Olympiads in Informatics, 2012, Vol. 6, 3–20

The Role of Contests in Changing Informatics Education: A Local View

Giorgio AUDRITO[1], G. Barbara DEMO[2], Elio GIOVANNETTI[2]
[1]*Dipartimento di Matematica, Università di Torino*
via Carlo Alberto, 10-10123 Torino, Italy
[2]*Dipartimento di Informatica, Università di Torino*
corso Svizzera, 185-10149, Torino, Italy
e-mail: giorgio.audrito@gmail.com, {barbara, elio}@di.unito.it

Abstract. Changing the teaching of informatics in secondary (and primary) education is in many countries difficult to achieve, owing to several reasons: the natural tendency of institutions and people to stick to well-established practices, a distorted view of informatics prevailing in society and among those who take the decisions, the specific fact that in several school systems the teaching of informatics is largely committed to teachers of other scientific disciplines. We describe the Italian situation and our experience in Turin where we have organised training stages for the Olympiad in Informatics and supported the organisation of other informatics contests. On its basis we argue how the different kinds of competition can help to change informatics education in Italy, by making problem solving and programming play a central role.

Key words: competitions, problem solving, programming in education, data structures, algorithms, programming techniques.

1. Introduction

In the last few years the need for a change in curricula, learning objectives and teaching techniques of computer-related disciplines in (primary and) secondary education has emerged in the public debate in a number of countries. The recent speech of the British Secretary of State for Education Mr. Michael Gove at BETT 2012 (BETT is the annual British Educational Training and Technology show) is a good example of governmental awareness on the subject. As is now widely recognized, time has come to replace the teaching of the so-called ICT with a real computer science curriculum, with emphasis on the word "science"; if we adopt the term traditionally used for computer science in continental Europe, but now also in UK, we can say that we must replace ICT with Informatics, or at least to promote a change in that direction.

In Italy too there has been a great discussion involving the academic and industrial computer scientists' communities, particularly during the months when the recent reform of secondary education was worked out, and eventually came into effect in September 2010. The reform, though introducing some positive novelties, is still far from satisfactory from the above point of view; nevertheless, the hope is that the general framework provided by the law may still be filled with some contents of the desired type.

In May 2010 the national associations of academic computer scientists of Science Faculties and Engineering had published a manifesto for informatics in the reform of secondary school (CINI, GRIN, GII, "Manifesto sull'Informatica nella riforma della scuola superiore", 2010), which is still the basis for the present efforts to change the way informatics is (not) taught in most Italian schools. It distinguishes three different, though related, meanings of the word "informatics":

1. *operational*: the set of all software and hardware objects;
2. *technological*: the technology that allows to make such objects;
3. *cultural*: the scientific discipline that is the foundation of such technology and thus makes it possible.

Informatics in the first meaning is what is sold in shops and department stores, and is for most people the main meaning of the word. The technological aspect is usually recognized as important, though often associated with computer geeks. As for the scientific aspect, few are aware even of its existence; also some scientists from the more traditional disciplines doubt that informatics is a separate science in its own right and not, at best, a (trivial) part of mathematics.

As has been repeatedly stated by many influential computer scientists, non-technical secondary education ("liceo" in Italy) should privilege this third aspect, giving informatics as a science the same dignity and importance as mathematics or physics, because of the essential contribution it can carry to a person's intellectual and practical formation. For example, in a world increasingly ruled by algorithms of the most diverse kinds, the very notion of algorithm and the basic principles of algorithmics should be familiar to everybody with a sufficient level of culture, exactly as any pupil is supposed to become familiar with the basic laws of physics.

On the contrary, in this kind of schools, like in society, the view of informatics as restricted to the first or at best the second aspect is still largely prevailing, also owing to the scarcity of teachers with a computer science degree (graduates in informatics tend to find jobs in industry or services and are not traditionally oriented to school-teaching careers).

To change school curricula has always been hard, as already observed by Papert many years ago (Papert, 1997); unfortunately, this is still true nowadays, particularly in informatics, for several reasons. On the one hand, as we have observed above, among those who according to the rules of the reform are going to teach informatics many are teachers of other disciplines, without specific competences in the subject. On the other hand, the computer science community can hardly compete with much older scientific communities, such as those of mathematics and physics, which have a long tradition in didactic research and a well established and organized presence and curricula in secondary education.

In this situation, the idea of using competitions as a lever to introduce in our schools "sideways" what is hard to be obtained directly, has gained in the latest years a growing consensus. Starting from an increased interest by many institutions in the traditional Olympiad in Informatics, several new competitions have been introduced over the years in many countries, and recently also in Italy: the Olympiad in Problem Solving, Kan-

gourou, Bebras, the ScratchDay, ZeroRobotics, RoboCUP, etc. Like the Olympiad in Informatics, they are organized at different territorial levels, and are therefore able to reach a great number of teachers and students. How to make them most effective for the required change of informatics education is a subject of discussion, to which we would like to give, in the following, the small contribution of our experience.

