Olympiads in Informatics, 2007, Vol. 1, 57–65

Slovak IOI 2007 Team Selection and Preparation

Michal FORIŠEK
Department of Computer Science, Faculty of Mathematics, Physics and Informatics
Comenius University, 842 48 Bratislava, Slovakia
e-mail: forisek@dcs.fmph.uniba.sk

Abstract. We describe the process of selecting and preparing the Slovak IOI team. The presented information is related to the situation in the year 2007. We present the structure of the Slovak national Olympiad in informatics and of the Slovak selection camp, including the rationale behind their current form. We list several other national activities that help our students acquiring extra-curricular knowledge in computer science. We discuss our cooperation with other countries in the region. Finally, we give several examples of tasks used in our Olympiad.

Key words: olympiad in informatics, Slovakia, IOI, training, preparation, programming competition, programming contest.

1. National Olympiad

1.1. *Structure*

The Slovak national Olympiad in informatics (OI) is divided into two categories. The easier category B is intended for ages 15 to 16, the harder category A for ages 17 to 18. (The rules specify exact limits based on the attended class, here we provide approximate ages instead.)

In category B the contest has two rounds – a home round and a regional round. In category A, there is an additional national round. The corresponding rounds in both categories have the same structure, we will describe them below.

1.2. *Home Round*

This is a long-term round. The contestants have at least a month of time to work on their solutions. Contestants are given four theoretical tasks of algorithmic nature. Refer to Appendix A for an example task.

One of the four tasks is accompanied by a study text. This study text (usually) describes a non-traditional computational model that will be used in tasks throughout the year, i.e., in all three rounds. In the accompanied tasks the students are supposed to solve a problem within the given computation model. The actual tasks used form a graduated difficulty series – many times a task from an earlier round serves also as preparation for harder tasks in later rounds. For more details and an example refer to Appendix B.

For each of the other three tasks the contestants are supposed to find a correct algorithm that is as efficient as possible, implement it, and provide a discussion of its correctness and time complexity. All submitted solutions are evaluated by human judges and awarded points on a scale 0 to 10.

Usually about 150 students participate in this round, and all of them are invited to the regional round.

1.3. *Regional Round*

The regional round takes place in all (eight) regions of Slovakia at the same time. In each region, the students are invited to some place where they have to solve four tasks in four hour. The type of tasks is identical to the home round (i.e., three algorithmic tasks and one task related to the computational model introduced in the home round).

All four tasks are solved on paper only. After the round is over, each region evaluates the submitted solutions. All solutions are then sent to the Olympiad's national committee for coordination. During the coordination process all solutions are re-evaluated (checking for potential mistakes in the preliminary evaluation), and the same point scale is applied to all solutions.

After the coordination the regional ranking lists are merged to create a national ranking list, and based on this ranking list approximately 30 best students are invited to the national round.

1.4. *National Round*

The national round consists of two competition days. The first day is similar to the previous rounds (four and a half hours, three theoretical tasks, one of them using the computational model).

On the second day the students have four hours to implement their solutions of the two algorithmic tasks. The solutions are evaluated using automated testing and points awarded to a solution are proportional to the number of test data batches it is able to solve correctly.

Note that the second day is similar to the International Olympiad in Informatics (IOI). For readers not acquainted with the IOI contest format we recommend referring to (Pohl, 2006) and (Dagiene, 2006). Same as at the IOI, compilers from C/C++ and Pascal are available on the practice day.

For each of the days the maximum is 30 points. Based on the total results of both days, the national champions are announced, and approximately ten best participants are invited to the national selection camp where the representation for the IOI is determined.

1.5. *Contest Format Rationale*

We strongly believe that the thinking process (in other words, problem solving process) is the most important skill we want to see in our contestants. This is what they will need in their future lives, should they pick a career in computer science.

The IOI, "practice only" competition style fails to achieve this goal properly. Practical competitions are, and always will be, at least partially focused on the actual implementation and marginal issues such as debugging techniques, library knowledge, etc.

In our point of view, the IOI competition format is in some sense just a necessary evil – the number of contestants and the amount of available time make it practically impossible to evaluate the IOI in any other way.

Slovakia's 5 million inhabitants make it a relatively small country and this is one of the factors that enable us to pick a different competition type for our national Olympiad. Therefore we still keep the competition mostly theoretical, and award points based on the thought process of the contestants. We are convinced that from a long term point of view, this competition type will actually be more beneficial for the contestants.

The contest format where the evaluation is based on ideas presented in the submitted work is also assumed to make the competition more accessible to girls. See (Fisher and Cox, 2006; Boersen and Phillipps, 2006) and (Boersen, 2006) for a discussion of related topics. We may support their theories. In 2007, out of 29 participants in the Olympiad's national round, four were girls, the average in past years is about two. Many more girls take part in the correspondence seminar (described below).

