Signals. `http://linux.die.net/man/1/strace`.
Fan, L., Peng, R. (2009). Task 2.3 Regions. In *21st International Olympiad in Informatics*, pp. 28–31.
Kolstad, R. (2009). Infrastructure for contest task development. *Olympiads in Informatics*, 3, 38–59.
Merry, B. (2009). Using a Linux security module for contest security. *Olympiads in Informatics*, 3, 67–73.

B. Merry took part in the IOI from 1996 to 2001, winning two gold medals. Since then he has been involved in numerous programming contests, as well as South Africa's IOI training program. He obtained his PhD in computer science from the University of Cape Town and is now a senior software engineer at ARM.

Olympiads in Informatics, 2010, Vol. 4, 95–103

© 2010 *Institute of Mathematics and Informatics, Vilnius*

Real Processes as Sources for Tasks in Informatics

Pavel S. PANKOV
International University of Kyrgyzstan
A. Sydykov str. 252, Apt. 10, 720001 Bishkek, Kyrgyzstan
e-mail: pps50@rambler.ru

Abstract. Most of tasks in informatics are set with a background story about real processes even though the circumstances seem to be strained a little. At the same time natural sciences (physics, chemistry, biology, genetics, astronomy, geology, etc. . .) contain many interesting laws and facts, some of which can serve as a natural base for tasks in informatics. A survey of some of the laws, facts and techniques (especially ones of discretization), chosen to illustrate a composition of genuine tasks, are proposed in the paper.

Key words: tasks, informatics, real processes, sciences, physics, chemistry, biology, geology.

1. Survey of Ways to Generate Tasks and the Aim of Paper

Burton and Hiron (2008) offered two opposite ways to create a good task: "To wrap an abstract task inside a good story" and "To look around and to take inspiration from real things". In Pankov (2008) we reviewed ways to generate task ideas based on "actors" and their actions in real and imaginary spaces and called such tasks *natural*. In other words, we have supposed that a good task should create a particular image in the mind of the contestant. In our experience, even questions using abstract mathematical spaces (Weeks, 1985) can be presented in a *natural* way in programming tasks. In Pankov and Baryshnikov (2009) we described some processes for taking any real (desirable) object (host country, city, university, sponsor, local sights, events, history, circumstances, etc. . .) and creating a task out of it for an informatics olympiad. The aim of this paper is to illustrate a possible involvement of scientific laws and facts into creating tasks in informatics but not to present ready-to-use tasks. Hence, we shall not give complete effective algorithms to all tasks.

We shall not consider the full procedure of creating a task with all of the corresponding restrictions, tests, etc. . . . It is not an aim of this paper and it was considered in details in Kemkes *et al.* (2007), Diks *et al.* (2008), Burton and Hiron (2008) and other publications. We shall give a brief description of "circumstances" or a background text of a task, which hopefully can be developed to full-grown.

We also shall not describe a sequence of tasks beginning from an evident one (with an algorithm to solve) and continuing to the more complex (where the problem of creating an appropriate algorithm arises).

In general, the following three types of tasks can be set if *the properties* of an object are given:

– Combinatory: how many *different* objects exist?

  REMARK. The *difference* between objects is also an important notion. Often, the more objects are considered to be equivalent to each other the more difficult is the task (the more difficult is application of standard algorithms). And there is a lot of equivalence for *real* objects (rotation, reflection, translation and equivalence of atoms) which makes tasks more interesting.

– Optimization: find the maximum (or minimum) of the all objects.
– Interaction: detect the object by a minimal number or a restricted number of requests (trials). Such interaction can be implemented either as an interactive task (the contestant's program calls the jury's procedure) or as a simple (batch) task (see Task 12).

By our observation, most of tasks generated in such way are too difficult (NP-hard). On low level olympiads such tasks may be given with fewer restrictions, so they can be solved by full sorting.

On high level olympiads special restrictions can be put in place to distinguish an interesting algorithm.

For instance, in string processing, restrictions may be put on number of given words, on lengths of words, or on the number of symbols.

2. Physics

2.1. *Crystals*

The following type of task is classical. We mention it for completeness as related to the physical notion of a crystal. The full object (pattern) is composed of repeated sample (or "tile"). Given information about the pattern, find the sample (or "the least possible"/"smallest possible" sample) this pattern can be composed of.

Also, a two-dimensional rectangle net is usual. A two-dimensional triangle, a two-dimensional hexagonal and a three-dimensional rectangle nets also exist and can be involved in tasks. An interesting example of a crystal is a snowflake, which can be defined as a bounded figure with rotation symmetry of 60^0 and mirror symmetry with respect to an axis (and, consequently, to two other axes) passing through the center.

Task 1. Starting with a black-and-white photograph of a snowflake with a known center and horizontal axis of symmetry, only some black points and some white points remain. Given the coordinates (pairs of integer numbers) of these points in a rectangle system of coordinates with the angle XOY equal 60^0. Can these points belong to an ideal snowflake?

Formulas to solve: clockwise rotation 60^0 : $X_{\text{new}} = X + Y$; $Y_{\text{new}} = -X$. Reflection with respect to the X-axis: $X_{\text{new}} = X + Y$; $Y_{\text{new}} = -Y$.

The following task imitates similar processes of crystallization and of systematic life expansion.

Task 2. Given a net or a graph (a net is a particular case of graph) and a natural number K. If a vertex of a graph is marked now then all its neighbors are marked at the next step.

How many vertices ("centers of crystallization", "vegetative planting stocks", "ant colonies", etc...) must be marked initially to mark the entire graph in K steps?

Such a task is relatively simple. By involving more than one type of crystals in the same media or more than one of (competing) species one can yield more variations and more of a challenge in a task.

2.2. *Conservation Laws*

Law of conservation of mass and of linear momentum:

There are some (massive pointwise) objects moving along a straight line. If two or more objects are too close and they clash then the operating person can slightly deviate some of them. Those which are not deviated, merge together. The velocity of the new/whole object is calculated using the law of conservation of linear momentum: if the objects had masses $M_1, \ldots, M_k$ and velocities $V_1, \ldots, V_k$ (positive or negative) then the velocity of the merged object is

$$V := (M_1 * V_1 + \cdots + M_k * V_k)/(M_1 + \cdots + M_k).$$

Task 3. Given number N of objects, (positive integer) mass $M[i]$, (integer) initial position $X[i]$ and (integer) initial velocity $V[i]$ of all objects, $i = 1 \ldots N$; $X[1] < \cdots < X[N]$.

Find the smallest possible absolute value of velocity of the merged object. Due to conditions of the task, the output is a rational number. So, it must be presented as <integer>/<natural> (as a fraction in its simplest form) where the HCF of the numerator and the denominator must equal 1.

EXAMPLE. $N = 3$, $X[1] = 10$, $X[2] = 20$, $X[3] = 30$, $M[1] = 100$, $X[2] = 500$, $X[3] = 104$, $V[1] = 7$, $V[2] = 20$, $V[3] = -7$. Answer: $7/51$ [mass of the merged object is 204].

Law of the conservation of the electrical charge.

Task 4. Given a graph, its V vertices are charged electrically (given integer nonzero numbers $C[i]$, $i = 1 \ldots V]$) and its arcs of given lengths $L[i, j]$ (natural numbers) are nonconductors, $i, j = 1 \ldots V$. Also, there is a given length $L1$ (natural number) of (conducting) wire. If the charged objects are connected with the conductor wire then their common charge is the sum of the all charges.

Find A) the smallest possible absolute value or B) the greatest possible absolute value of charge of any part of the graph which can be obtained by cutting the wire into pieces and connecting some vertices (objects) with these pieces along the arcs.

EXAMPLE. $V = 3$, $C[1] = 10$, $X[2] = -20$, $X[3] = -12$, $L[1, 2] = L[1, 3] = L[2, 3] = 100$, $L1 = 103$. Answer A: 2. Answer B: 32.

2.3. *Methods of Physical Investigation*

The process of balancing (with or without weights) is a source for many tasks.

Task 5. There is a balance scale and N objects of weights (kg) $W[1], \ldots, W[N]$ (given natural numbers). If difference between sides on the balance is greater than K kg then the scales turn upside down (K is a given non-negative integer). A robot can carry and put only one object on the scales at any given moment.

What is the minimum number of robots necessary to put all N objects on the scales?

EXAMPLE. $N = 6$, $W[1] = 1$, $W[2] = 8$, $W[3] = 5$, $W[4] = 20$, $W[5] = 20$, $W[6] = 1$, $K = 1$. Answer: 4 robots [in two steps].

Task 6. There are the scales and N weights (kg) $W[1], \ldots, W[N]$ (natural numbers) and an object of unknown weight $W1$. It is known that $1 \leqslant W1 \leqslant W0$; $W0$ is a given natural number; the HCF of $W[1], \ldots, W[N]$ is 1.

What is the minimum number of consecutive weightings necessary to detect $W1$ or to make the conclusion that it is impossible if the weights can be put A) on one of the scales? B) on both scales? (The three possible responses: the left scale is heavier; balance; the right scale is heavier).

EXAMPLE (a standard set of weights). $N = 4$, $W[1] = 1$, $W[2] = 2$, $W[3] = 2$, $W[4] = 5$, $W0 = 10$. Answer A): 3 weightings [$W[1]$ is not necessary].

Detecting particles. $N \times N$ square detectors form a big square. Particles fly above in straight lines. A particle flying across a detector (including its sides and vertices) sometimes activates it. A detector can be activated by more than one particle at the time. In such case, the precise number of particles crossing the detector is not detected by it.

Task 7. Given is an integer N and the list of activated detectors. Find the smallest possible number of particles which could activate these detectors.

EXAMPLE. $N = 5$, activated detectors: (1,1); (1,4); (5,1); (5,5). Answer: 2.

REMARK. Such tasks must be solved with integer numbers (with vulgar fractions). If division of numbers is used then there can arise mistakes because of rounding error.

3. Chemistry

3.1. *Chemical Formulas*

In this section we will not consider *real* chemical elements and molecules with their concrete properties; we will consider mathematical tasks arising in chemistry. Thus, we will use convenient denotations looking like chemical ones. Denote (conventional) chemical

elements with capital Latin letters, so the greatest possible number of elements does not exceed 26 (this is not essential). Let a number of atoms of each element in a molecule be written after the denotation of the element. Elements in a formula will be written in the alphabetical order.

So, the molecule of (conventional) water H_2O may be written as $A2K1$ or $P1Q2$.

Task 8. Given chemical formulas and the number of atoms of each element in these formulas find the greatest number of *molecules* of these types which can be made of these atoms.

EXAMPLE.

Input: Two formulas: $A5B1$; $B3C2$; three elements: A 20; B 10; C 3.

Output: 5.

$[4(A5B1); 1(B3C2)]$.

REMARK. If the mixture of these chemical substances is a gas then each molecule occupies same volume of space (in the initial approximation). So, the condition of the task is natural: find the greatest volume occupied.

Task 9. Given: chemical formulae before a reaction and possible chemical formulae after the reaction. Can such reaction exist from the mathematical standpoint of view? All listed chemicals must be involved. If it can, then find the smallest possible coefficients (natural numbers) in the formulae to make balance of chemicals before and after the reaction.

Solving. Obviously, this task is reduced to a system of linear homogeneous Diophantine equations (a positive solution is to be found). If two equations contain the same unknown then it can be excluded. So, we obtain one or more equations with different unknowns. Moving backward we obtain the solution if it exists. Outlines of the algorithm are seen as follows.

EXAMPLE 1 (iron oxidation).

Input: (before): $F1$, $O2$; (after) $F2O3$.

Output: Yes; $4 * F1 + 3 * O2 = 2 * F2O3$.

Solving of Example 1. The task is $X_1 * F1 + X_2 * O2 = X_3 * F2O3$, the system is $X_1 = 2 * X_3, 2 * X_2 = 3 * X_3$.

The first equation has the general solution $X_1 = 2 * T_1$, $X_3 = T_1$. Substituting we obtain: $2 * X_2 = 3 * T_1$, hence $T_1 = 2 * T_2$, $X_2 = 3 * T_2$; $X_1 = 4 * T_2$; $X_3 = 2 * T_2$. Taking the least possible value $T_2 = 1$, we obtain a solution.

REMARK. In real (simple) tasks the following algorithm is also valid. Choose one of the coefficients as 1. If all other coefficients are defined uniquely then they are rational numbers. Multiplying all coefficients by the LCM of all denominators we obtain the required natural coefficients.

EXAMPLE 2 (conventional).
Input: (before): $C2D3, B2C5, B3D2$; (after) $B3C3D3, C2D1$.
Output: Yes; $4 * C2D3 + 3 * B2C5 + 5 * B3D2 = 7 * B3C3D3 + 1 * C2D1$.

3.2. *Structural Chemical Formulae – Chemical Graph Theory*

In addition to a chemical formula, it is possible that bonds between atoms (or valence of each atom) are given. These bonds define a set of graphs. This set can be used to compose various other tasks.

Task 10. Given a chemical formula (atoms are vertices of a graph) and A) valences (number of arcs from each vertex) of all atoms or B) a list of atoms which each atom must be connected with. How many sufficiently different graphs (i.e. different chemicals) exist?

EXAMPLE A) (isopentans).
Input: formula $C5H12$; valence of C (carbon) is 4; valence of H (hydrogen) is 1.
Answer: 3 [five C atoms are connected with the following arcs: 1) 1–2, 2–3, 3–4, 4–5; 2) 1–2, 2–3, 3–4, 3–5; 3) 1–2, 1–3, 1–4, 1–5].

3.3. *Chemical Reactions*

There are M known chemicals; the first N of them are present.

Some of the chemicals can be obtained from other ones (we will only consider reactions that cause two chemicals to become one).

All reactions are given as four numbers $B1, B2, A$ (all different natural numbers in $1 \dots M$) and T (integer number denoting releasing heat, if $T > 0$, or required heat, if $T < 0$) indicating the A-th chemical is obtained of $B1$-th and $B2$-th ones.

Task 11. Find the most profitable way to obtain the given $A0$-th chemical of the list $M - N + 1 \dots M$ (if it is possible), i.e., such sequence of reactions $B1, B2, A, T$ that:
– all As are in $M - N + 1 \dots M$ and different; the last A is $A0$;
– each $B1, B2$ are either in $1 \dots N$ or of preceding As;
– all As except $A0$ are used in following reactions;
– the sum of all Ts has the greatest possible value.

3.4. *Methods of Chemical Analysis*

Task 12. Given the list of T trial chemicals, the list of U chemicals to be detected and the list of the results $R[i, j]$ (encoded as natural numbers in $[1 \dots K]$, K is a given natural number too) of reactions of i-th chemical of the first list with the j-th chemical of the second list, $i = 1 \dots N, j = 1 \dots M$.

Find the smallest possible number of reactions to detect the unknown chemical from the second list (if it is possible).

4. Genetics

4.1. *Reading Genetic Code*

The following situation is a classic example. We recall it for completeness.

Reading genetic code. There are many identical chromosomes and the task is to get to know the sequence of "letters" written on given chromosome. To read all the information on a long chromosome is too difficult, so chromosomes are split into pieces which are sufficiently small to be read easily. The splits are random.

Task 13. Given a set of words. Find the smallest possible length of a word containing all the given words as sub-words.

Task 14. Given a set of words. How many words of given length containing all the given words as sub-words exist? (If such words do not exist then output 0).

4.2. *Detecting Number of Chromosomes*

If two hereditary characters, defined by genes, are positioned on the same chromosome, then they are inherited simultaneously. Every gene of a child coincides with the corresponding gene of one of its parents. Suppose that some genes are of phenotype character (can be observed or detected in any way).

Task 15. Given the natural number $N > 2$ and set of M triples of words of length N: $P1$, $P2$, B fulfilling the condition: each letter $B[k] = P1[k]$ or $B[k] = P2[k]$, $k = 1 \dots N$.

Find the minimal number of subsets in a decomposition of the set $1 \dots N$ such that for every subset S and every triple $P1, P2, B$ the intersection $B \cap S = P1 \cap S$ or $B \cap S = P2 \cap S$.

REMARK. The researcher does not know the number of chromosomes and order of coding, so they have arranged ascertained characters arbitrarily.

EXAMPLE.

$N = 5$; $M = 2$;

$(TEWPT, EDWBV, TEWPV)$; $(DEWXT, EFWBT, EEWBT)$.

Answer: 3 [the first set: {1, 4}; the second set: {2, 3}; the third set: {5}].

The answer is the lower boundary for number of chromosomes of this species.

Solving of this task is relatively simple but demands fluency in treating sets.

5. Geology

Restoring the chronological sequence of sedimentary layers (of geological epochs).

Millennium after millennium, sedimentary layers are deposited onto the sea bed. Each epoch deposits its own layer. Due to the different geological environments, some layers

are absent in some parts of the sea bed. Researchers have taken samples in different places, and have separated each sample into layers, denoting all ascertained layers with letters.

Task 16. Given a set S of words. Each letter can be in each word only once; if one letter precedes other one, in any word, then the same must occur in other words too. Find a (long) word W containing these letters only (each letter only once) such that all given words can be obtained from it by erasing some letters. Is such word unique?

EXAMPLE.

Input: MTUG; TGH; TFH.

Output: not unique [three possible words: MTFUGH; MTUFGH; MTUGFH].

Solution. Probably, the following algorithm is the best.

Denote a set of non-empty words in S as S'. Firstly, $S' = S$.

$W = \mathrm{empty} - \mathrm{word}$; while S' is not empty {compare the first letters in all words in S': if there is the only one preceding to others then concatenate it to W and avoid it from all words in S' else {output "not unique" and stop}}; output W.

This task is relatively simple. But subsequent development in the area could turn some of the strata upside down. And then the following task arises.

Task 17. In conditions of Task 16, some (less than half of) words can be reversed. The quest is same.

6. Astronomy

There is a Sun in the center of a solar system and N Planets rotating around the Sun in circular orbits. When all inner Planets will overshadow Sun from the standpoint of the outermost (Nth) Planet simultaneously?

Choose the following measure of angles: the full rotation (of 2π radian) is equal to 1.

Task 18. Given natural $N > 1$, initial angles of all planets $A[1], \ldots, A[N]$ (as rational numbers between 0 and 1), periods of rotation of all planets $P[1] < \cdots < P[N]$ (as positive rational numbers and the maximal absolute values of angles of overshadowing $M[1], \ldots, M[N-1]$ (as (small) positive rational numbers). Find (if it is possible) the minimum number T (a rational number) such that at the moment T from initial moment the following inequalities will be primarily true:

$$\mathrm{abs}(\text{angle } N\text{th-Planet} - \text{the-center-of-Sun} - K\text{th-Planet}) \leqslant M[K],$$
$$\text{for all } K = 1 \ldots N-1.$$

REMARK. Actually, the planes containing orbits of different planets and of satellites, often do not coincide (eclipses of the Moon and of the Sun do not occur every month). But taking it into account only results in complicating the task too much, because the lines of intersection of these planes (so-called lines of nodes) constantly change.

7. Conclusion

We hope that this paper will promote the idea of involving laws, ideas and methods of sciences into informatics olympiads, making them more engaging for young people, and attracting contestants' attention to vast applications of informatics. Accordingly, this preposition can also inspire a greater interest of young people in learning sciences and perhaps even in helping to make an appropriate career choice in future.

References

Burton, B.A., Hiron, M. (2008). Creating informatics olympiad tasks: exploring the black art. *Olympiads in Informatics*, 2, 16–36.