In the next section we briefly describe the situation of informatics teaching in Italian secondary schools, and in the short Section 3 we comment on a natural role of competitions. In Section 4 we focus on the Olympiad in Informatics, while Section 5 describes other competitions, less aimed than Olympiad at discovering and cultivating excellence. In Section 6 we comment on the central role of programming induced by competitions. In Section 7 we draw some general conclusions.

2. Informatics in Italian Secondary Schools: Short History and Perspectives

The Italian system of secondary education consists of two levels, the lower secondary school or "Scuola Media", 11–13, and the upper level, 14–19, which is divided into three different channels: the general channel or "liceo", analogous to the French "lycée" or German "Gymnasium", the technical school type, and the vocational school type. The general channel, in turn, is divided into the scientific and the humanistic sub-channels, while the technical and vocational channels are obviously divided into the various technological specialisations.

Differently from most other countries, the subjects and their weights in numbers of hours are rigidly fixed within each kind of channel or sub-channel, with no possibility of free choice of optional subjects on an individual basis. However, in the pre-reform system a great number of variants had developed over the years within each established kind of school, and it had eventually grown into an almost inextricable but extremely wide spectrum of education programmes, thus putting a remedy to the formal rigidity of each of them. Many of these programmes were largely appreciated by pupils and families. The reform system has pruned the number of alternatives, in some cases by taking as a new standard for a channel what previously was a sub-sub-channel that had proved to be particularly successful; in other cases it has cut possibilities in a way that was strongly criticised by teachers.

Informatics as a distinct and autonomous subject in Italy has been traditionally present only in the technical schools for informatics, and – with a much lower importance – in other technical types of school. On the contrary, it has been almost completely absent in the general channel or "liceo".

As a matter of fact, the so-called National Programme for Informatics ("Piano Nazionale Informatica" or PNI) was launched in non-technical secondary education almost thirty years ago, but it was a programme mainly conceived by mathematicians as instrumental to a strengthened teaching of mathematics, and committed to mathematics teachers, whose competence in informatics was usually poor. Nevertheless, in the pre-internet era, when few or no software aids to mathematics teaching were available, PNI

meant introduction to programming, usually with Pascal, which was certainly, for the first time in that kind of schools, a pupils' (and teachers') exposure to informatics proper. Over the years, however, when tools like "Derive" or "Cabri" became of universal use, the need of even a limited capacity of programming in a general-purpose language disappeared, and informatics reduced to the use of specialized tools. Also the widespread offering, in schools, of the ECDL (European Computer Driving Licence) certification – though initially valuable – strongly contributed to the vision of computer science as the mere usage of software tools.

With the reform not much has changed with respect to informatics. The new "liceo scientifico" has two separate curricula: the first one, called "traditional", is similar to those previous special programmes that exhibited a strengthening of mathematics and scientific subjects, and like those includes not well-defined elements of informatics as part of the mathematics syllabus; the second curriculum, called "Applied Sciences" ("Liceo delle Scienze Applicate'), is also similar to a former programme ("Liceo Scientifico Tecnologico"), and features no Latin, a lighter weight of philosophy and history, a still increased weight of all the traditional scientific subjects, in particular physics and biology, and informatics as a separate subject, whose teaching is allowed to mathematicians and physicists besides computer scientists.

3. Algorithms and Creativity: A Role for Competitions

Teachers of scientific disciplines (both in secondary and in higher education) have witnessed in the last decades a growing request – from students and pupils – of precise and detailed "recipes", i.e., algorithms, for solving the problems and doing the exercises that are submitted to them. Many students seem to implicitly ask for a precise catalogue of all the possible problem schemes, with a solving algorithm for each of them: all they wish to learn is to execute such algorithms after putting in the initial data, as human computers. This parallels, not surprisingly, the general evolution of society where, as observed in Section 2, algorithms are going to permeate every aspect of life, from the famous financial algorithms blamed of being a cause of the present crisis, to medical procedures and protocols, to bibliometric methods for the evaluation of research, etc.

At the same time, and somewhat paradoxically, there is among pupils and students a widespread search for creative activities outside school, as in the adult society there is a search for creative jobs in contrast with routine work (the word "routine" used to indicate, in the early days of computer science, a program procedure). In particular, pupils of non-technical schools who approach informatics for themselves are mostly driven by a specific goal: to build their own computer game. Also our average undergraduate student of informatics, when it comes to the obligatory final stage with industry seems to show, according to the external tutors' reports, autonomy, originality and creativity, i.e., exactly those qualities they were lacking during the previous study years. It is presumably a problem of motivation, and of personal adequacy to the level of difficulty.

So, it is also not surprising that a number of pupils in any kind of school are attracted and strongly motivated by competitions, where they can test themselves on more cre-

ative tasks than the ones usually assigned in normal schoolwork. The challenge is how to exploit competitions for the needed change in education systems.