One more note on the IOI competition itself: With a strong theoretical background it is much easier for the contestants to subsequently adapt to the IOI competition style – they are halfway there, all they need is to practice implementing their ideas in an efficient and correct way (see also appendix B for rationale on including tasks in non-traditional computation models).

2. National Training

2.1. *Selection Camp*

The national selection camp is a camp organized for approximately ten best participants of the national round. The program of the camp is almost entirely focused on practical programming. During a typical day, there is a half-hour warm-up session and a three hour coding session in the morning, a four hour coding session in the afternoon, and a presentation of sample solutions thereafter.

In the warm-up sessions the contestants have half an hour to solve two simple theoretical problems on paper. The coding sessions try to mimic the IOI environment as much as possible. The set of tasks used involves a cross-section of past competition tasks used in national and international competitions. The task topics are selected in such a way that they cover most topics that tend to appear at the IOI.

Based on the total results of this selection camp the four best contestants are selected as the IOI team, four contestants are selected as the Central European Olympiad in Informatics (CEOI) team and six contestants (including the IOI team) are selected for the Czech–Polish–Slovak preparation camp (CPSPC, see below).

For CEOI, it is our usual practice to treat it as a practice competition before the IOI. Thus, last-year students that have already taken part in an international competition are usually not eligible for the CEOI.

2.2. *Correspondence Seminar*

For more than 25 years, the students at Comenius University in Bratislava organize a correspondence competition for secondary school students. The original motivation for starting this competition was the lack of qualified informatics teachers in places other than the largest three or four cities. In this way, all talented students are able to acquire extra-curricular knowledge, references to study materials, etc. And last but not least, the correspondence seminar plays a vital role in making these students interested in computer science and enabling them to choose a career in this field.

After the 25 years have passed, we have to sadly conclude that the main motivation behind having the correspondence seminar is still here. With the level of salaries at secondary schools, the number of good informatics teachers decreases at a steady pace. Thus the correspondence competition still plays a vital role in preparing our talented students for the IOI.

The competition consists of four series spread throughout the year. There are three different categories; we can roughly describe them as “beginner”, “ordinary”, and “extreme”. The last category had roughly 15 participants in the last year, and almost all contestants that qualified for this year’s selection camp were previously solving tasks of this category.

In each series, contestants in each category are given several tasks to solve. The contest mimics the home round of the national Olympiad. The difference is that the contestants receive their evaluated solutions back along with personal comments from the evaluators.

2.3. *Correspondence Seminar Camps*

Twice a year a camp is organized for about 30 best participants of the correspondence seminar. This camp takes about a week of time. An usual day consists of two series of lectures in the morning, and of sports, games, competitions (both scientific and other), and team-building activities during the rest of the day. The lectures are mostly focused on efficient algorithm design and implementations. Most of them are prepared by undergraduate university students who organize the camp, but usually there are several guest lectures held by our university lecturers. In each series of lectures two lectures of various difficulties run in parallel, to accommodate different skill levels of the participants. Much of the camp’s program is focused on presenting computer science as a fun and interesting topic, and on building a community of young people interested in it.

You can find more on the importance of this community building in (Forišek and Winczer, 2006). Refer to (Bell *et al.*, 2002) for an idea of how many of the camp’s activities look like, and to (Verhoeff, 1997) for a general overview of related topics.

2.4. *Programming Hatchery*

In the recent years I developed the following train metaphor: Computer science as whole and programming competitions in particular, can be regarded as a train. The train has

already left the station and it is still speeding up. For new passengers, catching it and getting on is becoming harder and harder. But is there any way to help them?

Actually, there is. Use a second train. Make it slow enough for them to get on and then speed up and allow them to jump to the first train.

This is exactly what we tried to achieve by starting an e-learning portal called Programming Hatchery. The first "chapter" of this portal is a gentle introduction to the area of programming contests. Subsequent chapters that are opened later offer a variety of graduated difficulty task series accompanied by study texts. The entire portal is available in Slovak, thereby compensating for the lack of good Slovak textbooks. Most of the study texts were adapted from the knowledge accumulated during the years by the organizers of the correspondence seminar. The students are given lots of valuable resources at their disposal, while being able to choose their own pace of progress. To date, this portal has got 655 registered users.

3. International Cooperation

3.1. *Preparing Problems for the National Olympiad*

Since 1993, when Czechoslovakia was split into Slovakia and the Czech Republic, we maintain an intensive cooperation with the organizers of the Czech national Olympiad. Both contests share the same problem set. For half of the years this problem set is prepared by one nation, for the other half by the other nation.

Thanks to this cooperation we have more time to gather and prepare more interesting competition problems.