Diks, K., Kubica, M., Radoszewski, J., Stencel, K. (2008). A proposal for a task preparation process. *Olympiads in Informatics*, 2, 64–74.

Kemkes, G., Cormack, G., Munro, I., Vasiga, T. (2007). New task types at the Canadian computing competition. *Olympiads in Informatics*, 1, 79–89.

Pankov, P.S. (2008). Naturalness in tasks for olympiads in informatics. *Olympiads in Informatics*, 2, 115–121.

Pankov, P.S., Baryshnikov, K.A. (2009). Representational means for tasks in informatics. *Olympiads in Informatics*, 3, 101–111.

Weeks, J.R. (1985). *The Shape of Space*. Marcel Dekker, Inc., New York.

P.S. Pankov (1950), doctor of physical-mathematical sciences, proffessor, corresponding member of Kyrgyzstani National Academy of Sciences, is the chairman of jury of Bishkek City Olympiads in Informatics since 1985, of National Olympaids in Informatics since 1987, the leader of Kyrgyzstani teams at IOIs since 2002. Graduated from the Kyrgyz State University in 1969, is a main research worker of Institute of theoretical and applied mathematics of Kyrgyzstani National Academy of Sciences, a manager of chair of the International University of Kyrgyzstan. Two his tasks were used at IOI's. In seventies he introduced the notion "validating computations" – strict proving of theorems by means of approximate calculations. His main field of research is interactive computer presentation of various objects. He developed natural motion in mathematical spaces which considered to be abstract earlier (such as Riemann manifolds) and independent presentation of notions (especially of verbs) of natural languages.

Olympiads in Informatics, 2010, Vol. 4, 104–112

© 2010 *Institute of Mathematics and Informatics, Vilnius*

The New Zealand Experience of Finding Informatics Talent

Margot PHILLIPPS

New Zealand Olympiad in Informatics
e-mail: margot.phillipps@gmail.com

Abstract. If Informatics, or indeed Computer Science, doesn't exist as a recognised, valued and suitably assessed senior high school subject, there is a challenge in finding students and training them to Olympiad standard. In New Zealand there are now 4 events which provide students with the opportunity to be selected for the New Zealand Olympiad in Informatics (NZOI) training camp. This has allowed the numbers of students able to be considered to expand from 5 in 2007 to 19 in 2010. In the other scientific disciplines, the olympiads all benefit from the presence of a sanctioned academic subject in high school. With the advent of a new "Body of Knowledge" (curriculum) and suitable assessment instruments to attract the more academically oriented students to Computing, it is hoped that the school system will become actively engaged in the NZOI program by encouraging students into contests.
Key words: informatics, olympiad, contest, training.

1. New Zealand and Its Education System

New Zealand is physically isolated, a long thin country spread over 2 major islands. The total population is just over 4 million with a concentration of 1.3 million living around Auckland city. The education system is free (public schools) with children typically starting on their 5th birthday. The school system consists of Primary (years 1 to 6), Intermediate (years 7 to 8) and Secondary (years 9 to 13). Until year 10, the curriculum is compulsory, but from year 11 onwards students choose 5 or 6 subjects per year. The main public assessment system for secondary school students is called NCEA, which is loosely modeled on the concept of mastery. The 3 levels correspond for the majority of students to years 11, 12 and 13.

Informatics does not yet figure in the curriculum. Senior courses in Computing are different in every school, as they are dependent on the skills and motivations of the teacher. The mechanisms for assessing it Computer Science an academic subject have been remarkable by their absence. For this reason, finding students in high school who may excel in Informatics has been an arduous task.

2. Identifying Informaticians – The Standard Route

In the 5 years of New Zealand's participation in the Informatics Olympiad, we have relied on the smallness of the country and personal connections to help identify potential students. There are 3 contests which are used to help our organization find interested students.

There is a tradition of New Zealand students sitting annual tests set in Australia such as those of Educational Assessment Australia (EAA), a testing organisation which specialises in annual large-scale testing of primary and secondary school students in the core curriculum subjects. The Australian Maths Trust holds a written informatics test, the Australian Informatics Competition or AIC (`http://www.amt.canberra.edu.au/aicsample.html`), and it has recently been introduced as an option for New Zealand students. The popularity of the EAA tests means students and their parents are used to the concept of paying a small fee for the student to sit an international test, and receiving comprehensive results, identifying the percentile the student is placed in. Names of schools with students from New Zealand who have performed exceptionally well in the AIC are communicated to the NZOI and their schools are then contacted. These students have been identified as exceptional problem solvers, and possibly algorithmic thinkers, and are invited to the training camp in summer for the Olympiad. Most will have no coding experience. Some online help is offered through a google group prior to camp. The value of such pen and paper competitions in Informatics is re-enforced by Burton (2008).

The second major competition through which students are identified is the New Zealand Programming Contest (NZPC `www.nzprogcontest.org.nz`). This contest has a 22 year history and is a team contest, initially designed for tertiary students. It is held over five hours on a Saturday in August and the contest is held at approximately 5 to 6 sites around New Zealand. The students can use any language the site will support. It is used by some students as a training ground for the ACM ICPC contest. In the last 5 years it has been opened to secondary students, and the problem set includes problems approachable by all secondary students with some knowledge of programming. Again the organiser of this contest, Phil Robbins, notifies NZOI of all school contestant's leaders and the students are then contacted and invited to the NZOI summer camp in December.

Such students are often self taught although there are small numbers of teams from schools where programming is taught. But to enter this contest the students have actually programmed, albeit typically in a non IOI language. They will almost certainly have had no formal algorithm training. Although the teams contain 3 students, it is very common for only one student to be interested in attending summer camp. Because the IOI is predominantly C++ (or C) and this is not often the language of first choice to begin learning programming with, this creates the necessity for some pre-camp training.

The third major competition is the Australian Computer Programming Contest run from the University of Southern Queensland by Michael de Raadt (`http://www.sci.usq.edu.au/staff/deraadt/acc/index.html`). This is also a team (of 3 students) event but it is held purely for junior (up to age 15) and senior high school students. It is typically held over a 10 day period in August and the school can nominate which

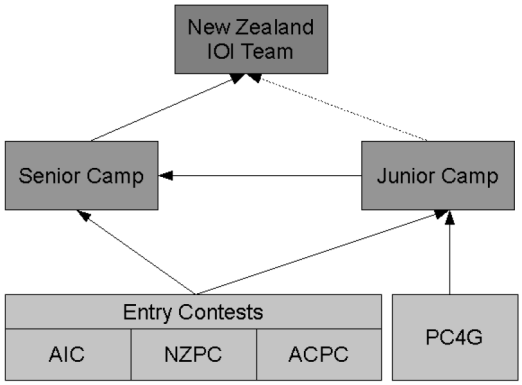


Fig. 1. Pathways to the New Zealand Olympiad in Informatics team.

2 hour time slot they wish to enter the contest in. The language can be of the students' choice but few students choose C++ (or Pascal). The organiser notifies NZOI of the schools which entered, and again the students who entered are invited to consider summer camp.

Fig. 1 shows these 3 competitions (the AIC, the NZPC and the ACPC) as "feeding" the summer training camps of the New Zealand Olympiad in Informatics organisation. Sample problems of these 3 contests are included in Appendix 1.

3. Identifying Informaticians – A New Proposition

It is relatively obvious that there is a gender disparity in much of the western world where informatics is concerned. There are many small inroads being made into addressing equity issues and one of these is a spin off from a Young Women's Programming Contest which ran from 1988 to 1999 in Auckland, New Zealand. Its primary aim was to attract young women to consider IT/Programming as a tertiary study choice and the top prize included a scholarship for each of the 2 members of the top team (Costain, 1999).

Although this contest became defunct, a number of the women originally involved in its organisation agreed it was worthy of resurrection, but in a different format. In 2008, a Programming Challenge 4 Girls (PC4G) was held for teams of 2 year 10 girls – (approximately 14 years old). The model is consistent with the ideas of Fisher and Cox (2006). It was held on one day late in the school year at a tertiary institution which hosted the catering, labs, staff and some of the prizes.

In 2009, it was held at 4 sites around the country and the aim is for many more in 2010. Marked work was graded into gold, silver, bronze and participation, and specially designed gold, silver and bronze medals were awarded.

The day consists of a morning's instruction and practice, followed by a challenge in the afternoon. Alice was chosen as the language as it is extremely easily learned, there is a lot of resource already developed and its colour and imagery is appealing to younger high school students. While the judging takes place, the girls are shown highlights of the host

institution or given something interesting IT-related to do or watch. A small presentation ceremony is then held where prizes and the medals are awarded.

The model of teaching and practice in the morning was chosen partly because many schools have no teachers competent in any form of programming. So the teacher sits with their teams and learns alongside the girls. While the girls sit the challenge the teachers are offered some professional development workshops related to Computer Science. This has proven very popular, with the teachers maybe recognising for the first time the power of their subject to intellectually challenge their students.

Fig. 1 shows this challenge feeding the Junior summer training camp.

4. Junior Summer Camp

With many students effectively being totally self-identified for the summer training camp, it became obvious that not all of those who attended camp coped with learning algorithms, and/or implementing them in C or C++.

So in January 2010, a Junior stream was added. Unlike the Australian Junior camp, which is a "baby algorithms camp", the Junior camp was to get students used to C and some basic searching and sorting algorithms (including a Bubble sort, despite Barack Obama's advice!). The intention is that the students will return to Senior camp the following year.

To find junior students we invited those from the (older) identifying contests to self-select. As a guide they were asked if they could solve one of the questions worth 10 points from the NZPC *on their own* (The NZPC has 3, 10, 30 and 100 point problems). Typically they would have worked in a team to solve one of these, so they were asked to honestly assess if they could do this individually.

In addition the girls who were in the top team from one site of the PC4G were approached and one of the girls decided to come to camp. For future years, the top teams from all sites for the PC4G will be invited.

The PC4G is not only a way of finding new programming talent, but it is also a new way for encouraging girls into programming.

Initially there were 6 students in the Junior camp, including 2 girls, although it became evident that one student had sufficient C to move to the Seniors camp. The remaining 5 learned about binary, how integers, characters and real numbers are stored, and enough C to implement simple sorts and searches. There were lectures and problems each day, with 3 tutors (one of whom was a girl) on hand to assist.

Two Junior camp student's reports are included in Appendix 2, illustrating that the inclusion of a Junior stream was worthwhile.

5. Senior Summer Camp

New Zealand first entered the IOI in 2006, with 4 students picked out of 5 who both entered the NZPC and were interested in being trained further. There was no camp that year but some ad-hoc training took place for 3 of the students.

A January Summer camp was established in 2007, with the help of the New Zealand Maths Olympiad. Only 4 students attended and staff who volunteered were naturally disappointed. However the volunteers have been remarkable in their loyalty and have continued to come to an increasing number and quality of student at camp. With our first success (a silver medal) in 2008 and with 2 bronze medals in 2009, the general perception of the lecturing staff is that we are attracting the right type of student and in (just) sufficient numbers.

By 2010, 13 students self selected for Senior summer camp, plus one moved from Junior camp (several of the Juniors competed for a place on the team and in fact scored better in several contests than some Seniors students).

6. The Future

The Ministry of Education has accepted the need for reform, although has retained Computing under one of the existing eight learning areas, namely Technology. The ministry is now overseeing the process of translating a "Body of Knowledge" for Computing into Achievement Standards. These are the accepted assessment tool for senior high school students within the NZ Education system (Many private schools and an increasing number of public schools offer alternative assessment regimes, in particular, Cambridge and the International Baccalaureate).

There are 5 strands to the Digital technologies Body of Knowledge, one of which is Programming and Computer Science. It is yet to be seen if this strand will develop into a fully-fledged course with sufficient Achievement Standards by year 13, but many of those working and advocating in this space are hopeful. The time frame is currently being negotiated, but many are now hopeful that the new level 3 (for year13) standards will be functional in 2012.

One of the battles has been to have the discipline accepted as an academic subject in schools, rather than the "Computing as Dumping Ground" model which has operated in many schools until now. With the hoped for recognition that should accompany the educational reform, NZOI can only benefit from having a greater talent pool to train and select from.

Appendix 1. Samples of Problems from the Entry Level Contests

1.1. *Sample Problem from the Australian Informatics Competition*

Packing

David wants to pack an aeroplane with bags. He has several large bags, medium bags and small bags. Each large bag weighs 7kg, each medium bag weighs 5kg and each small bag weighs 1kg. The aeroplane has a fixed weight limit w. David must meet this weight limit exactly, and would like to use as few bags as possible to do it.

David has devised the following set of rules for packing the plane.

Start by placing large bags into the aeroplane one at a time, until he is less than 7kg beneath the weight limit.

Continue by placing medium bags into the aeroplane until he is less than 5kg beneath the weight limit.

Finish by placing small bags into the aeroplane until he reaches the weight limit precisely.

For example, if the weight limit were $w = 20$kg, then David would proceed as follows. He would place two large bags into the aeroplane, giving a total weight of 14kg and leaving him 6kg beneath the limit. He would then place a single medium bag into the aeroplane, giving a total weight of 19kg and leaving him 1kg beneath the limit. Finally he would add a single small bag, bringing him to 20kg precisely.

Unfortunately David's method does not always use as few bags as possible. Consider a weight limit of $w = 25$kg. Here David would use three large bags (weighing 21kg) followed by four small bags (coming to 25kg total). Overall David has used seven bags, but he could have made 25kg using just five medium bags and nothing else.

Your task is to work out the smallest weight limit w for which David's method does not use as few bags as possible. What is the rightmost digit of your answer? (For example, if you think the answer is 48 then you should answer 8.)

(a) 0 (b) 2 (c) 4 (d) 6 (e) 8

1.2. *Sample problems from the New Zealand Programming Contest*

Problem B Palindrome Numbers 3 points

A word is a palindrome if it reads the same backwards as it does forwards. For example, words such as 'radar' and 'sees' are palindromes.

Mrs Jones, a primary school teacher, thought it would be good for her pupils if they could tell her whether a number had the same property, i.e., it read the same backwards as it did forwards. Numbers like 121 or 12421 would qualify, numbers like 123 or 1231 would not. Numbers like 10 would also not qualify – even though 10 can be written palindromically as 010, Mrs Jones is restricting this to numbers written in the normal decimal way, with no leading zeros.

This problem asks you to perform this task on each of a series of numbers.

Input to this problem is a series of integers (all between 1 and 99999) each on a separate line. The number 0 will be the last line of input and should not be processed. Output will be one line for each line of input, containing just the word 'yes' if the number qualifies as a palindrome number, 'no' if it does not.

Sample Input	Sample Output
121	yes
1231	no
12421	yes
0	

Problem E Earrings 10 points

At Pascal High School, lots of young girls insist on trying to get away with wearing non-regulation earrings. Mr Sneddon, the Associate Principal, sees red every time he spots a pair of long dangly earrings and confiscates them.

He keeps a numbered list of the girls from whom he has confiscated earrings. As he takes them, he uses a permanent marker to record the number of the owner on the back of each earring. Being a bit of a control freak, he also adds a letter to each one – A or B.

At the end of each term and after completing after-school detentions, the girls are to come and collect their earrings. Unfortunately one day Mr Sneddon dropped the envelope he keeps the earrings in and at the end of the term there is one earring that he cannot find.

Please tell him the name of the angry girl who only gets one earring back.

Input will consist of a number of scenarios. Each scenario will contain:

- A number n $(1 \leqslant n \leqslant 100)$ on a line on its own, which is the number of girls he has confiscated earrings from.
- n lines each containing a girl's full name (at most 60 characters in length).
- $2n - 1$ lines of data with a girl's number followed by a space then either an 'A' or a 'B'. These lines represent the earrings in the envelope: the number represents the position of the girl in Mr Sneddon's list, with 1 being the first girl. A girl's number will occur twice at most, with a different letter (A or B) for each number.

The last line of input will be a '0' on a line by itself. This line should not be processed.

Output should consist of the scenario number followed by the girl's name whose earring is missing, separated from the scenario number by a single space.

Sample Input	Sample Output
3	1 Alison Addaway
Betty Boolean	2 Helen Clark
Alison Addaway	
Carrie Carryon	
1 B	
2 A	
3 B	
3 A	
1 A	
2	
Helen Clark	
Margaret Thatcher	
1 B	
2 B	
2 A	
0	

1.3. *Sample Problems from the Australian Computer Programming Contest*

What are Well Ordered Numbers?

An integer is considered well ordered if each digit value increases from left to right.

For example, 123 is well-ordered because $1 < 2 < 3$, but 285 is not well-ordered because $8 > 5$.

Instructions

Write a program that will find and display all possible three digit well-ordered numbers.

Your output should contain eight numbers per line, with a tab character between each number.

Output should be in ascending order.

No other output should be produced by your program other than the numbers.

Example Output

123	124	125	126	127	128	129	134
135	136	137	???	???	???	???	???

...

2. Appendix 2: Summer Camp Reports

Summer Camp Report (1)

The NZOI Junior Training camp was a great learning experience for me. I stayed in Auckland for 5 days, learning to code some basic C/C++. I found that it was all explained very well, at a level that we all understood. Each day, we learnt something new, and then put that new knowledge to good use in a series of problems we had to solve. Some of these were easy, while some provided quite a challenge. I still have one or two codes that don't work quite right!

We ended the camp with a test, which took into account all the skills we had learnt over the course of the camp. It was much more difficult than the practice problems had been as we couldn't get much help from the tutors. Still, it was good to be able to measure my skills against those of the other juniors.

Camp wasn't all hard work, though. We had a lot of free time outside of training hours, which we spent either with fellow attendees, or "hard at work" on our computers "finishing the problems from the day", better known as surfing the internet. We even had an outing to a pool one afternoon, which was a great chance to better get to know our peers.

This camp was an awesome opportunity to learn more about coding, and computers in general. It helped me decide that I want to continue working in the computer world as a career, and I really hope I get to return next year as a senior.

Summer Camp Report (2)

On the 7th of January I flew up to Auckland where I was met at the airport by Margot Phillips and a few members of the NZOI camp. Being my first time flying alone it was a relief to see them just outside the gate exit! Shortly afterwards we were taken to the Grafton Hall's of Residence, where we dropped off our bags.

Then we walked to the University of Auckland's computer labs, which was to be our main workplace for the duration of the camp. After learning all about binary, hexadecimal and decimal numbers and various other things, we went back to the Halls of Residence to have dinner. The next morning we began to learn about C++.

It was quite cool, as I couldn't make any sense out of it before the camp, having learnt to program with BASIC.

Over the next few days I learnt a lot about the basics of C++, including variables, strings and arrays. Towards the end of the camp we began to do a few Informatics problems of our own. It was good after numerous tries to finally get 100% in the problem! On the last day in the Junior Camp we had a little contest of our own. It was quite interesting competing in an Informatics contest, albeit a small one.

Overall, the Informatics camp was a great experience and I hope to participate in more programming competitions this year, and hopefully I will be able to go to the senior camp next year.

References

Burton, B.A. (2008). Informatics olympiads: challenges in programming and algorithm design. In: Dobbie, G. and Mans, B. (Eds.), *Proc. Thirty First Australasian Computer Science Conference (ACSC 2008)*, Wollongong, NSW, Australia, CRPIT, 74, ACS, 9–13.

Fisher, M., Cox, A. (2006). Gender and programming contests: mitigating exclusionary practices. *Informatics in Education*, 5(1), 47–62.

Costain, G. (1999). Benefits of holding young women's programming contests? In: *Living Science Conference*, Wellington, New Zealand, pp. 155–164.