4. The Olympiad in Informatics

4.1. *General Characteristics and Present Trends*

The Italian Olympiad in Informatics (Olimpiadi Italiane di Informatica, OII) has, like their analogous in other countries, the ultimate aim of selecting the national team for the International Olympiad in Informatics (IOI). The whole selection process is organized on three different levels: a school level, which selects a fair number of pupils from every participant school, a regional level, which in each region selects a small number of pupils out of the previous phase, and finally the national level which selects the national team. For a detailed analysis, with statistical data, of ten years of Olympiad in Italy, see Italiani (2010) and Scarabottolo (2011).

It is interesting to single out the different characterizations of the three levels, and to recognize through the years some evolution trends, distinct for the different levels. First of all, we must remark that school-level tests also contain logical and mathematical problems and exercises besides programming problems, so as to be able to select potentially good candidates for the following phases even in that majority of schools where informatics is not present in the curriculum. The regional and the national tests, on the other hand, consist exclusively of programming problems, to be solved in the official languages of IOI (C or C++, besides Pascal).

At the regional level the difficulty, after a considerable increase in the years from 2000 to 2005, has settled on a kind of programming problems that often require good competences on the use of recursion and/or on graph algorithms. However, since at this level there are no execution time limits for the submitted solutions, often problems may be solved by brute-force methods, thus discouraging algorithmic creativity.

At the national level, on the other hand, after an analogous, though less pronounced, increase of difficulty, in the last years the extent of algorithmic competences has been narrowed, with the intent to reduce the advantage of informatics technical schools with respect to the other schools. So in the last editions dynamic programming problems, backtracking, minimum path or spanning tree or flow graph problems have no longer turned up. The effect of this has been a shift of the difficulty towards the mathematical aspect, with the introduction of types of problems whose solutions have little algorithmic content but are based on the results of preliminary non-trivial mathematical reasoning, often of combinatorial nature.

As an example of the latter trend we may cite the problem "A superheroes' school" of the national 2010 contest, where one requirement is the organization of a tournament with $\mathbf{2}^n$ participants in $2^n - 1$ rounds, so that each superhero meets every other: the solution lies in the fact that at the **k**th round the superhero **j** met by the superhero **i** is given by **j = i XOR k** (bitwise exclusive-or).

As one can see, these trends, which in some way go against the characterization of computer science as an autonomous discipline, stem from the absence of computer programming and algorithmics in Italian general education, and are therefore a symptom of the very disease they should help to treat.

4.2. *The Role of the Olympiad in Italian Secondary Education*

Informatics Olympiad, like all homologous initiatives, e.g., the Mathematical Olympiad, seeks individual excellence, which is extremely important, but one may wonder what their impact is on the general level of education, in the same way as one may ponder on the difference between searching for prospective sport champions and ensuring the possibility of a healthy sport practicing for the greatest number.

Also, the Olympiad is naturally centred on algorithmics, which as remarked above is certainly at the core of computer science, but does not exhaust it: other areas are at least as important, for example, abstraction and modelling of reality, concurrent and interactive programming, multidisciplinary informatics, just to name a few.

Algorithmic problems, however, are very good examples of tasks requiring a disciplined creativity, and may therefore be an answer to pupils' demands, at least in principle. As a matter of fact, we explain in our courses that there cannot exist an algorithm which, given a specification, i.e., the description of a computational problem, is able to derive a solving algorithm for it. The big issue is how to present such problems in an attractive and motivating way, not just as another kind of abstract exercises completely disconnected from reality.

The imaginative language in which Olympiad problems are traditionally formulated is a possible help, though to a university professor it may look a bit childish, or a useless and misleading disguise of a simple problem. Actually, at a more thorough reflection, this very disguise has a precise educational value for prospective computer scientists, who in their professional activities are often faced with the task of extracting from the customer's requirements, expressed in a concrete and often confusing language, the underlying abstract computational problem (which may sometimes be a well known problem with a known solving algorithm).

The organisation at three levels (school, region, country), common to most kinds of competitions, is useful since each participant may thus find a level roughly adequate to her/his abilities, instead of having to achieve a compulsory – and equal for all – level of competence, as in ordinary school learning. The drawback is that pupils that have poor performances in school contests or even in regional contests may feel frustrated and may develop a feeling of rejection for the whole discipline, even though, in some cases, their school proficiency is good. However, this is not necessarily a bad thing, since it may help pupils to assess their capacities and talents more precisely.

On the other hand the Olympiad experience, either in mathematics or in informatics, certainly has, for successful participants, a strong impact on the choice of the university course: they tend to enrol at Mathematics or Informatics respectively, while otherwise they would have probably chosen Engineering, especially in a city like Turin where the

renowned Polytechnic University also attracts students that would be better suited to other courses. This clearly emerges from the answers to the questionnaires submitted to pupils at the end of Olympic training stages and from the enrolment data in the following years.

4.3. *The Training Stages at the University of Turin*

The Department of Informatics of the University of Turin, among its various activities of popularization and promotion, in the last four years has organized a training stage for the regional contest, addressed to the pupils got through the school competitions.