3.2. *Czech–Polish–Slovak Preparation Camp*

Since 1999 the three nations mentioned in the title organize a joint week-long camp for their best students. Usually six contestants from each country take part in the camp. The organizers prepare a set of tasks for four IOI-like competition days. The camp usually involves some sports in the afternoons, and one day-long trip.

As an interesting bit of trivia, the main reason behind the order of country names in the camp's title is that the current abbreviation CPSPC is a palindrome.

3.3. *Cooperation with the Swiss Olympiad*

The Swiss Olympiad is a much younger contest than most other national Olympiads, currently it had 12 years. To be able to keep up with the more established contests, the Swiss Olympiad started several new activities to prepare their contestants. In some cases the Slovak Olympiad organizers volunteered to help. In the last year, there were two jointly organized events.

The first of these events was a Swiss Olympiad winter camp in Davos. Roughly 20 participants of the Swiss Olympiad were invited to this event, along with 3 participants

from Slovakia. The camp organizers were both from Switzerland (two) and from Slovakia (three, one of them currently at ETH Zurich).

The second event was that four Swiss contestants were invited to take part in the Slovak selection camp – which served as a week-long training for them.

4. Conclusion

We presented the most important activities related to the Slovak Olympiad in informatics. The main point in preparing our contestants for the IOI is in giving them a strong theoretical background, the ability to reason about algorithms and to express their thoughts, and only then we focus on the implementation part of the competition. We try not to forget that the competition is not a goal, but just a means to make computer science more popular, and to motivate talented students to study it.

5. Appendix A: Example of a Theoretical Task

5.1. *Problem Statement*

"*The fictional mountain*"

Michael and Joseph were planning a trip. Each of them took his own map and started to examine the track they agreed upon. On each of the maps, some points of the track had their altitudes displayed. After a while, Michael claimed: "That's interesting. If I only consider the altitudes on the track, it seems that we will only be going over one mountain – first up, then down." "That's nonsense!" answered Joseph. After a while they found out the reason: Michael's map was less precise, and it was missing some of the altitudes displayed on Joseph's map.

Your task is to write a program that reads the list of $N < 100000$ altitudes written on Joseph's map, and computes how many of them could have been shown on Michael's map as well.

EXAMPLE.

Input: $N = 12$, altitudes: `112 247 211 209 244 350 470 510 312 215 117 217`

Output: 9

Explanation: E.g., Michael's map could have contained these nine altitudes:

$$112, 211, 244, 350, 470, 510, 312, 215, 117.$$

Worse solutions:

* for 7 points out of 10, solve the task with the constraint $N < 1000$,
* for 4 points out of 10, solve the task with the constraint $N < 20$.

5.2. *Solutions*

This task admits multiple correct solutions with various degrees of efficiency. The simplest way is to check all 2^N subsets of the given numbers, and for each of them check whether it represents a mountain. This leads to a solution in $\mathrm{O}(N \cdot 2^N)$ time.

The efficient solutions are based on the concept of the longest increasing subsequence (LIS). Note that the mountain is a subsequence that contains first an increasing, then a decreasing part. The correct solution is to compute the length of the LIS ending at each element of the sequence, and the length of the longest decreasing subsequence beginning at each element of the sequence. (The latter is just the LIS in the reversed sequence.) Having these lengths, the answer can be computed in $\mathrm{O}(N)$ time. The lengths can be computed using one of the known LIS algorithms in either $\mathrm{O}(N^2)$, or $\mathrm{O}(N \cdot \log N)$ time. For discussion of these algorithms see (Cormen *et al.*, 1989).

There are other, less efficient solutions – e.g., one can use brute force to pick the top of the mountain and for each top compute LIS in each of the parts from scratch. This leads to a solution running in $\mathrm{O}(N^2 \cdot \log N)$ or $\mathrm{O}(N^3)$ respectively, depending on the LIS algorithm used.

6. Appendix B: Example of a Computation Model Task

6.1. *Rationale*

For the selected computation models we do not expect the contestants to have any prior knowledge. The study text accompanying the problem statements is supposed (and designed) to be their first introduction to the area in question. The text is written in such a way to be accessible to (talented) secondary school students – the model is explained as simply as it gets, parts that won't be used in tasks are omitted, etc.

The contestants' main task is to understand the study text. They have to be capable of understanding the given model, and forming a suitable mental abstraction. The tasks used in the home round usually serve as a kind of a verifier – the contestant who understood the model correctly should be able to solve them without any problems.

Occasionally, the home round task statement also helps to understand the model by pointing out some interesting concept that will be referenced in subsequent rounds.

In the following sections we will show an example justifying the claims above.

6.2. *Past Models*

In the past years, these topics from theoretical computer science were used as background for our competition tasks: deterministic Turing machines, Wang tiles (Wang, 1961), alternating Turing machines (Chandra *et al.*, 1981), Markov's registry machines, D0L-systems, a-transducers, reversible computers, and many others.