M. Phillipps was educated in mathematics and philosophy and received a BSc from the University of Otago and a BA from the University of Victoria. After 8 years working as an analyst-programmer, she undertook a year's teacher training diploma before joining the Auckland Institute of Technology (now AUT) to teach introductory programming and databases for 15 years. She taught similar courses at Unitec (Auckland) for 3 years before beginning as a high school teacher in mathematics and programming. In 2006, attendance at the IOI workshop began a voluntary career in adding New Zealand as a participant to the IOI, becoming the international director on the Board of the Computer Science Teachers Association (CSTA) and becoming the executive director of the Programming Challenge 4 Girls. She is currently the qualifications administrator for ACE training, a private computer training company.

Olympiads in Informatics, 2010, Vol. 4, 113–119

© 2010 *Institute of Mathematics and Informatics, Vilnius*

Validating the Security and Stability of the Grader for a Programming Contest System

Tocho TOCHEV, Tsvetan BOGDANOV
Sofia University, Bulgaria
J. Bourchier 5, 1164 Sofia, Bulgaria
e-mail: tocho.tochev@gmail.com, tsvetan.bogdanov@gmail.com

Abstract. Automated judging systems have had the tough task of ensuring the normal proceeding of programming contests for a long time. There are numerous ways proposed to secure the execution of a contestant's solution and many more actual implementations. In this article we review some criteria which the core of such systems must meet in order to be considered stable and secure. We will focus only on the core, as the functions it performs are similar across the variety of existing systems and it is the part of the system designed specifically to withstand attacks. In fact these criteria were created with their respective test cases in order to verify the grader we used in IOI 2009.

Key words: contests system, judging system, sandbox security, grader security, grader testing, grader test harness, "black box" testing, IOI 2009.

1. Introduction

Competitions in Informatics are used to evaluate the algorithmic thinking and programming skills of their participants. The first competitions involved judges manually reading and verifying the source code that the competitors submitted to them. With the evolution of the contests and of the difficulty of the problems this task became extremely daunting and potentially inaccurate.

This is how a myriad of automated judging systems were born – Moe, PC^2, USACO's, Top Coder's, Spoj0, SMOC are only some of the examples.

Of course, this presented new ways for people to interfere with the normal workflow of the competition by exploiting security loopholes. For example, to be able to run their programs using more resources than they are allowed, gaining insight about the test data, gaining access to a solution from another person, hindering the participation of other competitors, etc. The motivation for such misbehavior can range from material prizes, such as money, to acceptance in a better university. Therefore, it is important to have the automated system as secure as possible.

Unfortunately, the current implementations are uniform neither in architecture, nor in their approach to performing the submit evaluation. In this article we are going to focus on validating the security and stability of the core part of a programming contest judging system (the 'grader' as we will call it). The term stability stands for verifying the

correctness of the normal submits, while security focuses on filtering out the malicious ones.

Similar studies have been made by Forišek (2006), where he gives a classification of a number of attacks against programming contest systems. On the other hand the criteria we review are based on their usage in functional tests for SMOC, the system we used for IOI 2009 and the Bulgarian competitions in informatics. The need to create such a list arose when we unified the way SMOC and Moe (Mareš, 2009) sandbox the contestant's program execution. Moreover, any significant refactoring on a grader should be followed by thorough examination of the resulting module. We have limited the scope of these criteria to the grader, and will not discuss any attacks related to the parts of the system visible to the contestants (such as securing workstations, traffic sniffing, activity auditing, hacking the web server serving the contest system, "Denial of Service", and others).

Unfortunately, there is no such thing as 100% guarantee that a software solution is stable and secure. However, there are certain actions that can be taken in order to raise the confidence in it. The most common are:

1. Peer reviews of the source code.
2. Real-world testing by interaction.
3. Automated testing.

The code reviews are important and can identify problems the other methods cannot. However, as we are only human, they are not very reliable. Real-world testing by interaction, such as online contests, is a nice way to test the overall system. Unfortunately, it does not cover all cases and organizing such an event is time consuming. In contrast automated testing is "cheap" (write the test once, run it after you have changed the code), has good coverage, and can prevent regression bugs. It also allows for a quick inspection of a new server setup (e.g., different OS).

We will focus only on automated testing, however, it is always preferable to use a combination of these methods for verifying a programming contest system.

2. Implementing a Test Harness Around the Grader

2.1. *Functions of the Grader*

As we already stated the grader is the core of a programming contest judging system. In different systems this component may be comprised of several subcomponents, and it may have different levels of coupling with the other parts of the system.

In order to create a test harness we first need to know the functions that the grader performs. We define them as:

1. Compiling the source code.
2. Running the resulting program on some input in "restricted" mode (also known as sandboxing).
3. Running some checks on the program's output (for instance check how compatible the program's and the judge's outputs are).

The requirements that the grader must meet include:

1. Enforcement of the programming task's resource limits (processor, memory, number of threads/processes spawned).
2. Enforcement of unaided program runs – preventing communication with outside world, usage of temporally files or reads of the judge's output files; limiting precomputing, etc.
3. Enforcement of the usage only of tools approved for the competition (for example, restrict unsanctioned libraries or calls to external programs).

As can be seen the implications of an incorrect or compromised grader can be severe – usage of additional resources, obtaining access to the judge's output files and simply printing them, sending vital information about the input files back to the contestant, using pre-computed values between runs, "blocking" the grader machine and therefore hindering the whole competition, or in other way altering the results of the competition.

2.2. *Ensuring Stability*

However, before the grader is secured it must first properly perform its functions in an environment with no malicious programs. To verify that, we decided to create a simple task for each of the task types that we support. So we ended up with:

- Batch tasks – *aplusb*: Given 2 integers output their sum.
- Reactive tasks – *binsearch*: Given a range $[A, B]$ guess a number in that range, by asking "Is it x?", and receiving "up" and "down" hints.
- Output-only tasks – *output*: Give a file with a single line.

In order to ensure the stability we decided that we must cover at least the following cases:

- Correct solution.
  For each task type and for each programming language we have a correct solution. This assures us that all compilers are present and working, as well as that the whole process runs smoothly.
- Wrong answer solution.
  One wrong answer solution (for instance "off by one") per task type. As well as some solutions that are partially correct.
  For the output only task we had a solution which failed the format checker (which is responsible for accepting the output-only solution for judging).
- Time-limiting solution.
  For batch and reactive tasks we implemented a test solution that outputs the right answer and after that falls in an infinite loop. We did this in order to guarantee that even if the contestant's program is correct but fails to terminate we assign to it a time limit exceeded resolution.
  Other test solutions that need to be included for reactive tasks are deadlocking ones. Note that the deadlock might occur in both the contestant's judge's programs – an excessive read would cause the contestant to wait forever and not writing enough information might cause the judge's program to block.

It is important to note that writing a solution exceeding the allowed time limit might be tricky. For example, the following code will be optimized by the compiler "`for (long long i=0;i<(1<<50);i++);`" to constant time.

- Memory limit exceeding solution.
  We have C solutions with both dynamically allocated memory, and with static allocation. The amount of memory allocated needs to be slightly over the actual memory limit. A correct solution which uses slightly less than the actual memory limit should also be used.
  If there is a separate stack limit it should also be checked. And if there is no such limit, this also needs to be verified (i.e., static allocation should be done in the stack).
- Runtime error.
  There should be several sub-tests for runtime errors. A simple division by zero can be used to achieve a 'Floating point exception'. There also needs to be a solution that has a non-zero exit code as that can also signify runtime error. Furthermore, it is nice to have a simple invalid memory reference solution as some graders may filter these separately.
- Use of library functions.
  For every allowed language there need to be sample solutions that include permitted and restricted libraries. For example, for C++ there should be checks for `stl` (e.g., `vector`), `boost` and possibly `tr1`.

2.3. *Ensuring Security*

As we have stated the grader has three main functions – compilation, program sandboxing, and output evaluation – and each of them can be attacked.

2.3.1. *Attacks During Compilation*

Attacks aimed at compilation are very dangerous as they can make the grading system unresponsive or expose the judge's solution.

- Excessive submit size.
  The simplest "attack" is just pre-computing the answers for every possible test case, and inserting them in what becomes a 50MB source file submitted to the system. Therefore, the submit file size should be limited. Note that the actual limit may vary from task to task, since some tasks may allow more precomputing than others.
- Referencing forbidden files.
  Another attack is including files in the compilation that are not supposed to be included, such as "`#include<boost/lambda/lambda.hpp>`", or the more aggressive and dangerous "`#include"../solutions/judgestask1.cpp"`".
- "Denial of Service" and compile-time exploitation.
  Failing to impose reasonable limits (both time and memory) on the compiler can lead to pretty nasty attacks, such as "`#include"/dev/zero"`" (running "forever" while consuming more and more memory). A lack of compilation limit can

also be leveraged by the competitor to gather useful data using "Template metaprogramming" (e.g., compute the first N factorials or prime numbers, etc.). It is also possible for the contestant to cause "Denial of Service" by consuming too much memory or CPU during compilation.

2.3.2. *Attacks During Sandboxing*

As mentioned graders can use different ways of sandboxing – syscall interception (Mareš, 2007), virtualization, linux security modules (Merry, 2009), java virtual machine security profiles, and others. Depending on the type of sandboxing there are specific ways to try an attack.

However, the general things that we do not want contestants to do while sandboxed are:

- Be able to read/write files/directories.
  There needs to be a test for at least some of the important files that should be forbidden – the judge's solution, additional input files, output files, checkers and the grader itself. Some attention should also be paid when using syscall interception and killing the program when it seems to access 'unneeded' files as we found out that the current standard libraries of the programming languages seem to access some non-obvious 'files' (for instance, `qsort` can require access to `/proc/meminfo`).
  Also the policy which is taken towards the people who try to open files should not always be immediate ban, as we have witnessed lots of submits with forgotten "`freopen("mytest.txt", "r", stdin);`" statements.
- Be able to open sockets/access the network.
  It is imperative that the solutions cannot open sockets either as a server, or as a client. A break in this policy might allow the submit to mimic some important part of the system or send out vital information about the tests. Furthermore, this restriction does not only cover internet sockets but also any other protocol used by the contest system (e.g., unix domain sockets).
- Be able to spawn multiple threads.
  We usually want to check that the competitors cannot create multi-threaded applications as this might give them an unfair advantage. However, there might be competitions where this is actually encouraged.
- Be able to spawn multiple processes.
  The simplest test case here would be just using "`fork()`". Although, having multiple processes poses higher security threat than multiple threads, not every time that a contestant does an exec call, he should be banned. For example, some people under Windows use "`system("pause");`" for debugging and forget to comment it out before submitting.
- Be able to raise their privileges, or be able to break the sandbox.
  This is the class of attacks that are most sandbox-specific. The test cases can include buffer overflow attacks, `setuid` calls, breaking out of `chroot`, exploiting vulnerabilities in the system call wrappers, and others.

Although, it would be interesting to make a collection of such programs, it may be the case that they have to be tuned a little bit for the different sandbox classes.

- Protecting the judge's module in reactive tasks.
  In reactive tasks the judge's module must be protected against any attempts on its integrity. It is important to note that even a well sandboxed contestant's program can be successful in an attack if the judge's module is not written properly (e.g., allows buffer overflows from its input). Such an attack is difficult to perform and close to impossible unless the contestant is provided with the module's source code. Unfortunately, it is also hard to create a unified test case.
- Denial of Service during sandboxing
  There are a virtually infinite number of attacks in this class. However, there are several easy-to-test examples. First, the memory restriction model needs to disallow allocating any memory above the set limit. Any loophole (e.g., if the memory used is checked in regular intervals) might lead to exhaustion of the system resources (CPU or memory) and therefore non-responsiveness of the evaluating machine.
  Another check to consider would be outputting too much information (either on `stdout`, or on `stderr`). Although, the competitors might do this unintentionally, it can also lead to abnormal grader behavior. It can also lead to waste of disk space which will manifest itself later and should generally be avoided.

2.3.3. *Exploits During Checking*

The attacks that fall into this category are the result only of the output that the competitor's program has made. Mostly these exploits are not intentional, and are results from lack of foresight from the judges. Take for instance the segment "`while(i!=-1) {scanf("%d", &i); ...}`" in the judge's checker. It will lead to an infinite loop if the contestant never outputs -1. Therefore some precautions should be taken in terms of limiting the resources available to the checkers. Obviously, there are no checks that can cover large portions of the errors that can be done in checkers. However, a test that can be made with a custom checker that hangs and thus verifies the grader fallback mechanisms, in case of malfunctioning checker, are working (e.g., this is reported to the judges).

3. Conclusion and Further Work

It would be great if there was a complete checklist that would cover all possible vulnerabilities, however, such a thing cannot exist. What we have shown in this article is what we think are the most important tests cases. Every system for judging programming contests should be able to pass them in order for it to be considered at least somehow stable and secure.

Having such a checklist proved very useful for us during the preparation for hosting IOI 2009. We needed to organize a series of competitions, and each one of them introduced new challenges for the grading system – various operating systems, hardware configurations, as well as minor changes in the judging criteria. As part of the setup for

each of these events we needed to verify the functionality of the grader. Thanks to the test suite we painlessly exposed several issues and gained confidence in any modifications that we have made.

Furthermore, some of the test cases can be reused for verifying other functional areas. Such an example was the protocol change we made between the grader and the dispatching module. Moreover, we used the test solutions as model contestants' submissions to create load tests. However, notice that a comprehensive load test also includes solutions specific to the competition. Of course, for any large competition (such as the IOI) the host also needs to run a series of other tests.

For the benefit of anyone interested, we provide our tests as part of the SMOC's source code[1]. However, it seems to be a good idea to setup "a universal grader test repository", open to everyone, accepting proposals from contest system maintainers, in order not to duplicate the effort that goes into testing a grader.

Perhaps this work can be extended and can lead to a number of stability and security checks that every IOI host system can be verified against.

References

Forišek, M. (2006). Security of programming contest systems. In: *Information Technologies at School*, 553–563.

Mareš, M, (2007). Perspectives on grading systems. *Olympiads in Informatics*, 1, 124–130.

Mareš, M. (2009). Moe – design of a modular grading system. *Olympiads in Informatics*, 3, 60–66.

Merry, B. (2009). Using a Linux security module for contest security. *Olympiads in Informatics*, 3, 67–73.

T. Tochev is a master student in artificial intelligence at Sofia University. Former competitor himself, he helped with the technical organization of many Bulgarian competitions in informatics and some international ones – JBOI 2008, BOI 2008, and IOI 2009.

T. Bogdanov is currently completing his masters in software engineering at Sofia University. Involved with grading systems since the Balkan Olympiad in informatics in 2004, he has helped with grading many of the national high school competitions in informatics and IOI 2009.

[1] `https://svn.openfmi.net/pcms/trunk/test/grader/.`

Olympiads in Informatics, 2010, Vol. 4, 120–133

© 2010 *Institute of Mathematics and Informatics, Vilnius*

The Olympiads in Informatics as a Part of the State Program of School Informatization in Russia

Marina S. TSVETKOVA
Publishing House "BINOM. Knowledge Laboratory"
Proezd Aeroporta 3, 125167 Moscow, Russian Federation
e-mail: tsvetkova@lbz.ru, msvm@lianet.ru

Abstract. Many countries in the world pay much attention to questions of education informatization and support of accessibility of informatics education for students. In this paper it is presented the stages of development of education informatization in Russia and the important results of implementation of appropriate government programs. The following questions have here a special place: perfection of school informatics courses, taking into account new ICT available to schools in the country, computerization of all schools and the connection of schools to the Internet, development of educational media resources, Internet courses for secondary school students, the rise of the role of the Olympiad in Informatics for meeting the requirement of the country in preparation of competent IT-specialists, support of the rights of students in involvement in the Olympiad in Informatics and on the possibility of student preparation in professional-oriented informatics courses in school on a wide scale and the State support of winners and prize-winners of the Olympiad in Informatics.
Key words: informatics education, secondary school education, Olympiads in Informatics, IOI, computer science, informatization of the education system, State educational programs.

1. Introduction

In Russia during the period of 2000–2010, we have seen results of national programs which have led to two main priorities of the informatization of primary and secondary school education system:

- further development of educational programs particularly as applied to the ICT area for students and teachers;
- fcreation of a national collection of electronic educational resources and maintenance of easy access to them of all schools through connection of all schools to the Internet.

The path of developing IT resources in schools has been realized in Russia by the State programs on informatization schools, for the last 10 years, is a path of an education modernization with usage of new informational and communication technologies. The review of programs of informatization which were carried out is given in the Appendix 1 "*Overview of Programs and Projects for Education in Russia"* and in the Appendix 2 "*The Passport of the Federal Program of Education Development 2006–2010"*.

2. Formation of School Course of Informatics and the Olympiad in Informatics in Russia

Formation of school courses in Informatics and the Olympiad in Informatics in Russia is characterized by following stages:

1985–1995. Formation of the subject "Informatics" in Russian schools. Development scientists under the guidance of academician Ershov A.P. of working capability in information science at schools and the first book on informatics by 1985. Creation of interschool centers for instruction in informatics for students. Popularization of developing special details within the program. Creation in leading universities of the Russia faculties of calculus mathematics, cybernetics and applied mathematics. Creation of the All-Soviet Union Olympiad in informatics for schools (1988).

1995–2000. A computerization of schools on the basis of personal computers. Programs of a computerization of schools have been routed on equipment of special schools by ICT-class with Yamaha and PC of a domestic production (to 10 % of schools in the country). Result of implementation of the program – engaging in school of teachers of informatics and development in 1998 of a minimum of the contents of school informatics for schools as a whole. The attention to preparing ICT specialists at the same time has gained in strength. 20 Centers of new information technologies in leading state technical universities have been created. They became centers of the educational Internet network Ru-Net. This fact has affected positively on the Olympiad in Informatics.

2001. The Government of Russia conducted the purchase of 1–3 computers in each of 32 000 country schools. It is the program "*The Computer for country schools!"*. However, thus only about 18 000 teachers from these schools have completed courses on computer skills. The result of implementation of the program is that the computer has become a center of attention for teachers of the most remote schools. Country students who had interest in informatics may now get access to lessons after school together with a coach to learn information science on the computer. It has helped extend the scope of the Olympiad in Informatics to include country schools. However, the computer has not yet become a part of the normal educational resources of such schools, having not entered into the practice of teachers on a regular basis. Systematic programs to train teachers in country schools are however emerging.

2002–2004. The joint project of companies Intel and Microsoft "*Teachers to the future"* has drawn attention to ICT activity teachers (Intel, 2003). "*Teachers to the future*" centers in 8 regions of Russia were created. 10 000 teachers were trained at these centers in one year. Readiness of active teachers for usage ICT in the professional work on a constant basis has emerged. Teachers of informatics began to co-operate with other subject teachers on the basis of ICT. Teachers in general have manifested a readiness to use in their normal duties a computer workstation and additional digital equipment, digital instruments and sensors. Informatics became a part of the olympiad preparations for physicists, chemists and biologists. It has affected the contents of the practical parts of competition for these olympiads.