Since in many school contests programming may play a minor role, we found ourselves facing a great disparity of competences: from people who knew a great deal of algorithms and data structures, to boys[1] who hardly know how to write a program, or were lacking basic concepts like recursion. So our challenge was to take also the latter, in a few days, to a sufficient competence level for the regional tests.

Each day was organized in two parts: the morning devoted to a small number of lectures on programming or algorithmic techniques, for example recursion, loop design possibly through invariants, C++ STL, exhaustive search, greedy algorithms, graph visits, etc.; the afternoon in the computer lab, devoted to problems where the techniques explained in the morning must be applied. In the lab, each student tries individually to solve at his computer the problem proposed, with the teacher's active assistance.

In this activity the experience of one of the authors (G.A.), former medalled participant at IOI and now officially among the trainers of the national team, was of great value for devising exercises, while his direct knowledge, both passive and active, of competition mechanisms allowed him to help pupils with simple advice and tips.

The stage was supplemented by a virtual environment on a Moodle platform of the Department of Informatics, containing the presentations, the proposed problems, discussion forums, assignments, etc., as a means to continue the training and to keep a link with the department. The stage obviously involved a small sample of all the secondary school pupils in Piedmont; nevertheless, we think the method could be extended to learning informatics even independently from competitions.

4.4. *Two Problem Examples*

As a partial illustration of the techniques used in the stage, we report here a couple of problems among the ones we have proposed.

The first problem, in Fig. 1, is an example of how the greedy technique can be applied for finding a quick solution that, although not correct, may give the right result in many simple cases, and may therefore – according to the Olympiad's rules – ensure a non-null score with a little effort. The problem also allows to show how a correct solution can often be reached by means of the slightly more sophisticated technique of recursion with *memoisation*, i.e., (the recursive form of) dynamic programming.

The text is simple and the specification of the required result is easy to understand.

[1] Girls were unfortunately completely absent, see also Scarabottolo (2011).

In the new amusement park they want to build a copy of the tower of Ouchnoi, but due to budget restrictions they only have at their disposal a stock of rectangular stone slabs, and all the slab sides are different from one another! In building the tower, each slab (except the base, of course) must rest on a bigger slab, i.e., on one whose both sides are greater, possibly through a rotation by 90 degrees. Help the workers to build the highest possible tower.

Fig. 1. The tower of Ouchnoi.[2]

At each step, if the new slab is larger (in both dimensions) than the present base, add it as new base under the present one; if it is not, check whether it is still larger than the second slab of the tower: in such case remove the base and replace it with the new slab, because in this way you get a tower of the same height but with a smaller base, which is of course an advantage; otherwise simply discard it. Because of the last clause, this strategy is unable to find the maximal height for any input.

```
...
Slab base, second;
base = slabs[0];
int maxHeight = 1;
second.x = second.y = 0; // dummy null slab put on top
for (int i=1; i<N; i++){
  if (slabs[i].y > base.y) { // if the slab is larger than the base
    second = base; // the old base becomes the second slab ...
    fout « base.x « "," « base.y « endl;// ... definitively
    base = slabs[i]; // the new base
    maxHeight ++;
  } // otherwise check if you can replace the base
  else if (slabs[i].y > second.y)
    base = lastre[i];
}
```

Fig. 2. The tower of Ouchnoi: incorrect greedy algorithm.

Each slab is characterized by two integers, the x-side and the y-side, and in reading the slab list from input the slabs may all be saved with, say, the x-side as the greater. Then the first action that usually comes to mind consists in sorting the slab sequence w.r.t. to one side, for example x.

At that point we have to find the maximal non-consecutive subsequence that is also sorted w.r.t. the other side. To the more expert trainee this task may remind problems like the longest common subsequence, which may be solved by dynamic programming techniques. Less prepared pupils, like the ones in our class, tried a much simpler greedy method: build the tower from the top to the base by successively taking the slabs previously sorted in ascending order with respect to x, as in Fig. 2.

The reason why the naïve greedy strategy is not correct is that there are cases where consecutively removing more than one slab in order to accommodate a smaller base is necessary to allow building later a higher tower. For example, if the sorted sequence of slabs is [(6,4), (7,5) (8,1), (9,2), (10,3)], the greedy strategy builds a tower of height 2 with the first two slabs and then discards all the following slabs, since they have a smaller

[2] In Italian "Ahinoi", which sounds similar to "Hanoi", but literally means "ouch us", "poor us".

The highest tower having the kth slab as top slab is 1 plus the height of the highest tower having a larger slab as top slab; the candidate larger slabs are those that follow – or, in reverse order, precede – the kth slide in the sorted input sequence.

```
...
int maxHeightFromTop(int k) {
  int maxHeight = 1;
  int top_y = slabs[k].y;
  for(int i=0; i < k; i++) {//assuming slabs sorted in reverse x-order
    if(slabs[i].y > top_y) {
      int max = maxHeightFromTop(i) + 1;
      if (max > maxHeight) maxHeight = max;
    }
  }
  return maxHeight;
}
...
```

Fig. 3. The tower of Ouchnoi: recursive solution.

y-side than the base; on the other hand, by removing from the tower the first two slabs and starting anew from the third, one is able to build a tower of height 3. It is therefore natural to look for a recursive solution, i.e., a recursive function which returns the height of the highest tower having the kth slab as top slab, as described in Fig. 3.