6.3. *Example Study Text (partial)*

In this section we present a part of the study text for alternating Turing machines – a computational model with non-deterministic decisions of two types, corresponding to a "parallel and" and a "parallel or" on a massively parallel machine. Note that our formulation makes the topic slightly less formal, but much more accessible.

"*The Parallelizer*"

Beyond seven mountains and six valleys, the inventor Kleofas managed to construct a wondrous machine. He decided to call it the Parallelizer. At the first glance, it was just an ordinary computer. However, there was a small twist. Under some special conditions, the Parallelizer was able to run several copies of its program in parallel, without slowing down.

Kleofas designed a simple programming language for his new machine. The language looks just as Pascal for ordinary computers, there are only a few differences. First, we don't have the function `Random`, thus for each program its computation and its result are uniquely determined.

Upon termination, each program will return an exit code: either zero (meaning that its run was successful) or one (meaning that it was not). For clarity, we will have special commands `Accept` (terminate with exit code zero) and `Reject` (terminate with exit code one).

The final change will be the commands allowing us to make parallel computations: `Both(x)` and `Some(x)`. The command `Both(x)` stops the current program and creates its two identical copies. In the first one the variable x is set to 0, in the second one it is set to 1. Both copies are executed in parallel. If both of them terminate successfully, the original program is resumed. The command `Some(x)` works in the same way, with the difference that the original program is resumed as soon as at least one copy terminated successfully.

(The rest of the study text contained a more precise definition of these two commands, and two simple example tasks with solutions and commentary on the program's execution.)

6.4. *Example Task*

Given are two strings *haystack* and *needle*. Write a program for the Parallelizer that will successfully terminate if and only if the string *needle* is a substring of the string *haystack*.

6.5. *Solution*

While really simple, the optimal solution of this task contains two important concepts: using `Some` to make a parallel search (i.e., to emulate non-determinism, but this concept is never referenced explicitly), and using `Both` for parallel verification. In both cases, the lesson learned is that $\mathrm{O}(N)$ things can be done in $\mathrm{O}(\log N)$ time by repeated forking.

Many contestants were able to define functions `Exists(x,N)` and `Forall(x,N)` that do the same as `Same` and `Both` for x in the set $0, ..., N-1$. Using these functions a solution to the given task fits on a single line. Both functions were introduced in the sample solutions of the home round, and used in the next rounds as a building brick.

References

Bell, T., I.H. Witten and M. Fellows (2002). *Computer Science Unplugged*. A special version of the book available (retrieved Jun 2007) at
`http://www.google.com/educators/activities/unpluggedTeachersDec2006.pdf`

Boersen, R., and M. Phillipps (2006). Programming contests: two innovative models from New Zealand. Presented at *Perspectives on Computer Science Competitions for (High School) Students*. Retrieved Jun 2007 from
`http://bwinf.de/competition-workshop/RevisedPapers/1_BoersenPhillipps_rev.pdf`

Boersen, R. (2006). Inclusive programming contests = Inclusive problem sets. In V. Dagiene and R. Mittermeir (Eds.), *Information Technologies at School*, pp. 535–544.

Chandra, A.K., D.C. Kozen and L.J. Stockmeyer (1981). Alternation. *Journal of the ACM*, **28**(1), 114–133.

Cormen, T., C. Leiserson and R. Rivest (1989). *Introduction to Algorithms*. MIT Press.

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

Fisher, M., and A. Cox (2006). Gender and programming contests: mitigating exclusionary practices. *Informatics in Education*, **5**, 47–62.

Forišek, M., and M. Winczer (2006). Non-formal activities as scaffolding to informatics achievement. In V. Dagiene and R. Mittermeir (Eds.), *Information Technologies at School*, pp. 529–534.

Pohl, W. (2006). Computer science contests for secondary school students: approaches to classification. *Informatics in Education*, **5**, 125–132.

Verhoeff, T. (1997). The role of competitions in education. Presented at *Future World*: *Educating for the 21st Century*. Retrieved Jun 2007 from
`http://olympiads.win.tue.nl/ioi/ioi97/ffutwrld/competit.pdf`

Wang, H. (1961). Proving theorems by pattern recognition. II. *Bell System Tech. Journal*, **40**(1), 1–41.

M. Forišek is currently a graduate student at the Comenius University in Slovakia. He received a master's degree in computer science from this university in 2004. His research interests range from theoretical computer science to education of mathematics, informatics, and algorithms. He is an elected member of the International Scientific Committee of the International Olympiad in Informatics. For many years he has been an active organizer of national and international programming contests, including several years of the Internet Problem Solving Contest (IPSC) and Central European Olympiad in Informatics 2002. As a student he was a very successful participant in programming contests, he was awarded gold medals both at the IOI and at the ACM ICPC World Finals.