3. Internet Connecting All Schools in Russia and Internet Olympiad Development

Internet connecting all schools in Russia and internet olympiad development is characterized by following stages:

2001–2005. The Federal program of a development of education was accepted. The State school educational standard (SSES, 2004) was accepted in April, 2004. An important aspect of the standard is that in all school subjects usage of ICT is recommended. The standard of the school Informatics subject contents was developed. This subject has been introduced as federal lessons in primary school (in 3–4 grades), in basic school (in 8–9 grades), and presented in professional oriented high school (in 10–11 grades). Thus, the school syllabus for informatics became continuous with this introduction in all steps of instruction at school. It is important to note, that including ICT into the context of of the general experience for children and into school education promoted conditions for mass development of ICT activity for children: readiness of students to use ICT in general educational activity at each school. It has led to sharp growth of the scope of Olympiad in Informatics. The Olympiad in Informatics have gained wide scope and from 2005 were conducted in all regions of Russia.

2002–2005. The Federal program "*Development of the common educational informational space"* (DCEIS, 2002) introduced a new level of schools informatization in Russia. 30 000 schools gained computer multimedia class-rooms with 15 computers in each, also with a projector and the modem for connection to the Internet. Licenses of Microsoft software for 30 000 schools of the Russian Federation was purchased.

A media CD from 72 topics in various spheres of educational activity of schools was developed and equipped for computer class-rooms and also for children's homes and boarding schools with more restricted functionality. The package of disks for educational assignment is distributed together with computers to school media libraries. 10 000 schools have gained connection to the Internet (basically through the telephone channel), and 5 000 schools have gained a one-sided broadband satellite channel for transmission of streams of educational and cultural-educational information through the Internet. Thanks to engaging of leading universities 17 state educational portals have been developed on all subject domains. The collection of such portals is presented on the state educational portal of Russia (EDU, 2004). The general school education portal (GEP, 2004) was also created (Tsvetkova and Gridina, 2003).

In the area of "*Electronic libraries",* electronic library space presented by links in an educational window has been formed. Internet representations of leading publishing houses of the educational literature in the country have been created.

42 regional centers of distance instruction of teachers (scope – 50% of regions of the Russian Federation) with a dedicated Internet channel have been generated. About 150 000 educators have been trained for gaining common ICT competence.

A major result has been attained – the informational space of schools has been created. The school has become the guarantor of informational preparation of children. The ICT class has become an integral part of a school's resource. Special places at school have ICÆ workstations of teachers configured according to subject orientation, with digital

laboratories. As a result of implementation of the program the Olympiad in Informatics have taken a modern form on the basis of the automated solutions evaluation system. All regional stages of the Russian Olympiad in Informatics (RusOI) began to operate under uniform instructions (the Rules for the Russian secondary students Olympiad of 2003). These are conducted as computer rounds. At a regional stage the use of uniform tasks and tests for their evaluation began, and developed by the Central methodical commission of the RusOI.

2005–2008. In Russia, on the basis of international experience and support of the World Bank the project of National training foundation "*Informatization of education systems"* (NTF, 2005) was realized (Tsvetkova, 2005). Its main objective was discovering and implementation of new educational models of instruction of children in the informational space of school on a systematic basis, particularly by preparation of teachers in the field of ICT. Implementation of network technology for use by the teacher and training teachers became an important part of the project. To enable these tasks a network including 232 region interschool teachers support centers in 7 regions (each of the federal districts) of Russia, namely Khabarovsk, Krasnoyarsk, the Stavropol, Perm, the Chelyabinsk, Kaluga areas and the Republic of Kareliya were created. Centers are intended for continuous instruction and support of teachers in the sphere of using ICT in educational process and providing an easy approach to digital educational resources for all teachers and pupils. All 232 centers have been connected to a dedicated Internet channel equipped by servers.

As a result 250 000 teachers have passed specialized instruction on development of ICT. Mechanisms of approbation of new techniques of instruction on the basis of ICT, including distance methods have been generated. In regions of the project the Internet Olympiad for a school stage of the RusOI have been created. It has allowed the building for the first time model of the scope of the olympiad for all pupils enthused by informatics using networks. For Russia, with difficult geographical conditions and remote schools, the Internet Olympiad is an absolute must.

The major result of the design was the State Internet-collection of digital educational resources (SIC-DER, 2007), easily accessible for all schools of the Russia.

The main results of the project "*Informatization of education systems"* have been the creation in 7 regions of Russia of the common models *of the interschool informational space* as a complete infrastructure, embracing itself for all teachers and all schools of the regions. In this environment the coordinator is the regional Data center. In the project regions of Russia, the conditions enabled system integration of school education into the common informational-educational space of Russia (Tsvetkova *et al.*, 2007). A weak link for other regions of Russia and schools was the absence of the Internet at schools.

4. State Support for School Informatics and for the School Stage of the RusOI

Recently the government of Russia has payed major attention to the support of school informatics and the school stage of the RusOI and as a result of it each secondary student

of Russia has had an opportunity to study informatics at school and to participate in a school stage of the RusOI. Development of such support is characterized by following stages.

2006–2008 and 2009–2010 embrace two stages of the Federal program of education development (FPRO, 2006). Putting new computer classes to country schools, installing school computer workstations for managers and teachers, creation of a Federal system of informational educational resources (FSIOR, 2006) and establishment of Internet collection for all subjects of the State educational standard, development of systems for processing by schools circulation of documents, and informational support for schools (a network of regional educational portals), new mechanisms for economic development of school business, a reinforcement of the social role of the teacher, development of new educational standards for schools, a pedagogical education and higher education program taking into account the Bolonsky agreement, creation of a system of innovative universities and national exploratory universities – are all leading directions of an education modernization in Russia in this program. Upgrading of vocational training has in turn entailed unification of demands to the graduate of school and allocation of equal possibilities for sampling professional opportunities on the basis of the Unified State Examination. In total, certification for students has been a great value of the olympiad. The RusOI became a method for entering universities via budgetary places, without examinations for winners and prize-winners.

The same years, the Priority national project "*Education*" of the Russian Federation Government (PNPE, 2006) began. This project has supplied connection to the Internet 54 000 out of 60 000 Russian schools. On the basis of competition, 10 000 teachers – winners of this competition, and 3 000 innovative schools have been selected. These schools have been equipped with modern ICT and began to be named "Digital school". In the project a special part is allocated support talented youth. All winners and prize-winners of the final stage of the RusOI gain a bonus from the President; depending on the rate of this bonus the student might personally win a modern computer. Using their connections to the Internet, schools and municipal centers in regions of Russia conduct the Internet Olympiad. The portal of the RusOI (RusOlymp, 2007) and a consistently constructed state database of ratings of participants following the results of regional and final stages of the RusOI enables a basis on which winners and prize-winners are defined. On the portal, collections of competition tasks in all school subjects for the years of conducting the RusOI, with analyses of solutions, are presented. All this allows each pupil to prepare for the RusOI even if they are not independently present at the teacher's school.

The Internet has given the opportunity to indicate to students creativity in other ICT spheres. Major growth has helped the development of school sites, school Internet newspapers, social educational networks for parents and children, the Internet – showing the availability of relevant regional services, high-grade distant courses of profile assignment, Internet libraries, museums, collections of teacher's techniques and educational collections in school subjects.

The federal purpose-oriented program of development of education in Russia in 2006–2010 shows new quality – it merges all directions of development of schools, thus informatization is a catalyst for developments of the entire education system. The further

extension of informatization of education is something in which teachers and pupils are showing interest and asking questions. It can now be said that the informatization process has developed to a new phase.. This phase mirrors first of all what schools are asking for Tsvetkova (2009). New requirements of schools are mirrored in the State project *"Our new school. 2020*" (ONS, 2010).

It is necessary to note that by 2010 all school subjects will be supplied with sets of electronic educational resources (FSIOR, 2006). Their delivery in schools will be carried out through the Internet. It will be possible to report on the creation in Russia of a Federal system of informational educational resources opened for all schools. In addition schools will have obtained the license software "First help" and sets of free software.

A special part of the Federal program of development of education in 2006–2010 was dedicated to solving problems of modernization of management of education and financing of schools. *"The complex project modernization of education"* (CPME, 2006) addressing these problems has been implemented. The outcome of this project was a new system of financing of schools which takes into account the numbers of pupils in schools, and the obligatory educational services. This guaranteed a set of the state educational services and participation in the RusOI for each pupil.

This has all influenced a reinforcement of the attention of directors of schools to the subject "Informatics" and the Olympiad in Informatics at schools. It is important, that the first schools (open) stage of the RusOI be conducted in each school. Then any talented child remains under attention as should happen according to the law on education of Russia. It is now in place in new regulations for the school olympiad in Russia 2009.

All these programs for 10 years of education infomatization in Russia have allowed for the generation of uniform informational educational space for schools in Russia. In any school, the teacher and the pupil become participants of this space. The big investment in development of this informational educational space is carried out by building the regional programs of informatization and a development of education generally. To use and develop these programs it is recommended to use a series of books "*Informatization of Education"* by Publishing houses "BINOM. Knowledge Laboratory" (BKL, 2006). The Publishing house BKL releases also a series of books on Olympiad Informatics and has website to support of the teachers in this area.

5. Conclusion

In this article a long-term experience of development of education informatization in Russia and the important results of implementation of appropriate government programs have been presented. The author hopes that describing the systematization of the programs of education infomatization in Russia will be useful for colleagues from other countries including members of the IOI society. Results of development of the information educational environment in regions of Russia also deserve attention and it is recommended to familiarize with corresponding materials regional educational portals (REP, 2009).

References

BKL (2006). Publishing houses “BINOM. Knowledge Laboratory” (in Russian). http://www.lbz.ru, http://www.metodist.lbz.ru.
Bryansk educational portal. http://www.edu-bryansk.ru.
CPME (2006). The complex project modernization of the Russian education (in Russian). http://www.kpmo.ru/kpmo.
DCEIS (2002). The Federal program “Development of the common educational informational space” (in Russian). http://elementy.ru/Library9/Progr630.htm.
Design ”Education system Information” in Khabarovsk territory. http://rkc.ippk.ru.
EDU (2004). State educational portal of Russia (in Russian) http://www.edu.ru.
Educational informational portal of Hunts-Mansijskogo autonomous region. http://www.eduhmao.ru.
Educational portal of a city of Murmansk. http://www.edu-murmansk.ru.
Educational portal of Krasnoyark. http://www.cross-edu.ru.
Educational portal of the Pskov area. http://www.pskovedu.ru.
Educational portal of the Rostov area. http://portal.rsu.ru.
Educational portal of Tatarstan Republic.http://edu.ksu.ru
Education system Information in Kareliya Republic. http://iso.karelia.ru.
Education system Information in the Perm edge. http://imc.ocpi.ru.
Education system of Stavropol Territory. http://www.stavedu.ru.
FPRO (2006). The Federal program of a development of education (in Russian). http://www.fcpro.ru.
FSIOR (2006). Federal system of informational educational resources (in Russian). http://fcior.edu.ru.
GEP (2004). The general school education portal (in Russian) http://www.school.edu.ru.
Informational educational portal of the Ulyanovsk area. http://www.sinncom.ru.
Informational portal about educational possibilities in Pribaikalye. http://www.baikalnarobraz.ru.
Intel (2003). The join project of the companies Intel and Microsoft “*Teachers to the future*” (in Russian). http://www.intel.com/cd/corporate/education/emea/rus/elem_sec/programs/teach.
Khabarovsk edge informational educational network. http://abc.edu-net.khb.ru.
Kostroma an educational portal. http://www.kostroma.edu.ru.
Lipetsk regional educational portal. http://deptno.lipetsk.ru/eduportal.
Moscow education: an informational portal of Department of education of Moscow. http://www.mosedu.ru/.
Novgorod educational portal. http://edu.novgorod.ru.
Novosibirsk open educational network. http://www.websib.ra.
NTF (2005). The project of National training foundation “Informatization of education systems” (in Russian). http://portal.ntf.ru/portal/page/portal/iso/about/iso2008.
Omsk educational server. http://www.omsk.edu.ru.
ONS (2010). State project “Our new school. 2020” (in Russian). http://www.educom.ru/ru/nasha_novaya_shkola/.
Open Sochi education: an informational portal. http://www.sochi.edu.ru.
Perm regional educational portal. http://eduport.perm.ru.
Perm urban school portal. http://www.schools.perm.ru.
PNPE (2006). The Priority national project ”Education” of the Russian Federation Government (in Russian). http://mon.gov.ru/pro/pnpo/int.
Portal ”Science and education of the Udmurt Republic”. http://www.udmedu.ru.
Regional informational-educational portal of the Ministry of Education of the Saratov area. http://edu.seun.ru.
Regional portal of educational community of Orenburzhye. http://www.orenport.ru.
REP (2009). The Portal of education, a science and the youth policy of the Kabardino-Balkaria Republic. http://www.edukbr.ru.
REW (2006). Web-site “Russian educational window” (in Russian) http://window.edu.ru.

RusOlymp (2007). The portal of all school Olympiad of Russia on 20 subjects (in Russian). http://rusolymp.ru.

School portal of the Kaliningrad area. http://www.school.baltinform.ru.

Science and education of the Samara area: a regional portal. http://samara.edu.ru.

SIC-DER (2007). State Internet-collection of digital educational resources (in Russian). http://www.school-collection.edu.ru.

SSES (2004). State school educational standard (in Russian). http://www.ed.gov.ru/ob-edu/noc/rub/standart.

Tsvetkova, M., Gridina, E. (2003).Рубикатор общего щбразования и уго место в среде навигации портала общего образования для различных групп пользователей. Сборник научных статей "Интернет-порталы: содержание и технологии", Выпуск 1 (Internet-portals: content and technology, Vol. 1, 498–510) Prosveschenie, Moscow (in Russian). http://www.edu.ru/db/portal/e-library/00000017/00000017.htm.

Tsvetkova, M. *et al.* (2005). Межшкольные методические центры. Организация работы (Inter-school methodical centers. Organizational activity). World Bank, National training foundation, Moscow (in Russian). http://iso.ntf.ru/DswMedia/1_mmc_organizaniiyarabotyi.pdf.

Tsvetkova, M., Dyilyan, G., Ratobyilskaya, E. (2007). Модели комплнксной информатизации общего образования. Серия: "Информатизация образования" (Complex models of school education informatization. Series: "Informatization of education"). LBZ (BINOM. Knowledge Laboratory), Moscow (in Russian). http://www.lbz.ru/katalog/products/literatura-dlja-shkol/informatizacija-obrazovanija/modeli-kompleksnoj-informatizacii-obschego-obrazovanija.

Tsvetkova, M. (2009). Модели непрерывного информатизованного образования. Серия: "Информатизация образования" (Models of continual information education. Series: "Informatization of education"). LBZ (BINOM. Knowledge Laboratory), Moscow (in Russian). http://www.lbz.ru/katalog/products/literatura-dlja-shkol/informatizacija-obrazovanija/modeli-nepreryvnogo-informacionnogo-obrazovanija.

Vladimir educational portal. http://www.edu.wladimir.ru.

Yaroslavl center of telecommunications and intelligence systems in education. http://www.edu.yar.ru.

Appendix 1

Overview of Programs and Projects for Education in Russia

The name of the program/project	Purposes	Main results	Financial sources
1990–2000			
Programs of computerization of schools.	Computerization in education	Equipment of basic schools by classes with Yamaha and small computers of domestic production (up to 10% of schools in the country).	Federal budget
The program of informatization for higher education		20 Centers of new information technologies on base of the most advanced high schools of the country are created. They became units of first educational Internet-network Ru-Net	Regional budget
1992–2002			
Soros project	Creation of network educational community	High schools and Soros's schools are created, community of Soros teachers is generated	Investments of Soros fund

The name of the program/project	Purposes	Main results	Financial sources
2001–2010 The federal program of development of education Since 2005 this program is incorporated with the program of informatization of education (see item 4)	Modernization of structure and the contents of the Russian education	The State educational standard (it is accepted in April, 2004) in which in all disciplines using ICT is taken into account The contents of subject "Computer science and ICT" is updated. This subject is entered as obligatory into the basic step training (8–9 classes), and submitted in the senior step (10–11 classes) as profile with alternative (elective) courses. The Unified State Examination (since 2002 for 2007 as experiment)	Federal and regional budgets
2001–2005 The presidential program "Children of Russia"	Computerization of rural schools Support of children's creativity	In 2001 it is purchased 2–3 computers in each of 32 000 rural schools. 18 000 rural teachers have passed courses of the computer literacy	Federal budget
2001–2005 Target program of Ministry of Education and the Government on support of talented youth. Since 2006 the National project for education, a direction "Support of capable and talented youth" is added		Financing of realization of 20 All-Russia Olympiads for secondary school students in subjects of the curriculum. Financing of participation of Russian teams in International Olympiads (mathematics, informatics, physics, chemistry, biology, geography)	Federal budget The budget of the national project on the premium for instructors and talented youth
2002–2005 The Federal target program. "Development of the uniform educational information environment"	Set of the purposes on all levels of informatization of education – school, vocational training, additional education	30 000 comprehensive schools and 3500 establishments of initial vocational training have received a computer class (1–2) with 15 computers in each. The license of Microsoft products for all schools of the Russian Federation is purchased. The complete set of disks for educational purpose is supplied together with computers (27 names). The media library with 72 names on various directions of educational activity free-of-charge for all schools and technical training college is developed 17 educational portals on all subject domains are developed, which collection is submitted on the state educational portal. The portal of national library is generated. The state portal of open education is generated. 120 000 educational workers has passed training on ICT competence.	Federal and regional budgets in equal shares

The name of the program/project	Purposes	Main results	Financial sources
		42 new regional centers of information technologies in education (50% of regions of the Russian Federation) with allocated Internet channel are generated. 10 000 schools have received connection to the Internet (basically through the telephone channel), 5 000 schools have received unilateral broadband channel of the satellite Internet for transfer to territories the educational and cultural-educational information. 18 specialized ICT centers are generated on the basis of high schools of the country. The network of the federal centers on all directions of educational information is generated. All children's homes and boarding schools for children with the limited opportunities are equipped with computer classes and media libraries.	
2002–2004 Joint project of Ministry of Education and the Microsoft and Intel companies "Training for the future"	Training of teachers of various subjects to new educational technologies on the basis of the international experience	The centers "Training for the future" are created in 8 regions of the country with a class with 25 computers and an access to the Internet. 10 000 school teachers have passed training in these centers	The regional budget and investments of the Intel and Microsoft companies
2005–2008 The project "Informatization of the educational system"	To generate and approve model of the system approach to informatization of schools of the country	The network of the regional interschool methodical centers in 7 regions of the country is created. The centers are intended for continuous training and support of teachers in ICT and granting's of free and equal access to educational services of all pupils through methodical and maintenance service by the centers directly on places of residing. All 232 centers are connected to the allocated Internet channel and equipped with servers. The network is offered as model for distribution in the country. The national Internet collection of electronic educational resources (servers and service – State Institute of Information Technologies and Telecommunications) with an easy access for all schools of the country is being created.	The federal budget (means of the loan of IBRD) and not less than 50% of in addition cumulative regional and municipal budget