Pupils rapidly discovered that, while for small inputs the recursive algorithm returns the right solution, with a greater number of slabs the execution time rapidly grows beyond any reasonable limit.

They were then helped to reflect on why this happens, and thus came to the obvious conclusion that the cause is the exceedingly great number of recursive calls that re-compute solutions of the same sub-problems, like in the paradigmatic example of the naïve recursive version of the function that computes the nth Fibonacci number. So the *memoisation* was added (using a vector) and an efficient solution, partially described in Fig. 4, was finally found.

Problems directly or indirectly involving graphs are rather common in the Olympic competitions, and are also a kind of problem that is usually a fun for pupils. An example from the ones proposed in our training stage was, in its abstract form, the following: given a weighted directed acyclic graph, and given a source vertex *A* and a sink vertex *B*, find a path from *A* to *B* such that the weight of the heaviest of its edges is minimal. Its formulation in Fig. 5 translated the acyclicity condition into the metaphor of downhill-only paths, and interpreted weights as danger or risk values.

A similarity with the least path problem immediately suggested an *à la Dijkstra* approach: keep for each vertex a *distance*, representing the global risk of the best path to that vertex, and visit the graph possibly updating the distances. The distance is, as in Dijkstra's or Prim's algorithms, initialized to zero for the start vertex and to infinity for the others. However, differently from those algorithms, breadth-first visit does not work, since the temporary best path to a vertex becomes definitive only when all paths to that vertex have been considered.

An array is added to keep the computed results of the recursive calls.

```
...
int maxHeightMemo[MAX_N];
...
int maxHeightFromTop(int k) {
  if (maxHeightMemo[k] > 0) // if the value has been computed before,
    return maxHeightMemo[k]; // return it without re-computing it
  int maxHeight = 1;
  int top_y = slabs[k].y;
  for(int i=0; i < k; i++) {//assuming slabs sorted in reverse x-order
    if(slabs[i].y > top_y) {
      int max = maxHeightFromTop(i) + 1;
      if (max > maxHeight) maxHeight = max;
    }
  }
  // memo(r)izes the result and returns it
  return maxHeightMemo[k] = maxHeight;
}
...
```

Fig. 4. The tower of Ouchnoi: recursive solution with *memoisation*.

Tom runs in the Boxcar Race of the Alps where each participant, in a self-made motorless vehicle, has to reach the base of a mountain from an upper starting point without destroying or damaging her/his rudimentary vehicle. They can choose any downhill path in a network of roads and trails. Tom has a map of all the downhill pathways with their intersections and has annotated with a risk value every stretch between two intersections. The global risk of a path is the maximum risk of the stretches composing the path. Help Tom to find the least risky path."

Fig. 5. Boxcar race.

In the morning lectures on graphs we had presented a number of basic graph algorithms and techniques: some pupils were then able to see through the "downhill" metaphor an altitude order, and recognize it as a topological order. As a matter of fact, by visiting the vertices in a topological order we are ensured that when we visit the ith vertex we have already examined all the edges to that vertex, and its distance is therefore definitive; we only need to check and, if necessary, update the distances of the adjacent vertices. The topological sorting was performed in advance by the algorithm presented in the lectures.

We found that the fancy formulation of the problem, owing to its concrete character, was for our pupils easier to understand than the abstract formulation which, on the contrary, is usually easier for teachers. At the same time, knowing how to obtain the topological order without having to reinventing it from scratch was of course important, and contributed to convince the trainees that study is useful.

First perform (through depth-first visit) the topological sorting on the graph, and let *sortedVertices* be the result array; then examine the vertices in that order, and for each vertex u consider all its adjacent vertices v: for each v, the global risk of the path from *start* to v via u is the maximum between the global risk from *start* to u and the risk of the edge from u to v; if the global risk of the new path is lesser than the previously computed global risk from *start* to v, update the path to v and the global risk to v. In the code fragment below we only show the updating of the global risk.

```
...
// the vertex 0 is the start vertex
topologicalSorting(0);
globalRisk[0] = 0; // is the distance: the start vertex has distance 0
                   // the other vertices have distances initialized to infinity
for (int i = 1; i < N; i++) globalRisk[i] = INFINITY;
for (int i= N-1; i > 0; i-) {
  int u = sortedVertices[i];
  for (int j=0; j < degree[u]; j++) {
    int v = adjacent[u][j]; // for each vertex v adjacent to u
    // evaluate the global risk in the path to v via u
    int vRisk = max(globalRisk[u], risk[u][j]);
    // if lesser, update the global risk to v
    if(vRisk < globalRisk[v]) globalRisk[v] = vRisk;
  }
}
...
```

Fig. 6. Boxcar race: a solution.

5. Other Informatics Contests in Italy

Olympiad in Informatics, with its search for excellence, cannot alone serve the purpose of stimulating the interest in computer science. In the last years some other informatics contests have been introduced in Italy at various levels. Among them, Olympiad in Problem solving and Kangourou Informatics already involve large numbers of participants from throughout the country; the Scratch Day, celebrated every year in schools and institutions throughout the world, and joined last year by a Piedmont's technical school with a local initiative, has become this year a national event, the Italian Scratch Festival. In addition, there is a number of educational robotics yearly meetings and competitions, where being able to program robot behaviours is often a key element for success in the competition.

Some of the initiatives emphasize problem solving as a preliminary ability for programming, and are based on the observation that technicalities of actual coding may often obfuscate the clarity of the algorithmic core (as anybody who has written a non-trivial program in a real language knows). At the same time, programming remains the end to which, in the intention of the proponents, these initiatives should ultimately lead: programming is present, though with small exercises, in the upper competition levels.

In the following, after a short account of the Olympiad of Problem Solving and Kangourou, amply described elsewhere, we extend a bit more on the newly introduced Scratch Festival, which – being completely based on a "programming for everybody" paradigm – may act as a driving factor for the introduction of programming-based curricula at all school levels.

5.1. *Olympiad of Problem Solving*

The Italian Olympiad of Problem Solving (IOPS), created – after some years of local experiences – by professor Giorgio Casadei, was organized by the Italian Ministry of Education for the first time as a national contest in the school year 2008–09. It was reserved to pupils of the last year of both primary education and lower secondary education ("scuola media"), but because of the big interest it aroused among the teachers, the following editions were opened also to other year classes: in 2011–12 it globally involved pupils from the last two years of primary school, all the three years of lower and the first two years of upper secondary education, so in the ages from 9 to 16, of course with different features and difficulty levels for each age range. The attendance at such editions reached about 30.000 pupils.

IOPS is based on a methodology of the kind "Computer Science Unplugged" (CSU), i.e., computer science without a computer, but it also aims at making young pupils able to understand and use simple semi-formal languages. As a matter of fact, typical contest problems are in a CSU form, but some exercises require the usage of a pseudo programming language for describing the solution clearly and unambiguously (Casadei and Teolis, 2011).

The specific contribution of IOPS to the change of informatics education is its addressing the lower age ranges with tasks that develop typical informatics abilities in a pure way, without the technicalities of a real programming language. Another essential feature is the mode of interaction with schools, which is not limited to one or two annual events, but continues along the whole school year, with frequent training sessions, intermediate assessments, monthly school competitions, etc., all supported by a dedicated web environment.

5.2. *Kangourou of Informatics*

It is the extension to informatics of a mathematics contest which was originally started in France where it has a well-established and successful tradition. Conceived and organized by a group of professors and researchers of the University of Milan, its features are thoroughly described in Lissoni *et al.* (2008) and Lonati *et al.* (2011). It is addressed to lower secondary schools and to the first two years of upper secondary schools, i.e., to students in the age from 11 to 16.

Characterizing aspects are the game-contest modality with its emphasis on fun, the absence of prerequisites in programming or other specific knowledge, and – not least – the fact that competitions are not among individuals, but among four-person teams, thus giving value to cooperation capacity, which is essential in today's working environments, particularly in computer science, and more generally in today's society.

All these features, and especially the last, should profitably find a place in the reorganisation of informatics education.

5.3. *Italian Scratch Festival*

Scratch, the visual drag-and-drop programming environment for children and young pupils created by the MIT Resnick's team (Resnick *et al.*, 2007), is being adopted, after the reform, by a number of Italian schools in the first year of technical education as an alternative to the ICT syllabus, a sort of ECDL-like general introduction to computers, which remains the first-year standard in most technical schools. In this way introduction to programming is no longer reserved to the informatics specialization, but for the first time it is present in the curricula of every type of technical school from the very beginning.

With Scratch the emphasis is therefore back to programming like in the earliest times of computers in education, but in a completely new non-textual approach which is a lot of fun (with the words of the above cited British education secretary: "Instead of children bored out of their minds being taught how to use Word and Excel by bored teachers, we could have 11-year-olds able to write simple 2D computer animations using an MIT tool called Scratch"). In the higher classes of technical education in informatics, of course, Scratch is replaced by C, C++, Java, etc.

In the year 2010–11, the first one in which the school reform was effective and new curricula had to be established, a group of teachers of the "Vallauri" Technical School in Fossano (Piedmont), who had been (and continued to be) involved in IOPS competitions, adopted Scratch, and published a Scratch-based textbook in Italian (Barbero and Demo, 2011; Barbero *et al.*, 2011).