The name of the program/project	Purposes	Main results	Financial sources
		250 000 teachers will pass training on ICT competence. The model of remote profile training of pupils of the senior school is generated on the basis of the regional centers of information. The mechanism of support of creative teacher's competitions is generated on the basis of network educational projects. 2% of creative teachers will be selected and maintained by grants for distribution of network initiatives to all schools of pilot regions	
In two stages **2006–2008** **2009–2010** The federal target program of development of education	Full informatization of all educational establishments of the country	Connection to the Internet of 54 000 schools of the country. Additional equipping of rural schools with new computers. Additional equipping of schools with workplaces for managers and teachers. Filling of the national Internet collection with electronic educational resources and curriculums on all subjects of the State educational standard. Regular training of teachers in the volumes allowing within five years all teachers of the country to raise their ICT qualification. Development of standard of ICT requirements and ICT certificate for educational workers. Modernization of vocational training. Creation of the new educational standard for establishments of initial and average vocational training. "Informatics and ICT" is entered as the basic subject in size of 78 hours of training	Federal budget Regional budget Investments
2006–2010 The Priority national project "*Education*" of the Russian Federation Government **2010–2020** The State project *"Our new school. 2020"*	Introduction in all schools of the country of new educational technologies on the basis of the Internet resources	Connection to the Internet of 54 000 schools of the country. Formation for all schools of system of safe use of the Internet. Formation of pedagogical community of creative teachers (the President premium) and innovative schools (the President premium). Support of capable and talented youth. Entering in the Bolonsky agreement process	Federal and regional budgets

Appendix 2

The Passport of the Federal Program of Education Development 2006–2010

The name of the Program	–	The Federal target program of development of education, 2006–2010
Date of decision-making on development of the Program	–	The order of the Government of the Russian Federation No. 1340-r, 3 September, 2005
The state customers of the Program	–	Federal agency on education, Federal agency on science and innovations
The state customer – co-ordinator of the Program	–	The Ministry of Education and Science of the Russian Federation
The basic developer of the Program	–	Federal Agency on Education
The purposes and tasks of the Program	–	The basic strategic purpose of the Program is the maintenance of conditions for satisfaction of needs of citizens, societies and a labor market in education of high quality by creation of new institutional regulation mechanisms in the educational sphere, updating of structure and the contents of education, development of fundamentality and a practical orientation of educational programs, formations of system of continuous education Strategic tasks of the Program are: • perfection of the contents and technologies of education; • development of quality maintenance system for educational services; • increase of a management efficiency in educational system; • perfection of economic mechanisms in educational sphere
The major target indicators and parameters of the Program	–	Densities of number of children of the senior preschool age, trainees in system preschool* educations in alternative forms Densities of pupils of 9–11 classes training under programs of professional-oriented preparation, individual curricula and programs of professional-oriented training Amount of the educational establishments realizing new State educational standards of the general education, including requirements to a level of preparation of graduates of various steps of the general education and a condition of realization of educational activity Densities of number of the pupils training in system of interschool additional education Densities of number of the occupied population which passed improvement of professional skill and professional retraining Densities of number of graduates from educational establishments of vocational training (including enlisted in Armed forces of the Russian Federation), employed within 1 year Densities of number of graduates from establishments of the vocational training which has mastered educational program by use of remote training Densities of number of establishments of the vocational training having access of local educational networks to global information resources Share of the foreign students training in the Russian establishments of vocational training on a commercial basis. Densities of number of the Russian higher educational institutions accredited by foreign accreditation agencies

		Densities of number of youth from the low-income families, living in the rural areas, entered in higher educational institutions Increase of the rating of Russia by results of the international inspections of quality of education (PISA, etc.) Densities of number of graduates from the educational establishments employed within 1 year on the received specialty, from an aggregate number of graduates Growth of number of trainees in frameworks of integrated establishments of the general education Growth of number of trainees within the framework of the integrated establishments of vocational training Growth of total amount of the research works executed in all-nation universities and system-forming educational institutions of the higher vocational training Share of the incomes received from enterprise and other commercial activity in the consolidated budget of educational sphere Growth of finances involved in educational sphere Increase in densities of number of the pupils who are taking training under programs with use of the network approach Growth of number of the automated workplaces intended for management personal in educational sphere
Terms and stages of realization of the Program	–	2006–2010. The first stage (2006–2007) includes the works connected with creation of models of educational development in separate regions, model approbation, and also with the beginning of scale transformations and experiments. At the second stage (2008–2009) the priority is given to the actions directed on purchase of the equipment, the investment (modernization of a material educational infrastructure and others high cost works), realization of methodical, personnel and information supplies of the Program At the third stage (2010) the actions are realized directed basically on introduction and distribution of results, received in the previous stages
Sizes and sources of financing of the Program	–	The total amount of financing of the Program in the prices of corresponding years makes 61952.35 million rubles, including: • due to means of the federal budget – 45335.02 million rubles; • due to means of budgets of subjects of the Russian Federation – 12501.74 million rubles; • due to inappropriate sources – 4115.58 million rubles
Expected end results of realization of the Program and parameters of social and economic efficiency	–	New standards of the general education will be developed and introduced for 60 percent of educational disciplines. The amount of programs of the vocational training, which has received the international recognition, will increase in 1.3 times in comparison with 2005 The share of the pupils receiving education with use of information technologies, will increase in 1.5 times in comparison with 2005. Changes in system of additional education of adults will allow to train in 1.3 times of more citizens in the age of 25–65 years in comparison with 2005 Increase of a rating of Russia in the international inspections of quality of education up to a level being average (20 place) for the countries which are included in the Organization of Economic Cooperation and Development (now Russia occupies 30 place) is predicted The share of foreign pupils in system of average and higher vocational training, including trainees on a commercial basis, will increase from 0.9 up to 1.6 percent

The share of the pupils who have entered educational institutions of average and higher vocational training by results of uniform graduation examination, will increase from 40 up to 90 percent

The share of the educational institutions realizing the programs of two-level vocational training, will increase from 15 up to 70 percent

The share of the finances received from the commercial and noncommercial organizations for financing of education, in a total sum of educational costs will increase in comparison with 2005

Alignment of access to reception of quality education will be provided by means of:

- distribution of various models of education for children of the senior preschool age with the purpose of maintenance of equal starting opportunities for the subsequent training in an elementary school, professional-oriented training;
- creation of the all-Russian system of an estimation of quality of education and system of continuous vocational training;
- advanced development of national universities and system-forming high schools as integration centers of science and education for preparation of highly professional staff

M.S. Tsvetkova, PhD in pedagogic science, associate professor of Academy of Improvement of Professional Skill of Educationalists, prize-winner of competition "The Teacher of Year of Moscow" (1998), main expert of state projects of school education informatization in the Ministry of Education of the Russian Federation (2001–2005), the expert of the World bank project "Informatization of Education System". Since 2002 Marina is a member of the Central Methodical Commission of the Russian Olympiad in Informatics, the pedagogic coach of the Russian Team on the IOI.

Olympiads in Informatics, 2010, Vol. 4, 134–141

© 2010 *Institute of Mathematics and Informatics, Vilnius*

An Enticing Environment for Programming

Tom VERHOEFF

Department of Mathematics and Computing Science, Eindhoven University of Technology
Den Dolech 2, 5612 AZ Eindhoven, The Netherlands
e-mail: t.verhoeff@tue.nl

Abstract. While teaching a course on the foundations of informatics to non-CS students, the author wanted to offer a programming challenge without burdening the participants with the numerous details that typically accompany the use of practical programming languages and tools. In particular, there should be no need to install an editor and execution environment (compiler or interpreter). Furthermore, the programming language should be sufficiently simple and clean. However, the author did not want to design a completely new language with tools.

This article presents *Tom's JavaScript Machine* as an attempt at providing a simple and enticing environment for programming, and reports some experiences. *Tom's JavaScript Machine* is freely available on-line and only requires a web browser that supports JavaScript. It includes a simple 3D-variant of Turtle Graphics (for browsers that support the HTML5 canvas element) and an instructive programming challenge with extensive (inter)active hints.

The ideas behind *Tom's JavaScript Machine* can also be applied to create problem-specific environments for informatics contests. However, the current implementation still has some shortcomings that need to be addressed.

Key words: programming tools, JavaScript, self-reproducing programs, study material development.

1. Introduction

In Fall 2009, I taught an Honors Class on the foundations of informatics as a science (Verhoeff, 2009). The participants were selected second-year students from various disciplines, excluding computer science, at Eindhoven University of Technology. We used the book *Algorithmic Adventures* by Hromkovic (2009). This book briefly describes the birth of informatics as a science, focusing on the notion of an algorithm as an object of scientific study. It then presents the exciting things we have learned about algorithms, in particular, the limits of algorithmic computability, our struggles with efficiency and algorithmic complexity, the surprising powers of randomness and approximation, how algorithms changed the world of cryptography to accomplish the unbelievable, and DNA computing and quantum computing as radically different approaches to do computations.

The course was specifically not about programming. Nevertheless, it is useful to do some programming to get a better feel for algorithms. With informal descriptions of algorithms it is too easy to trick oneself into believing that something 'works'. This is especially the case for the following challenge, which I found very instructive when I first encountered it myself.

> **Challenge:** Write a *self-reproducing* program that processes no input and that generates its own listing as output.

If you have never attempted this yourself, then I encourage you to give it a try. It is not easy, requires perseverance, and will teach you about the link between biology and computer science. One needs to have various creative insights to tackle this challenge.

Unfortunately, if you want students without any background in programming to work on a challenge like this, then you need to introduce them to some practical programming environment. Without a well-defined language, it will not be clear whether the problem was really solved. Typically, they would need to install some tools, like a program editor and a compiler or interpreter. Then they would need to learn the syntax and semantics of the programming language, and how to operate the tools. This poses quite a big threshold.

I considered various options. Python (2010) came closest to being minimally obtrusive. However, it still did not match my ideal of zero install and immediate interaction. Then, it struck me that JavaScript run from a web browser could be considered as well.

2. Tom's JavaScript Machine

JavaScript is a fairly clean and simple programming language, standardized under the name ECMAScript since the late 1990s (ECMA, 2010). It has features from both functional and object-oriented programming. An in-depth treatment can be found in Flanagan (2006). JavaScript may not have a good name among some groups, but by certain measures it is the most-used programming language of this day.

In no time, I was able to put together a web page with three text areas and a button (see Fig. 1). In one text area, the user enters some input, in another one the user can edit a JavaScript program text, and output is shown in the third text area. When the user clicks the *Run* button, an embedded script (itself also written in JavaScript) evaluates the string `s` in the program area as JavaScript program, through the standard JavaScript function `eval (s)`. I added some minimal facilities for user input and output, because JavaScript by itself does not offer that. The result is *Tom's JavaScript Machine* (Verhoeff, 2009b).

Nice things about *Tom's JavaScript Machine* are that

- it is *zero install*, providing that you have a computer with a web browser supporting JavaScript (version 1.5 or higher; this is available in all modern browsers);
- it offers *immediate interaction*: just type in your JavaScript program text, some input, and click *Run*. Because input is not 'consumed', you do not need to retype it when you want to run your program again.

The following program, which adds a sequence of input numbers, illustrates the facilities for input and output.

Fig. 1. The main user interface of *Tom's JavaScript Machine* consists of three text areas and a *Run* button

```
1  var sum = 0;
2  while(moreInputs()){
3    sum = sum + readNum();
4  }
5  writeln(sum);
```

Once the initial version was created, some further wishes naturally arose and were easily added, including the following.

- A summary of JavaScript basics.
- Example programs that can be loaded into the machine with one click.
- A user-selectable separator to split input (default: a space).
- A *Challenge* button that clears the input area, executes the program, and compares the output to the program text.

The result is a surprisingly usable programming environment (which I even use myself in some situations). Section 4 discusses some limitations.

3. Facilities for Developing Study Material

While developing a series of hints for the challenge of writing a self-reproducing program, it occurred to me that some further facilities would be useful for teachers. Writing study material for programming courses is inherently a cumbersome task. On one hand, there is the text to be written, e.g., using LaTeX. On the other hand, there are programs and program fragments to be included in the text, possibly with some input and corresponding output. It requires good discipline and preferably some good tools to maintain consistency of all the material, while the text, programs, and inputs evolve. In the traditional approach, whenever a program or input changes, the program must be run again to produce up-to-date output, and all of this must be incorporated (possibly via inclusion) in the text.

In the case of JavaScript programs, it is convenient to write the study material in HTML with embedded scripts, in JavaScript. I extended *Tom's JavaScript Machine* with special facilities for writing study material:

`_parseURI()` to open the web page with the machine and initialize its input area, program area, and output area, and some other parameters with values taken from its URI (web address) in the form

```
.../machine.html?_program=...;_input=...
```

`_machine_link()` to generate an embedded hyperlink with given values for various machine parameters, i.e., in the preceding form;

`_output_of()` to return as string the output of a program given as string, (optionally, input and separator can be passed as well);

`_inject()` to inject a given text with given background color (yellow for input, green for program, and blue for output).

For instance, the following piece of HTML code generates a 'program box' with green background showing the program `writeln(1+1);writeln(1+1);`, an 'output box' with blue background showing the output `22`, and below it a link to the machine for loading this program.

```
<script type="text/javascript">
var _prog = "writeln(1+1);\n";
_inject(_prog, 'programbox');
_inject(_output_of(_prog), 'outputbox');
_machine_link('Load in the machine', _prog);
</script>
```

For more details, see the *About* link in *Tom's JavaScript Machine*. Using these facilities, it is easy to write study material that incorporates programs with input and corresponding output. The hints for the challenge also involve programs that take programs as input and/or that generate programs as output. All of this is neatly handled by these facilities.

4. Experiences and Further Wishes

Tom's JavaScript Machine was used by six participants of the Honors Class to try their hand at the challenge of writing a self-reproducing program. Only one of the students had some prior experience with C, and another with PHP. All of them were able to use the machine immediately to experiment and start on the challenge. And all of them were able to solve the challenge up to a certain level, guided by the hints.

It should, however, be noted that the JavaScript programming language has its own quirks that do get in the way. For the challenge, this turned out to be in the area of strings.

In particular, traversing the characters of a string and constructing expressions to yield specific strings are somewhat awkward. This distracts from the more abstract aspects of the challenge. It is clear that the students found this annoying and that it reduced their interest and limited their progress.

Nevertheless, I consider *Tom's JavaScript Machine* a success, because it demonstrates a new way to offer a low-threshold programming environment and a new way to develop and deliver accompanying study material. In fact, for some computational tasks, I now use the machine myself.

There are some obvious shortcomings to the current implementation:

- the programming language is JavaScript (fully, and only); although JavaScript is nicer than often believed, it is not 'beginners proof';
- the edit facilities in the text areas for input and program are rather limited; in particular, there is no line numbering, no syntax highlighting, and no code completion;
- there is no facility to save and load input and program texts (though the available mechanism to clone the machine in its current state mitigates this somewhat);
- the feedback on syntax and runtime errors is limited and browser dependent;
- there is no facility for interleaved input and output, other than using a standard JavaScript function like `prompt(s)`.

For specific problem domains, it is possible to develop specialized versions of the machine. I created an experimental version of the machine for *3D turtle graphics* (Verhoeff, 2010). The implementation is based on the HTML5 canvas element. Note that HTML5 is not yet an official W3C standard, and that not all major browsers support its current definition. Fig. 2 shows a 3D turtle graphics program and its output on input `5 25 2`, which is a pentagram tilted over 60°. The turtle looks like an aircraft.

```
1 var t = new TurtleGraphics.Turtle();
2  var N = readNum(); // number of corners
3  var k = readNum(); // step size
4 t.Roll(-60);
5 for var i = 0; i != N; ++i) {
6   t.Move(5);
7   t.Turn(k * 360 / N);
8 }
9 t.DrawTurtle('blue');
```

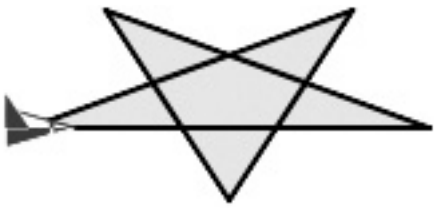


Fig. 2. 3D turtle graphics program and its output for input `5 25 2`.

Similarly, one can imagine having specialized versions of the machine to deal with user interface design, where the programmer gets access to some input fields and buttons. Or a specialized version for manipulating the *Document Object Model* (DOM) that underlies the processing of web pages in browsers. The following program 'hacks' the title of the machine's web page:

```
document.title = "This page was hacked :-)";
```

A version specialized for exploring *numerical algorithms* also comes to mind. Note that JavaScript supports the double-precision 64-bit format conforming to IEEE Standard 754 (IEEE, 1985). Thus, the examples from Horvath and Verhoeff (2003) can be tried in *Tom's JavaScript Machine*. A merger with the floating-point calculators of Vickery (2010) would be interesting in this context.

Finally, in the same vein as the challenge of writing a self-reproducing program, it is imaginable that problem-specific specializations of (an environment like) *Tom's JavaScript Machine* are offered in an informatics contest.

5. Discussion

It should be noted that *Tom's JavaScript Machine* has not been used extensively. My initial motivation for developing it was solely to provide, to non-CS students, an easy environment for the challenge of writing a self-reproducing program. The Honors Class *Algorithmic Adventures* involved no other practical programming, though I am tempted to change that next year. The students learned about programming by doing it on one particular problem. I do not expect that this will work for everyone.

The main reason for presenting it here is that *Tom's JavaScript Machine* offers an environment for programming that differs from more traditional environments. In particular, it is immediately available, it is very easy to use especially with small programs, and it is easy for teachers to develop study material.

As noted in Section 4, the current implementation does have some shortcomings, but the directness of the environment compensates for that. I hope that others will pick up these ideas, develop them further, and investigate how such environments can be put to good use in teaching.

6. Conclusion

Tom's JavaScript Machine offers an enticing environment for programming, with a low threshold. It is zero install, only requiring a modern web browser. There is even a version that you can download, so that you can use it locally without internet access. The interface is very simple, enabling users to start programming immediately.

Additional features of the machine make it easy to develop, in HTML, educational material that incorporates programs with sample input and output, and that the user can load into the machine with a single click. These programs are embedded in the study

material and they are executed as the page is loaded. Program errors are reported immediately. Therefore, the consistency between text, programs, input, and output is easy to maintain. It is also straightforward to feed the output of one program as input into another program, or to use the output of a program as a new program to process some input. This is illustrated in the 40 pages with hints for the challenge of writing a self-reproducing program.

An experimental version of the machine offers 3D turtle graphics based on the HTML5 canvas element. In a similar vein, other special versions of the machine can be constructed. For instance, one can offer a programming environment in the area of user interface design, involving various input fields and buttons, or an environment that involves the internal structure of web pages, based on the Document Object Model (DOM), etc. The use of environments like *Tom's JavaScript Machine* in informatics contests could be interesting as well, and needs further development and investigation.

There are some obvious shortcomings to the current implementation, but they are not show stoppers. The current version demonstrates that this approach is promising, and I hope that it will be explored by others.

References

ECMA International. Industry association dedicated to the standardization of information and communication systems, including ECMAScript.
URL: `www.ecma-international.org` (accessed April 2010).

Flanagan, D. (2006). *JavaScript: The Definitive Guide*. Fifth Edition. O'Reilly.

Horvath, G. and T. Verhoeff (2003). Numerical difficulties in pre-university informatics education and competitions. *Informatics in Education*, 2(1), 21–38.

Hromkovic, J. (2009). *Algorithmic Adventures: From Knowledge to Magic*. Springer.

IEEE (1985). *ANSI/IEEE Standard 754-1985 for Binary Floating-Point Arithmetic*.

Python Programming Language.
URL: `www.python.org` (accessed April 2010).

Verhoeff, T. *Honors Class Informatics*. Course web page 2009.
URL: `www.win.tue.nl/~wstomv/edu/hci` (accessed April 2010)

Verhoeff, T. (2009). *Tom's JavaScript Machine*.
URL: `www.win.tue.nl/~wstomv/edu/javascript` (accessed April 2010)

Verhoeff, T. (2010). 3D turtle geometry: Artwork, theory, program equivalence and symmetry. *J. of Arts and Technology*, 3(2/3), 288–319.