On the occasion of the international Scratch Day 2011, the "Vallauri" School organized an internal Scratch Day with the participation of all its first-year classes; the teachers' association Dschola, the Piedmont regional consortium CSP for applied ICT research, and the Department of Informatics (Dipartimento di Informatica) of the University of Turin all gave their support to the event and ensured cooperation for it. The Day was a great success, and also attracted teachers from other schools; so this year for the Scratch Day 2012 Dschola decided to organize an Italian Scratch Festival (`http://www.associazionedschola.it/isf`).

The importance of Scratch lies exactly in its being an easy-to-use programming environment that also attracts the less talented or less interested pupils. In an era dominated by multimedia web contents, smartphone apps, sophisticated electronic games, etc., an initial approach to informatics based on a poor programming environment, on a textual programming language and on algorithmic problems of abstract nature, such as finding the maximum of a number sequence, performing a binary search or sorting, is often perceived as an obsolete technology and leads to rejection.

Scratch is a real, though non-textual, programming language, "that makes it easy to create interactive stories, animations, games, music, art, and to share them on the web" (scratch.mit.edu). For that reason it is highly motivating for pupils of different ages, and therefore also easily accepted by teachers who can thus avoid being faced with uninterested and bored audiences.

Its expressive limitations, such as the absence of recursion or of classical data structures do not matter in the context of an initiation to programming, and are counterbal-

anced by its being based on programming concepts like concurrency and message-passing that are of paramount importance in today's computer science. Besides, such limitations have been mostly removed in the BYOB Scratch extension developed at Berkeley University, and in fact some teachers, after Scratch, are going to experiment with BYOB in the next classes (Harvey and Mönig, 2010). Also, a Scratch extension with procedures is announced by MIT for the end of May 2012.

6. Programming as a Core of Informatics Education

Olympiad in Informatics and – with a completely different character – Scratch contests are both exclusively based on programming; the other kinds of competitions are also aimed at developing programming abilities. We think this is highly positive, since we believe, like many authors, that programming should be at the centre of informatics education in schools. The basic fascination of informatics is its creative dimension: inventing an object, the program, which seems to find a life of its own, whatever the level of difficulty and the field of application.

Obviously the goal is not to make of every student a professional programmer, but the not much easier one of turning into reality the idea expressed by Sysło (2011) that "everybody can learn programming", and therefore "computer programming (in any sense) ... should be the competence of everyone".

Actually, while initiation to elementary programming may be made almost deceptively easy, teaching how to write elegant and correct programs for even basic algorithmic problems is difficult, unless one only considers the few gifted students. Giving pupils the solution of a problem, even at the end of their unfruitful efforts, and explaining why it works without telling how to get to it, is not enough. An introspective effort is necessary by the teacher in order to communicate how the solution is reached.

Learning to program requires a delicate balance between creativity and application of rules, between talent and study, as in any non trivial human undertaking. If on the one hand we must contrast the widespread idea that the ability to solve problems may be gained by memorizing a catalogue of suitable recipes to be applied mechanically (i.e., algorithms), on the other hand we must counter the opposite belief that talent is the only thing that matters, and that it is not necessary to study programming patterns and algorithmic techniques.

For example, it may be necessary to show how to use inductive reasoning not only for recursion, but also for iteration, where often a non-trivial loop can be conceived by an invariant-driven design method (it is important, however, that loop invariants are not presented in a formal way but applied as an informal technique: imagine the situation at the generic intermediate step, describe it on paper, possibly with a drawing, and then start coding the iteration step on the basis of it).

In 1971 Wirth drew attention to the role of the chosen programming language in the intellectual process of solving a computational problem: "Program construction consists of a sequence of refinement steps [...] The direction in which the notation devel-

ops during the process of refinement is determined by the language in which the program must ultimately be specified [...]" (Wirth, 1971). Some years later Iverson considered programming languages as "tools of thought", by remarking that their executable character opened the possibility of experimenting with thought (Iverson, 1980), in the same way as physicists always experimented with matter. Although solving problems (or writing procedures) with pencil and paper is a valuable means for countering the thoughtless approach to computing and showing the need of thinking before typing, running and debugging the programme one has designed and coded should remain the final aim.

Modern programming languages are, as Simone Martini writes in Martini (2011), "elegant and sophisticated linguistic tools that informatics has developed to describe effective procedures". Being confronted with an expressive programming language of the present generation should therefore be inescapable for anyone in the course of secondary education. The usage of languages like Python, where indenting is part of the syntax, is then a possible choice, which has been successfully made by A.R. Meo's group in lower secondary school (Martina *et al.*, 2010). Moreover, Python can be combined with a multimedia approach (Guzdial and Ericson, 2010) which makes it more attractive.

Also, since "in informatics, whose primary essence lies in the immateriality of the *linguistic expression* of computation and interaction, really *form is substance*" (italic in Martini's Italian original), even at an early stage the importance of program elegance and "beauty" should not be neglected, starting from a correct indenting, a meaningful choice of identifiers, up to the clarity of the program structure, the avoidance of redundancies, etc. Unfortunately, these aspects cannot be checked by an automatic corrector, and therefore cannot be evaluated in programming contests. In the Olympiad in Informatics, for example, the important thing is to get to write whatever program, possibly quick and dirty, gives the right outputs for as many inputs as possible, if not for all. However, we found that even for the Olympiad these aspects are not completely immaterial, since it is often the case that an "ugly" program is also wrong or inefficient, and bad style is usually error prone.