Vickery, C. *Interactive IEEE-754 Floating-Point Calculators and Reference Material*.
URL: `www.purl.oclc.org/vickery/IEEE-754/` (accessed April 2010).

T. Verhoeff is an assistant professor in computer science at Eindhoven University of Technology, where he works in the Ggroup Software Engineering & Technology. His research interests are support tools for verified software development and model driven engineering. He received the IOI Distinguished Service Award at IOI 2007 in Zagreb, Croatia, in particular for his role in setting up and maintaining a web archive of IOI-related material and facilities for communication in the IOI community, and in establishing, developing, chairing, and contributing to the IOI Scientific Committee from 1999 until 2007.

Olympiads in Informatics, 2010, Vol. 4, 142–150

© 2010 *Institute of Mathematics and Informatics, Vilnius*

Selection Mechanism and Task Creation of Chinese National Olympiad in Informatics

Hong WANG
Department of Computer Science and Technology, Tsinghua University
100084 Beijing, China
e-mail: wanghong@tsinghua.edu.cn

Baolin YIN
School of Computer Science and Technology, Beihang University
100083 Beijing, China
e-mail: yin@nlsde.buaa.edu.cn

Rujia LIU
Eryiju Science and Technology Company Limited
100084 Beijing, China
e-mail: rujia.liu@gmail.com

Wenbin TANG, Weidong HU
Department of Computer Science and Technology, Tsinghua University
100084 Beijing, China
e-mail: tangwb06@gmail.com, wd.h@163.com

Abstract. In this article, we present a general overview and associated issues of the selection mechanism for the IOI China team of the Chinese National Olympiad in Informatics (CNOI), and some approaches and management experience to assure task quality during the past years. First, the existing contests and activities of CNOI are introduced. Then we focus on multi-test selection and multi-aspect assessment for top ranked contestants, building up of task creation team, task innovation and improvement, etc. A task example from the IOI China Team Selection Competition is also given in the appendix.

Key words: selection mechanism, contestant assessment, task creation, task innovation.

1. The Existing Contests and Activities of the Chinese National Olympiad in Informatics

The NOI Scientific Committee (NOISC) and NOI Competition Committee (NOICC), under the guidance of the China Computer Federation (CCF), are responsible for the technical organization and management of CNOI. There are several related contests and activities every year in China (Wang, 2007). The major events are given as follows, in increasing order of task difficulty:

(1) *National Olympiad in Informatics in Province*

National Olympiad in Informatics in Province (NOIP) is held in the middle of October and November each year. Contestants are divided into two levels: junior and senior, which two rounds for each level.

Preliminary round: a conventional paper-based test consisting of multiple-choice questions and short-answer questions. It‘s an elementary contest aiming to test the basic knowledge and skills of the contestants. On average, over 80,000 contestants participated each year in this round from NOIP’2006 to NOIP’2009. After this round, the top 15% contestants advanced to the final round.

Final round: a computer-based programming contest. Contestants have 3 hours to solve 4 tasks which have the same format as IOI tasks, but are more elementary.

(2) *National Olympiad in Informatics*

NOI test includes two competition days (5 hours for 3 tasks on each day) and one week activity similar to the IOI. NOI is the highest level national-wide contest of China. The contestants of NOI are usually winners of province-wide competitions. A team competition was introduced in NOI’2006. NOI sets gold (10%), silver (20%) and bronze (30%) medals.

(3) *NOI Winter Camp Competition*

NOI Winter Camp Competition (NOIWCC) contains an intense training course of one week during January. Students are required to take a five-hour competition on the last day – that‘s the NOIWCC. The tasks of NOIWCC are more difficult than those of NOI competition. The scores of NOIWCC are also considered in selecting the IOI China team (see below).

(4) *The Final China Team Selection Competition*

The final China Team Selection Competition (CTSC) is the most important competition for the IOI China team selection, which is always held in early May every year. The format is similar to the IOI (two days, 5 hours for 3 tasks each day). Its size and influence is not as big as NOI, but the tasks of CTSC are arguably the most difficult among all CNOI contests.

2. Selection Mechanism for the Contestants of the IOI China Team

After practicing and exploring more than 20 years, we have established a strict mechanism with a set of rules, for selecting talented students and building the IOI China team. The mechanism and rules are based on different investigations and comprehensive surveys, including multiple contests and paper defence. It guarantees that the selected contestants have outstanding programming skills, psychological quality and comprehensive ability.

2.1. *Multi-Test Selection*

Obviously, there could be some contingency for an individual contest score because of task style and other reasons. It's necessary to arrange multiple contests in order to reflect the general ability and actual quality of candidates. Our IOI team selection process can be divided into four stages:

(1) The top 20 contestants from the last NOI contest form National Training Team (NTT) for the IOI of next year (candidates for the IOI China team).
Take the selection of the IOI'2010 China team as an example. The process began in NOI'2009 during July–August of 2009. The top 20 contestants of NOI'2009 became the members of the NTT for IOI'2010. In the next three stages, we will track the performance of the NTT members and finally build the China Team from them.
(2) All the NTT members must participate in the NOI Winter Campus training and final competition-NOIWCC. The competition score of each individual NTT member contributes to his total score for IOI team selection. At present, NOI WCC has a weight score of 25 (out of 110, see below).
(3) The final and most important selection competition for IOI team – CTSC. All the NTT members must participate in CTSC. This contest has a total weight score of 60 (30 for each day).
(4) A few hours after CTSC, we accumulate the total score for each NTT member. Only top 6 students have the opportunity to participate in the final oral defence, from which we select the best four students to form the IOI China team.

2.2. *Multi-Aspect Assessment*

Besides contest scores, we also make comprehensive assessments related to oral expression, psychological quality and English proficiency of NTT members. A personal statement and letter of commitment is requested from each individual member. The main assessments include:

- During nearly one year between last the NOI and next CTSC, every the NTT member is required to design their own contest problems as part of their obligatory homework. At the meanwhile, there is some necessary discussion and communication.
- During the NOI Winter Camp training and contest, each NTT member must participate in an oral paper presentation and defence (10 min + 5 min). The judgement from the jury has a weight score of 15.
- In the final oral defence for the top 6 contestants, each student is expected to introduce himself in English and answer questions in front of the jury and hundreds of spectators.
- Each contestant is required to provide a personal statement and letter of commitment.

Table 1
Selection process and corresponding weight scores

Items	Weight score	Remarks
Homework	10	
NOI Winter Camp Competition (NOIWCC)	25	
Paper presentation and oral defence	15	15 minutes/contestant
China Team Selection Competition (CTSC)	60	2 days, 30 for each
Total score	110	

2.3. *Simulation Training before IOI*

After selecting 4 contestants, we also arrange a training competition between IOI contestants and ACM/ICPC contestants. The problems are usually adopted from previous ACM/ICPC competitions. Each IOI contestant solves the problems by himself, while ACM/ICPC contestants solve problems in teams of three.

The purpose of the simulation training is to provide a contest atmosphere for IOI contestants to retain optimum status for the upcoming IOI. Moreover, it also provides a chance for discussion with ACM contestants, especially the former IOI medallists. Contest experience and problem solving tactics of these former IOI medallists can benefit current the IOI team considerably. They're especially helpful for contestants who is about to participate in the IOI for the first time.

2.4. *Brief Summary of the Selection Process*

To sum up, after all 20 members of NTT is selected from NOI, there are four contests and activities that contribute to the selection process. They are shown in the Table 1, with corresponding weight scores.

Finally, after we sum the scores for all four parts, top 6 contestants of NTT will get the chance to participate in the final oral defence. After that, the best 4 contestants forming IOI China team are announced.

3. Several Approaches to Assure Task Quality

Task creation is at the heart of contest arrangement. High-quality tasks not only boost the overall public valuation of the competition, but also ensure effective selection of talented students and fair game.

3.1. *Organizing the Task Creation Team*

(1) Core members of task creation committee consist of former IOI and NOI medallists, experienced ACM/ICPC contestants (both present and former) and faculty

members of university. This assures the members have a good understanding of high level training and the ability of solving contest problems. Because the members have their own jobs or studies, they serve as volunteers usually.
(2) Members of two committees, NOISC and NOICC, play a key role in the work of contest task creation. NOISC holds meetings to discuss the tasks for every contest.
(3) Besides faculty members of university, there are also some student members in NOISC. They are undergraduate students or graduate students, including PhD candidates. This special treatment plays a very important role in the task creation.

3.2. *Preserving Task Innovation*

(1) Core members of task creation team are usually active in the various programming contests. They participate in the ACM/ICPC and other programming competitions with strong competency. For example, the 6 members of task creation team for NOI2009, making use of the time between the two contest days of NOI2009, flew to Shenzhen to take the Tencent Innovation Programming Contest. The 6 members won 2nd–6th and 9th place respectively in the competition. All the members of task creation team are the ACM/ICPC present contestants or former ones, including ICPC world finals gold medallists. This assures the NOI tasks with the advanced issues and concepts.
(2) Students with strong theoretical and algorithmic background tend to utilize the ideas from novel algorithms and data structures for the task creation. There were tasks related to all kinds of applications of algorithms and knowledge, such as the maintenance of the segment tree, computational geometry and network flow, etc. This extended the range and depth of tasks.
(3) Problem setters are often trying to make problems interesting to attract more contestants. There were quite a few tasks with interesting stories behind, such as the task "Plants vs Zombies" (NOI'2009), N^2 digital games (puzzle, CTSC'2009), Target-shaped Sodoku (NOIP'2009). These tasks are attractive to students. It stimulated youth's creative imagination and study enthusiasm.

3.3. *Task Discussions and Publications*

(1) Summing-up and idea-exchanging are also very important to improve the task quality. When the manuscript of a task is finished, we arrange different discussion and to ask for suggestions.
(2) There is a task solution report and discussion after every contest. It is usually arranged in the afternoon of the each contest day. Contestants are encouraged to ask questions and join discussions during the session. It is very helpful to the discovery the imperfection of tasks and usually leads to quality improvement for the future.
(3) The Publication of the yearbook of NOI. We have published a yearbook of NOI each year since 2006. Each yearbook covers all the tasks with solutions, from all 4 contests during that year. Official test data and reference solutions can also be

found in the companion CD. The yearbook has become the necessary reference material for informatics olympiad field in China. We began to publish NOI tasks in English since 2009.

(4) We also encourage and support the publication of books related to NOI trainings and contests, especially comprehensive skills, the art of programming languages, algorithms, contest task solutions, and dedicated contest training textbooks in algorithms and data structures. Mr. Rujia Liu, past NOISC student member and current NOICC member, has designed over 30 tasks for CNOI and ACM/ICPC Asia regional contests. He published his first book "*The Art of Algorithms and Programming Contests*" in 2004. Then he translated "*Programming Challenge*" into Chinese and published it in 2009. Recently, he's planning to publish a book series of the art of algorithms and programming contests.

4. Conclusion

This article gives some ideas and existing practice establishing the selection mechanism for the IOI China team, and some approaches and management experience to assure tasks quality of CNOI during the past years. The selection mechanism and assessments for the best contestants has been proven to be effective. CNOI plans to continue building up a powerful task creation team, and preserving task innovation and exploration by the untiring effort of NOISC and NOICC under the guidance of the CCF.

Appendix. IOI China Team Selection Competition (CTSC) Task Example

Magic Garden
(Proposed by Weidong Hu, revised by Madhavan Mukund)

[Task Description]
The Magician Dongdong has a beautiful magic garden that is full of flowers all the year round.

The garden's watering system has been specially designed by Dongdong. He has conjured up n magical taps suspended in mid-air that are connected to the river nearby. When Dongdong waters the garden, each tap sends out a stream of water in an arc that falls on a single plant. Thus, the n taps together water exactly n magic plants.

Dongdong has placed all taps at the same height h. The water flows out horizontally from each tap when it is opened. The horizontal speed at which the water flows out of the tap varies from tap to tap. Since it is a magic garden, there is no air resistance and the stream of water that emerges horizontally from each tap traces out a perfect parabola under the influence of gravity. The acceleration of gravity in the magic garden is denoted by g.

The taps are arranged such that at most three streams of water pass through any single point in the air. The flow of one stream is not affected in any way when it crosses another stream in mid-air.

Over the years, the river has become polluted due to the growth of factories. Thus, Dongdong has to purify the water that falls on the plants. He can only purify the water after it emerges from the taps.

To purify the water, he creates an invisible magic filter at some height that he chooses. The magic filter is a horizontal convex region, parallel to the ground. Any water that passes through this magic filter will be purified. The energy used to create the magic filter is directly proportional to the area of the filter: 1 unit of energy is required to create 1 square unit of the filter.

Dongdong needs a filter that will purify all the water that emerges from the n taps. He wants to minimize the energy he uses to create the filter that he requires.

The taps and plants in the garden are described in terms of a 3D rectangular cartesian coordinate system. The northwest corner of the garden at ground level is the origin for the coordinate system. The x-axis runs from west to east, the y-axis runs from north to south and the z-axis runs vertically from bottom to top.

In terms of this coordinate system, the position of the ith tap is represented as (x_i, y_i, h), while the position of the jth magic plant is represented as $(x'_j, y'_j, 0)$.

[Input Format]

The first line contains two real numbers h and g, the height of the taps and the acceleration of gravity in the magic garden. The second line contains an integer n, the number of taps. This is followed by n lines, each line contains 4 integers x_i, y_i, x'_i, y'_i, separated by spaces, where (x_i, y_i, h) gives the position of the ith tap and $(x'_i, y'_i, 0)$ gives the position of the plant that is watered by this tap.

[Output Format]

The output should be a single real number, the minimum energy that Dongdong needs to use to purify all the water. Retain at least 3 digits after the decimal point in your solution.

[Sample Input]

```
36 2
3
99 100 105 100
101 100 95 100
100 99 100 105
```

[Sample Output]

```
0.000
```

[Sample Explanation 1]

All streams pass through the point (100, 100, 35). Create a filter at this point with 0 area. Since the area is 0, so is the energy required to create the filter.

[Sample Input 2]

```
10 9.8
```

```
3
0 0 0 0
1 0 100 0
0 50 0 1
```

[Sample Output 2]

```
25.000
```

[Sample Explanation 2]

Create a filter at height 10 in the shape of a right-angled triangle with vertices (0, 0, 10), (1, 0, 10) and (0, 50, 10). The area of this triangular filter is 25.000, so 25.000 units of energy are needed.

[Scoring]

For each test case, if your answer differs by at most 0.001 from the standard answer, you score 100%. If your answer differs by more than 0.001 from the standard answer but at most by 0.002, you get 50% of the score. Otherwise, you get 0.

[Constraints]

For 20% of the test cases, $1 \leqslant n \leqslant 10$;
For 50% of the test cases, $1 \leqslant n \leqslant 50$;
For 100% of the test cases, $1 \leqslant n \leqslant 100$;
$0 < h \leqslant 10000.0, 0 < g \leqslant 100.0, 0 \leqslant x_i, y_i, x'_i, y'_i \leqslant 1000$.

[Hint]

Let $L = (x_i - x'_i)^2 + (y_i - y'_i)^2$,
Initialized speed $v_0 = \sqrt{\frac{Lg}{2h}}$
The speed in horizontal direction at time t: $v_H(t) = v_0$
The speed in vertical direction at time t: $v_V(t) = gt$
The speed at time t: $v(t) = \sqrt{v_H^2(t) + v_V^2(t)}$

References

CCF (2007–2010). The competition rules and measurements of Chinese NOI, *The File of NOI SC and NOICC of CCF*.

CCF (2009). *The Yearbook of China National Olympiad in Informatics 2009* (CNOI2009), China Computer Federation (Eds.), Henan Publication Group of China.

Wang, H., Yin, B. (2006). Visualization, antagonism and opening – Towards the future of the IOI contest, In *1st Workshop on Computer Science Competitions Reform*, Germany.

Wang, H., Yin, B., Li, W. (2007). Development and exploration of Chinese national olympiad in informatics (CNOI). *Olympiads in Informatics*, 1, 165–174.

H. Wang is an associate professor at the Department of Computer Science and Technology, Tsinghua University. He received his PhD degree in computer science from Tsinghua University in 1993. He has served as the chairman of Scientific Committee of National Olympiad in Informatics (NOISC) of CCF since 2006.

B. Yin is a professor at the School of Computer Science and Technology, Beihang University. He received his PhD degree in computer science from the Department of Artificial Intelligence, the University of Edinburgh in 1984. He has served as the vice chairman of Scientific Committee of National Olympiad in Informatics since 1999.

R. Liu received his master degree from the Department of Computer Science and Technology, Tsinghua University in 2008. He has been a coach of IOI China National Training Team from 2002 to 2008. He is currently a member of Competition Committee of National Olympiad in Informatics.

W. Tang is a senior student in the Department of Computer Science and Technology Tsinghua University. He has been a coach of IOI China National Training Team from 2008. He is currently a student member of Scientific Committee of National Olympiad in Informatics of CCF.

W. Hu is a PhD candidate in the Department of Computer Science and Technology, Tsinghua University. He has been a coach of IOI China National Training Team from 2008. He is currently a student member of Scientific Committee of National Olympiad in Informatics.

Olympiads in Informatics, 2010, Vol. 4, 151–157

© 2010 *Institute of Mathematics and Informatics, Vilnius*

IOI Israel – Team Selection, Training, and Statistics

Ela ZUR, Tamar BENAYA
The Openu University of Israel, Computer Science Department
Ravutzky 108, 43107 Raanana, Israel
e-mail: {ela, tamar}@openu.ac.il

David GINAT
Tel-Aviv University, Science Education Department
Ramat Aviv, 699978 Tel-Aviv, Israel
e-mail: ginat@post.tau.ac.il

Abstract. We outline Israel's IOI (International Olympiad in Informatics) project. Israel joined the IOI in 1997 and has participated in the IOI ever since, apart from 2008. We describe the selection and training process in Israel, and provide some statistics. The selection and training process is composed of four stages: a self-study preparation; a national competition, an advanced training and team-selection stage; and the national team's preparation to the IOI. The presented statistics involve Israel's medals throughout the years, and some statistics about the top 30 students, who reached the advanced training and team-selection stage.

Key words: programming contests, IOI.

1. Introduction

The IOI – the International Olympiad in Informatics – is the primary computer science (CS) competition for young students, up to the age of 20. The IOI is one of six annual international youth olympiads, including: the IMO in mathematics, the IPHO in physics, the ICHO in chemistry, the IBO in biology, and the IAO in astronomy. The IOI is hosted every year by a different country. It started with 13 participating countries, in Bulgaria in 1989, and expanded to 80 countries today.

The primary goal of the IOI is to stimulate challenges in CS among exceptionally talented young students from all over the world, and have them share scientific and cultural experiences. Each participating country conducts a preparation process, and brings an IOI team, which includes four contestants. In the IOI, the contestants compete individually in the course of two competition days, each involving three challenging algorithmic tasks, to be solved and programmed.

The task solutions require careful task analysis, insightful correctness and efficiency considerations, and skilful programming implementation. Creativity, competence in algorithmic topics (Verhoeff *et al.*, 2006), and implementation accuracy are essential. The better half of the students in the two-day competition win gold, silver, and bronze medals.

Different countries invest different amounts of effort and resources in preparing their IOI teams (e.g., Diks *et al.*, 2007; Casadei *et al.*, 2007; Forisek, 2007; Kolstad and Piele, 2007), yet the preparation outlines seem similar. A call-for-participation engages an initial amount of interested students, from whom the top ones are chosen, through a selection and training process. In what follows, we briefly describe the selection and training process in Israel, and then display some statistics of this process and of Israel's participation in the IOI.

2. The Selection and Training Process in Israel

In Israel, the IOI project is operated and supported by Tel-Aviv University, the Open University of Israel and the Ministry of Education. The primary objective of the project is to offer challenges in CS to motivated students, who show interest and competence in problem solving in general, and algorithmic problem solving skills in particular.

The project is composed of four stages: a self-study stage towards a national competition; a national competition, an advanced training and team-selection stage, and the national team's preparation to the IOI. The different stages are operated by a small training team, of five to six trainers – the head coach and his deputy, a couple of high-school teachers, and a couple of former IOI contestants.

The 1st stage is conducted in the beginning of winter. It starts with a call-for-participation sent to high-schools and posted in the national CS teachers' website (maintained by the high-school CS inspector in the ministry of education). Then, some of the project trainers (the head coach and the former IOI contestants) lecture about the IOI project in different high-schools and learning centers of talented young students. The interested students are referred to the project's website (`http://www.tau.ac.il/~cstasks`), and are encouraged to prepare to the national competition, by self-studying rather basic programming and data-structure constructs (e.g., recursion and trees) and solving previous national competition tasks.

The 2nd stage is conducted in the late winter (February). It involves the national competition, which is a three-hour exam, with pencil and paper. The students are gathered together, and are asked to solve four algorithmic tasks, and provide a written description of their solution idea and their solution code, or pseudo-code (according to their preference). The goal of the exam is to identify the students that demonstrate the highest potential, primarily in problem solving. Thus, the CS knowledge required at this stage is relatively basic.

The first task of the national competition usually requires recursion, which may be implemented with a rather simple dynamic-programming scheme. The second task involves a mathematical game, or a similar task, whose solution is based on a hidden invariant property. The third and fourth tasks are more involved, in terms of the required insight and the solution scheme. Yet, the code required for each of the tasks is rather short. The students are explicitly directed to focus on task analysis, and carefully notice correctness and efficiency considerations. In grading their solutions, we particularly examine their

creativity, accuracy, and scientific discipline. We pay less attention to detailed programming features, as long as the criteria indicated above are met.

The amount of students taking part in this stage diverts between the years, from about one hundred to several hundreds (see next section). We select the best 30 students, plus possibly a few additional ones, in cases where there are females or students from remote schools that are close to the top 30. All these students are invited to the next stage. Although only a small amount of students advance to the next, 3rd stage, our experience shows that the vast majority of students enjoy the challenge of the competition, and many return a year later, following some better preparation.

The 3rd stage is conducted in the spring. Our objective in this stage is to teach the top 30 students more advanced algorithmic and problem solving features, and test them about these features. The top four students of this stage are chosen to the national team. This stage involves 5–7 practice days (one or two such days a week). It does not involve a camp (as offered in some other countries), but rather a day gathering in a computer lab, due to our limited resources.

Each practice day lasts 8–10 hours. Prior to that day, students are asked to study particular topics (e.g., basic graph algorithms). In the first part of the day, they are posed with three algorithmic tasks to program in five hours, which involve the indicated topics and the previous days' topics. The students are asked to both program their solutions and write on paper their solution's underlying idea. At the end of this activity each student is interviewed about his/her solution. Our goal in the interviews is to examine their insight and extract potential errors and difficulties that arise and recur. In addition, the student programs are tested on diverse test-cases.

Following the interviews and the program evaluations, all the participants are gathered for a two–three hour discussion on the day's task solutions and their related CS topics. The discussion involves particular focus on insightful analysis, common errors, and essential efficiency considerations. The latter is particularly underlined, as many of the posed tasks may be solved in several ways, of different time and space complexities. We strongly emphasize two elements: potential and recurring errors, and algorithmic and problem solving features used in the day's task solutions, which are relevant beyond these tasks (e.g., particular task representations and illuminating perspectives). Some of these elements are described in papers and columns of the third author of this paper (e.g., Ginat, 2001; 2003a; 2003b; Ginat and Hasty, 2007).

At the end of the practice day, the students are asked to: program at home alternative solutions that were discussed, and further study the algorithmic and problem solving features that were examined. At the end of these 5–7 practice and evaluation days, we select to the national team the four students that demonstrated the best accumulated performance, in both algorithmic problem-solving and programming. The rest of the students are encouraged to return in the following year and convince other students from their schools to join as well.

The 4th stage is conducted thereafter and usually lasts up to two months, until the IOI. In this stage, the team is directed to learn and practice the topics relevant for the IOI, solve previous IOI and additional olympiads tasks, and thoroughly practice the programming

features required in the IOI. The team members meet with the project trainers once every one or two weeks, practice task solutions, discuss solutions, and receive advice and tips from previous team members who competed in the IOI. A particular emphasis is put on one's selection of test-cases before submission. The teams' record throughout the years is described in the next section.

3. Some Statistics

3.1. *Medal Distribution*

Table 1 presents the numbers of gold, silver and bronze medals received by the Israeli team since 1997, excluding 2008, when Israel did not participate. The four gold medals were earned by four different students.

3.2. *Participation in the National Competition*

The number of students who participate in the national competition varies from year to year, ranging from 40 (in the 1st year) to 507 (Fig. 1). These students come from approximately 20 to 70 different high schools located all over the country.

3.3. *The Top 30 Students*

Following the national competition, the best 30 students are selected for the next, advanced stage. These students come from 59 high schools located all over the country. Fig. 2 shows the distribution of the students among 30 high schools from which more

Table 1
Achievements of the Israeli team in the IOI

Year	Gold medal	Silver medal	Bronze medal
1997	–	2	2
1998	–	2	2
1999	–	1	1
2000	1	2	–
2001	1	1	1
2002	1	–	2
2003	–	1	–
2004	–	1	3
2005	1	2	–
2006	–	–	3
2007	–	4	–
2009	–	–	1
Total	4	16	15

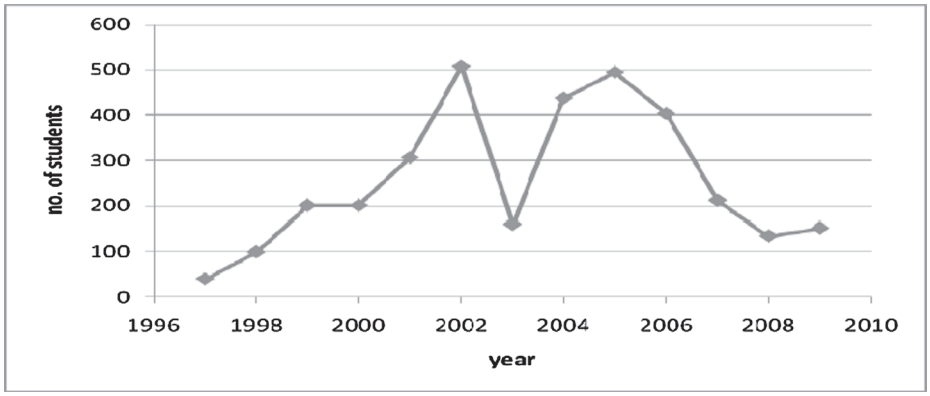


Fig. 1. Number of participants in the national competition.

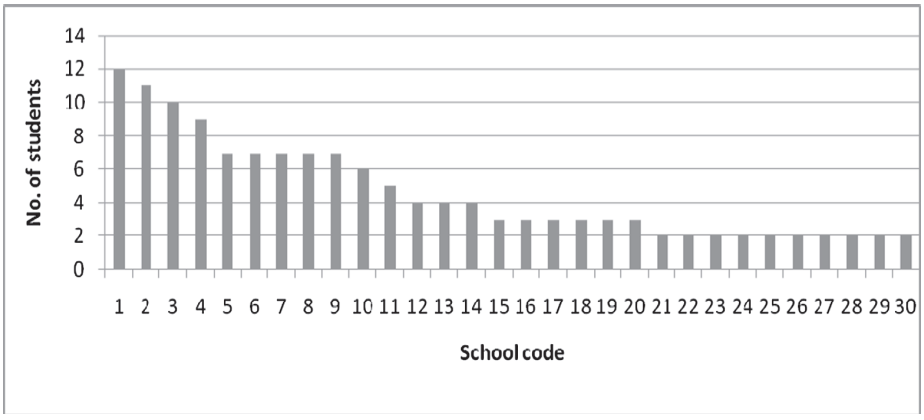


Fig. 2. Top 30 students – school distribution.

than one student was selected in the years 2004–2009. The rest of the 29 high schools had one student who made it to the top 30.

From Fig. 2 we can see that the best students are concentrated in a relatively small number of high schools. 50% of the best students come from 17% of the high schools. Fig. 3 shows the geographical distribution of the students among the different high schools in the years 2004–2009.

One can see that about 60% of the students come from the center of the country. An interesting fact, not in the figures above, is that in all the twelve years of the IOI activities in Israel only one female student entered the top 30 student group, in 2009. This female was actually among the top 8 students. Unfortunately, although she was very competent in algorithmic problem solving, she was less able in programming and solution implementation. Another interesting anecdote is that all our four gold medalists came from schools that do not have many representatives among the top 30 students.

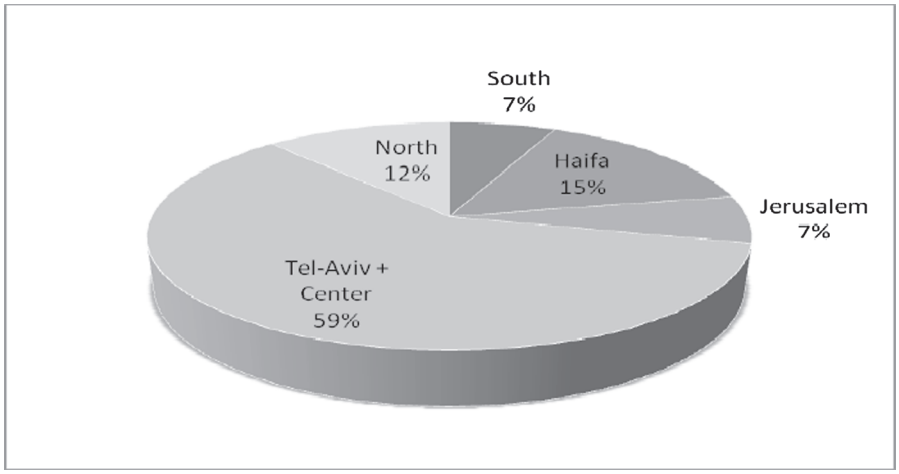


Fig. 3. Top 30 students – geographical distribution.

All in all, the IOI project in Israel is rather modest. Our hope is to extend our resources and activities in the coming years, expand our training team, hopefully with additional IOI veterans, and attract a larger number of interested students (males and females) already in the early stages.

References

Casadei, G., Fadini, B., Genovie De Vita, M. (2007). Italian olympiads in informatics. *Olympiads in Informatics*, 1, 24–30.

Diks, K., Kubica, M., Stencel, K. (2007). Polish olympiad in informatics – 14 years of experience. *Olympiads in Informatics*, 1, 50–56.

Forisek, M. (2007). Slovak IOI 2007 team selection and preparation. *Olympiads in Informatics*, 1, 57–65.

Ginat, D. (2001). Misleading intuition in algorithmic problem solving. In: *Proc of the 32nd ACM Computer Science Education Symposium – SIGCSE*, ACM Press, 21–25.

Ginat, D. (2003a). The greedy trap and learning from mistakes. In: *Proc of the 34th ACM Computer Science Education Symposium – SIGCSE*, ACM Press, 11–15.

Ginat, D. (2003b). Board reconstruction, colorful challenges column. *SIGCSE Bulletin*, 35(4), 25–26.

Ginat, D. (2007). Hasty design, futile patching and the elaboration of rigor. In: *Proc of the 12th Conference on Innovation and Technology in Computer Science Education – ITiCSE*, ACM Press, 161–165.

Kolstad, R., Piele, D. (2007). USA computing olympiad (USACO). *Olympiads in Informatics*, 1, 105–111.

`http://www.tau.ac.il/~cstasks`.

Verhoeff, T., Horvath, G., Dicks, K., Cormak, G. (2006). A proposal for an IOI syllabus. *Teaching Mathematics and Computer Science*, 4, 193–216.

E. Zur is involved in the Israel IOI project since 1997, and repeatedly served as a deputy leader. She holds a PhD Degree in computer science education from Tel-Aviv University. She is a faculty member of the Computer Science Department at the Open University of Israel. She designed and developed several advanced undergraduate cmputer sience courses and workshops, and she serves as a course coordinator of several courses. Her research interests include distance education, collaborative learning, computer science education, computer science pedagogy, teacher preparation and certification and object oriented programming.

T. Benaya holds a MSc in computer science from Tel-Aviv University. She is a faculty member of the Computer Science Department at The Open University of Israel. She designed and developed several advanced undergraduate computer science courses and workshops, and she serves as a course coordinator of several courses. She also supervises student projects. She is a lecturer of computer science courses at the Open University of Israel and Tel-Aviv University. Her research interests include distance education, collaborative learning, computer science education, computer science pedagogy and object oriented programming.

D. Ginat heads the Israel IOI project since 1997. He is the head of the Computer Science Group in the Science Education Department at Tel-Aviv University. His PhD is in the computer science domains of distributed algorithms and amortized analysis. His current research is in computer science and mathematics education, focusing on cognitive aspects of algorithmic thinking.

Olympiads in Informatics, 2010, Vol. 4, 158–169

© 2010 *Institute of Mathematics and Informatics, Vilnius*

Get Involved! The IOI Workshop 2010, Its Goals and Results

Wolfgang POHL[1], Benjamin A. BURTON[2], Valentina DAGIENĖ[3], Jittat FAKCHAROENPHOL[4], Michal FORIŠEK[5], Mathias HIRON[6], Mārtiņš OPMANIS[7], Bronius SKŪPAS[3], Willem van der VEGT[8]

[1] *Bundeswettbewerb Informatik, Ahrstr. 45, 53175 Bonn, Germany; e-mail: pohl@bwinf.de*
[2] *School of Mathematics and Physics, The University of Queensland, Brisbane QLD 4072, Australia; e-mail: bab@maths.uq.edu.au*
[3] *Informatics Methodology Department, Institute of Mathematics and Informatics, Akademijos 4, Vilnius LT-08663, Lithuania; e-mail: {dagiene, bskupas}@ktl.mii.lt*
[4] *Department of Computer Engineering, Kasetsart University, Bangkok, Thailand; e-mail: jittat@gmail.com*
[5] *Department of Informatics, Faculty of Mathematics, Physics and Informatics, Comenius University, Mlynská dolina, 842 48 Bratislava, Slovakia; e-mail: forisek@dcs.fmph.uniba.sk*
[6] *France IOI; e-mail: mathias.hiron@gmail.com*
[7] *Institute of Mathematics and Computer Science, University of Latvia, Riga, Latvia; e-mail: martins.opmanis@lumii.lv*
[8] *Windesheim University for Applied Sciences, School of Education, PO Box 10090, 8000 GB Zwolle, The Netherlands; e-mail: w.van.der.vegt@windesheim.nl*

Abstract. In May 2010, the third IOI workshop took place in Schloss Dagstuhl, Germany. It was motivated by the discussions held at and after the panel session of 2009's IOI conference in Plovdiv. There, discussions focussed on communication and collaboration among the IOI community, as well as communication of the IOI competition to outsiders. At the workshop, members of the IOI community met to develop a first version of an IOI Wiki as a tool for communication and collaboration, and devised suggestions on how to visualize IOI-style contests to make them more accessible to the outside world.

Key words: programming contests, IOI, wiki-based collaboration, visualization.

1. Motivation and Introduction

The International Olympiad in Informatics (IOI) has a 20-year tradition, but keeps evolving. Regularly, IOI organizes a workshop, in order to bring forward new ideas and to explore how they could be implemented. In recent years, IOI workshops were held in Germany (Schloss Dagstuhl, 2006) and in The Netherlands (Enschede, 2008).

In the discussions that were held during and after the panel discussion at the IOI Conference 2009, "communication" appeared to be the central keyword. Many open questions are linked to this topic:

- Communication beyond the IOI community:

  - How to increase awareness of IOI, and what are the problems to be solved while pursuing that goal?
  - What image of IOI should be communicated to the outside world?
- Communication within the community:
  - Why does the IOI community not communicate between IOIs? Are there tools or incentives that could stimulate this communication?
  - How to encourage communication between contestants?
  - What are the subjects of common interest to communicate about?
- Communication and sharing resources:
  - How to encourage and facilitate the sharing of resources for training and contest organization?

These and related topics were to be the focus of the IOI Workshop 2010, to be held again at Schloss Dagstuhl, Germany. In their call for contributions, the workshop organizers had envisioned a meeting of people who were interested, willing and capable to make first steps of devising, developing, and implementing solutions to the problems discussed.

The submissions focussed on two main areas: On the one hand, several authors stressed the need for collaboration among IOI delegations and the need for tools to support that collaboration. On the other hand, some submissions suggested new ways of enriching IOI-style competitions with a higher degree of visual appeal, in order to make them more accessible for both persons with expertise in the competition area (like members of the IOI community themselves) and people with mere interest in the proceedings and results of the competition (like friends or relatives of the participants). Hence, during the workshop, most of the time the participants divided into corresponding working groups. In the following two sections, we report on the results of these working groups.

2. IOI Wiki: Getting the Community Involved in Collaboration

It is obvious that in most IOI countries, two kinds of activities are done on the national level[1]:

- Selection of IOI participants, typically by means of a (national) contest or specific exams.
- Training of IOI participants or team candidates, sometimes within, sometimes following the selection process.

Often, the people who are involved in the above IOI-related activities, are working in a more general context of informatics education: as school teachers, as academics who are responsible for curriculum development or teacher training related to informatics education in schools, or as activists that bring forward non-curricular education activities in countries where informatics education in school has been established insufficiently

[1] We do not give explicit references, but many papers on such activities have been published in previous "Olympiads in Informatics" volumes.

or not at all (examples: USACO training, French training site "france-ioi.org", German community site "einstieg-informatik.de").

Within all these activities, the persons involved do similar things, use similar material, are interested in similar news, and discuss similar ideas. Unfortunately, much of this work at individual and national levels does not make it into the collective knowledge of the IOI and informatics education community. This is particularly true of knowledge that is more practical than academic, which often does not appear in conference papers or talks even though it deserves attention and could be beneficial and stimulating for others. In particular, there is no organized way of exchanging what is perhaps the most valuable "raw material" for national team coaches: tasks that are good for use on the level of national IOI training and preparation.

Therefore, it was suggested that we establish a system for sharing information and material – including tasks amongst other things – among the IOI community and other informatics educators. This was done as follows:

- Before the workshop, Mathias Hiron conducted a detailed survey of tools that could be used to build such a system. According to his suggestion, the working group chose the MediaWiki software along with Semantic MediaWiki extensions. This would allow the material on the system to be systematically organized and retrievable.
- During the workshop, two working groups developed infrastructure and sample content for various aspects of this wiki, including a database of publications and a repository of tasks (each discussed separately below). Another working group developed a taxonomy of IOI topics and activities to tie together the different aspects of the wiki, and to assist with categorization.
- Implementations were done in a prototype wiki at `http://www.bwinf.de/ioi-cooperation`.

2.1. *A Suggestion for an IOI Taxonomy*

In a MediaWiki, content items are often assigned to categories. Categories are then organized into a hierarchy, and the Semantic MediaWiki extension is able to infer that, if category A is a subcategory of category B, and a content item is explicitly assigned to category A, it also belongs to category B. A sound hierarchy of categories therefore is quite useful when retrieving content items.

In one working group of the workshop, a taxonomy of categories for the IOI Wiki was discussed. Three main branches of this taxonomy were suggested (see Fig. 1):

IOI-Item the kinds of objects that content items (i.e., Wiki pages) may describe, e.g., contests, delegations, people, publications, and tasks;

IOI-Content a taxonomy that characterizes the scientific area IOI is related to, which becomes useful for categorizing tasks (amongst other things); the main branches of this sub-taxonomy cover data structures, algorithms, algorithmic strategies, and programming;

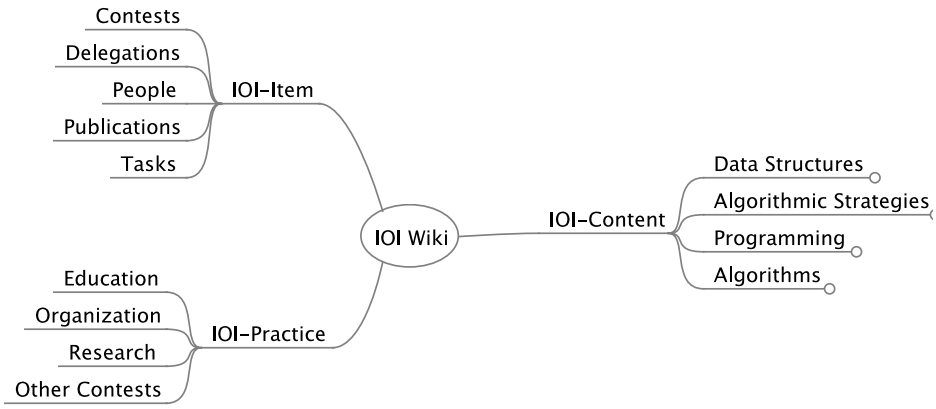


Fig. 1. The main branches of the suggested IOI taxonomy.

IOI-Practice the areas where members of the IOI community are typically active, like education, organization (of contests), research, and other contests (like ACM ICPC etc.); this branch might be used to classify people and publications.

2.2. *A Database of Publications*

The first archival component developed for the wiki was a database of publications relevant to the IOI. This database aims to help users locate specific publications, and more importantly to expose them to new material that could be beneficial to their own endeavours. Since the workshop this database has been populated with over 80 books, journal articles, conference papers and online articles relevant to the IOI community.

An important outcome of the publications database was to illustrate how MediaWiki and the Semantic extension could be used to develop a rich archive of user-editable data. In effect, this paved the way for the next (and more important) development: a repository of contest tasks.

2.3. *A Task Repository*

For organizers of IOI-style programming contests, and for IOI team coaches all over the world, good tasks are a most valuable resource. Various members of several IOI bodies (GA, IC, and ISC) together submitted to the workshop the suggestion to develop an international repository of tasks. Such a repository would be a collection of tasks and associated resources (such as test data, translations and solutions), with the following main purposes:

1. To offer a general and usable resource that team leaders can use for training – that is, a large pool of tasks indexed by difficulty, algorithm type and so on.
2. To assist less experienced countries with their own exam setting, by explicitly flagging and quarantining tasks that have not been widely published (and therefore can be considered for use in other exams).

At the workshop, a Semantic MediaWiki template for tasks was developed. According to this template, an item in the task repository would consist of the following parts:

- identification of the task: a unique ID within the repository, information about the source and the previous usage of the task, and license information.
- general information: the task type, the topics involved (with reference to the IOI-Content part of the taxonomy), a very brief description of the task, a brief sketch of the solution, and a difficulty estimate.
- task details: the task statement itself, a solution document, details on limits, and a grader.
- a file archive containing typical task files: statement, test cases, etc. in a specified format.

Fig. 2 shows an example task page in the Wiki. Of course, not all details of a task repository could be defined and implemented at the workshop. Further discussion within the whole IOI community is needed to make this ambitious project succeed.

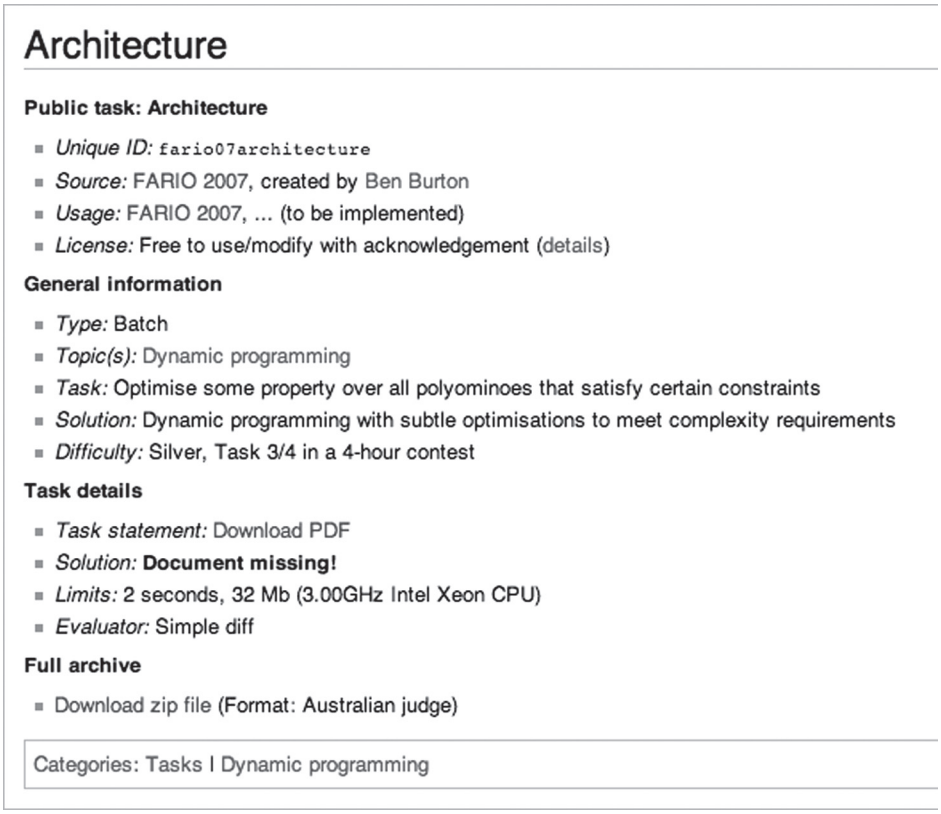


Fig. 2. Example task in the task repository.

2.4. *Technical Aspects*

Data in the IOI Wiki are typically stored in a well-structured way, organized by Semantic MediaWiki templates. Fig. 3 displays the edit page for a specific user (top); the editor window contains this user's template code. Below, the resulting user page is shown; the list of papers authored by this user is automatically generated, according to the template.

Most parts of the IOI Wiki are planned be open to the public. That is, any member of the IOI community shall be allowed to edit most pages of the Wiki. Editing of structured content as seen above can be done pragmatically by copying and modifying code of existing pages. In the IOI Wiki, editing of template-based pages will be made possible via forms; at the workshop, significant steps towards implementing forms into the IOI Wiki were made.

While the majority of IOI Wiki contents shall be publicly accessible and editable, access to the task repository needs to be limited. At the workshop, it was argued that finally two separate wikis should be setup for the IOI community: one public, one private. However, double maintenance of data must be avoided. At the workshop, we investigated techniques to automatically transfer data between two wikis.

2.5. *Further Wiki Content*

In addition to publications and tasks, the IOI Wiki could contain much more material of interest to the IOI community. For instance, the wiki already contains frameworks for a directory of people (including members of the IOI community as well as other people relevant to IOI) and a list of delegations to the IOI. Further content might include a calendar of events, or material for secondary-school-level education in informatics, etc. – suggestions are very welcome.

3. Visualization: Getting People Involved in IOI

Human beings are visual animals, attracted and influenced by what they see. So far, the visual appeal of programming contests, and of IOI in particular, is more than lacking. In brief: IOI is fun for contestants but boring for spectators. At the workshop, we tried to summarize different ways to increase the attraction value of a contest using visualization. Three main aspects were discussed:

1. visualizing the scoreboard;
2. visualizing the output of contestants' programs;
3. visualizing the input or the task itself.

3.1. *Visualizing the Scoreboard*

Ideas from the Thailand Code Jom competition scoreboard (where teams are represented as "bubbles" floating under water) and, in parts, from the Gapminder software (see `www.gapminder.org`) were combined into a prototype of a new IOI scoreboard.

Fig. 3. A user page: structured data and automatic content generation.

Fig. 4 is a screenshot of that prototype. The current standings are presented as a 2-dimensional plot of contestants' IDs, where the y-axis represents the current score, and the x-axis can be chosen to represent various parameters – including age, number of submissions, or delegation – thus enabling the scoreboard to show various aspects of the current standings. The audiences can also choose to narrow the scoreboard's display down to the standings of a particular task, country, or contestant that they know. Of course, such a highly visual scoreboard could just provide a display alternative to a standard tabular scoreboard with alphanumeric display of contestants' names and current scores.

Beyond general standings, the scoreboard should also provide access to competition status as well as to background information for each contestant. For that purpose, the individual contestant "bubbles" (or lines in a tabular display) should be linked to user pages. Fig. 5 shows two example user pages, with overall statistical information on the contestant's current standing (left) and task-specific details (right).

3.2. *Visualizing the Output of Contestants' Programs*

We analyzed previous tasks of IOI and some other contests to see if and how the output of a task could be visualized. With respect to presenting the output of a contestant, a task may fall into one of the following five categories:

Not Suitable It is hard or impossible to present the output using a graphical presentation. This can be the case where output is a single number, or where the data used include very large numbers.

Static Boolean The output is of a static nature. The program has reached a way to solve the task, the answer can be shown, but not the way to get to the answer. There is only one good solution for the task with a specific test case; the contestants answer is either right or wrong.

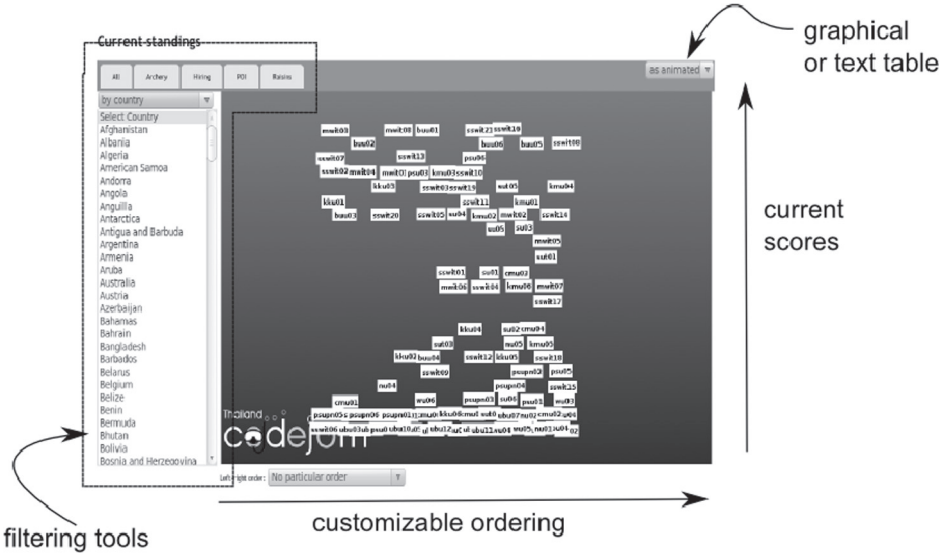


Fig. 4. Prototype of an interactive scoreboard.

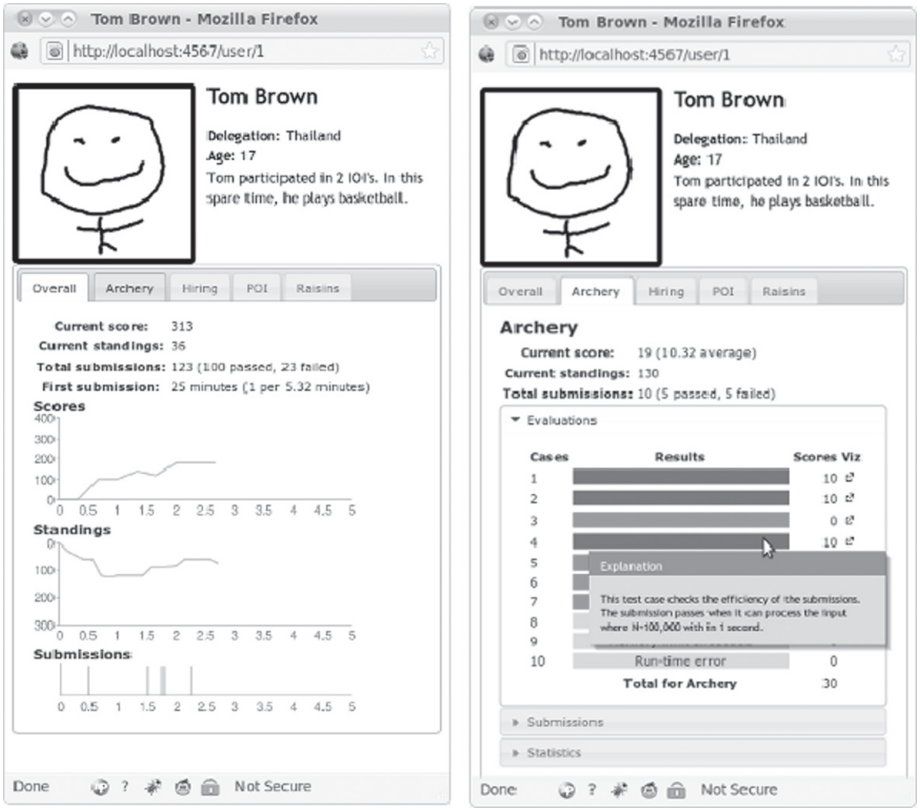


Fig. 5. Example user pages linked to the scoreboard.

Static Fractional The output is of a static nature, but partial credits for suboptimal solutions are allowed.

Dynamic Hidden The output is the result of some intermediate steps. These steps however are not part of the output. In this case it is hard or even impossible to present the output using a graphical presentation.

Dynamic Disclosed When the output gives the answer to the question and the way to reach it, a nice presentation is possible. A spectator can get the chance to browse through a solution and to examine the steps taken by the program.

Another issue concerns the size of the test cases. When the sizes can get very large, maybe only the small test cases can be presented.

Three categories are suitable for visualization right away: Static Boolean, Static Fractional, and Dynamic Disclosed. At the workshop, plentiful examples of all these categories were found among tasks of former IOIs and other IOI-style contests, for instance: Packing Rectangles (IOI 1995, category Static Boolean), Map Labeling (IOI 1997, Static Fractional), and Underground (IOI 1999, Dynamic Disclosed). For further details, see the special report of the visualization group.

3.3. *Visualizing the Input or the Task Itself*

The task description can be and in many tasks has been illustrated by images and diagrams. But additional tools, like a sample program to visualize the action of the intended program or the structure of the data used for the task, can not only help contestants but also clarify the tasks for the audience.

4. Conclusion

At the IOI workshop 2010, we made concrete first steps towards (1) communication and collaboration within the IOI community, and (2) increasing the visual accessibility of the IOI competition. We developed prototypes of an IOI Wiki and a scoreboard visualization tool, and made specific suggestions for visualizing contest tasks and the output from solutions. In doing this, we aim to revitalize the ongoing discussion on how to further proceed to reach our long-term goals. If the outcomes of this workshop can activate and involve many IOI community members and other people in this discussion, the workshop can be considered a success.

Acknowledgements

Benjamin Burton, Mathias Hiron, and Wolfgang Pohl prepared the IOI workshop 2010 as a team. They thank all submitters and workshop participants in particular for their suggestions as well as for their hard and highly effective work on the workshop projects. We thank Gordon Cormack, who gave a tele-presentation at the IOI workshop 2010, enlightening the workshop participants on plans for IOI 2010 to increase the accessibility of the IOI competition. This presentation highly motivated and inspired the workshop participants and the visualization group in particular.

The authors at the IOI Workshop 2010 in Schloss Dagstuhl.
From left to right: Jittat Fakcharoenphol, Bronius Skūpas, Wolfgang Pohl, Valentina Dagienė, Benjamin Burton, Mathias Hiron, Michal Forišek, Willem van der Vegt, Mārtiņš Opmanis.

W. Pohl has been responsible for Bundeswettbewerb Informatik, the German national contest in computer science for high school students, since 1999. It is among his duties to take care of Germany's IOI training, selection, and participation; for the IOI, he served as IC member from 2003–2006. In addition, he initiated and now manages the German implementation of the international Bebras contest and started "Einstieg Informatik", a web portal and community for youth interested in computer science.

B.A. Burton directed the Australian IOI training programme from 1999–2008, and currently sits on the IOI Scientific Committee. In 2013 he will chair the Host Scientific Committee for the IOI to be held in Brisbane, Australia. He enjoys working on research projects in geometry and topology that lie at the crossroads of mathematics and computer science, and is supported by the Australian Research Council under the Discovery Projects funding scheme (project DP1094516).

V. Dagienė is head of department at the Institute of Mathematics and Informatics. She has published over 150 scientific papers, written more than 60 textbooks in the field of informatics for secondary education and teacher training. She has been chair of Lithuanian Olympiads in Informatics for many years, organized Baltic Olympiad in Informatics in 1997, 2002, and 2005, and established the International Contest on Informatics and Computer Fluency "Bebras". She is vice chair of the IFIP (International Federation for Information Processing) TC3 Committee on Education, and member of the International Committee on Olympiads in Informatics (2006–2012). She is the editor-in-chief of the international journal "Informatics in Education" (since 2002).

J. Fakcharoenphol works at the Department of Computer Engineering, Kasetsart University, Bangkok Thailand. He was part of the training team of Thailand delegations until late 2009. As Thailand will host the IOI 2011, he is a member of the ISC.

M. Forišek is a lecturer at Comenius University in Bratislava, Slovakia. In years 2006 to 2012 he is serving as a member of the International Scientific Committee of the IOI, and presently he is the chair of this committee. He is also involved in organizing various other international competitions, notably CEOI 2002, 2010 and the annual Internet Problem Solving Contest.

M. Hiron is the president of France-IOI, the organization that is in charge of selecting and training French students for the IOI.

M. Opmanis is researcher at the Institute of Mathematics and Computer Science of University of Latvia. He has been team leader of Latvian team at all IOIs since 1996 and at many Baltic OIs since 1995. He headed the jury of the Baltic Olympiad in Informatics in 1996, 1999, and 2004. His main interest is in preparation of good tasks for various competitions in informatics and mathematics.

B. Skūpas is a PhD student in the Institute of Mathematics and Informatics, which is taking care of finding and training gifted students in mathematics and informatics. Also, he is teacher at Vilnius Lyceum. He is interested in automatic grading systems.

W. van der Vegt is teacher trainer for mathematics and computer science at Windesheim University for Applied Sciences in Zwolle, the Netherlands. He is contest director for the first rounds of the Dutch Olympiad in Informatics and joined IOI in 1992.

Olympiads in Informatics

Volume 4, 2010

ISSN 1822-7732

0868-4952(2010);1-Q