The problem is that pupils entering secondary schools have very different levels of familiarity with computers and informatics, thus partially contradicting the idea of youth as "digital natives" (Prensky, 2001). The idea has nevertheless some truth, if one considers the fruitfulness usually achieved by an approach based on games and playful environments. In this sense, programming competitions are certainly not the only means to stimulate interest in computer science and to promote an improvement of informatics education. For example, also books like the one by Tomatis (2010), a graduate in CS from our university, on "Mathematics and informatics for crimes" are worth attention. The author shows in it how to apply simple algorithmic or programming techniques to police investigations, criminal cases, etc.; an attractive speaker, he gives lectures and presentations in schools and at the university. Again, the topics play an essential role, for the same reason why detective stories and thrillers have hugely multiplied in the last decades both in literature and in the cinema, becoming a key factor of success.

7. Conclusions

In an educational system like ours in Italy, where computer science is still largely perceived as a merely practical skill and is being taught by teachers with poor informatics competence, competitions may play a key role.

As a matter of fact, creating new teachers through special university degree courses or qualifying the existing teachers through formal educational programmes are processes that may take years if performed in isolation with respect to present school life. On the other hand, the introduction of competitions, starting from the lower secondary classes, and not immediately aiming at excellence, is a means for promoting the desired change, provided some conditions are satisfied.

Firstly, competitions must not be limited to small groups of gifted persons, but have to involve the greatest possible number of pupils and teachers. As a matter of fact, the involvement of present teachers is essential for reshaping syllabi and the way their contents are learnt. Competitions represent a unique opportunity for teachers and pupils to learn together and from one another, working on concrete tasks of common interest.

To this purpose, activities must be initially proposed which are algorithmically elementary but attractive for their character (games, multimedia applications, etc.) and with friendly programming environments. Languages like Scratch, EasyLogo (Salanci, 2010), Python, and contests based on that kind of programming can be useful in such first phase. In the following years more algorithmic aspects should be introduced: a core, roughly coinciding with what is needed for the regional level of the Olympiad in Informatics, should be common to all kinds of schools; other aspects should possibly depend on the different school channels, with or without dedicated competitions. See for example the educational robotics activities described by Barbero *et al.* (2011).

Secondly, and equally important, training for competitions should be integrated in everyday school work, and supported by virtual environments where pupils may interact with schoolmates and teachers and thus find help, discuss problems and solutions, etc., on the model of what is being done with the IOPS. School networks and cooperation with the university are necessary to factorize the efforts and reach a critical mass for the management of such platforms.

This second condition is not easily put into practice in a generalized way: even the best-promoted experiences have only reached well-definite groups of schools and pupils. Extending them to the whole school population would require a qualitative leap that seems hard to perform. A greater commitment is needed by the university, whose support should be more continuous than usually is. Competitions may in any case work as a master key to open the doors of the most insensitive schools and institutions to other actions, which finally lead to a correct teaching of this still largely misrepresented science.

Acknowledgments. We thank Paolo Pasteris, of the technical staff of the Dept. of Informatics of the University of Turin for the constant support in the organization and management of the training courses and laboratories for IOI.

We also thank Giorgio Pidello from Liceo Scientifico “Marie Curie” of Grugliasco, Alberto Barbero from the Istituto Tecnico per Informatici “Vallauri” of Fossano (both

schools are in Piedmont), and all the members of the Dschola working group "New informatics curricula in secondary education" for the many useful discussions and suggestions on the problems of secondary education.

References

Barbero, A., Demo, G. B. (2011). The art of programming in a Technical Institute after the Italian secondary school reform. In: *Proceedings ISSEP 2011*, Bratislava.

Barbero, A., Demo, G. B., Vaschetto, F. (2011). A contribution to the discussion on informatics and robotics in secondary schools. In: *Proceedings RiE*, 2nd International Robotics in Education Conference, Wien.

Casadei, G., Teolis, A. Comprendere e comunicare in modo effettivo: computational thinking. *Annali della Pubblica Istruzione*, 4–5, 15–36.

CINI, GII, GRIN, Manifesto sull'Informatica nella riforma della scuola superiore. `http://www.grin-informatica.it`.

ECDL – European Computer Driving License. `www.ecdl.org/`.

Gove, M. 'Harmful' ICT curriculum set to be dropped this September to make way for rigorous Computer Science. `http://www.education.gov.uk/inthenews/inthenews/a00201864/harmful-ict-curriculum-set-to-be-dropped-this-september-to-make-way-for-rigorous-computer-science`.

Guzdial, M., Ericson, B. (2010). *Introduction to Computing and Programming in Python*, A Multimedia Approach, 2/E, Pearson.

Harvey, H., Mönig, J. Bringing "No ceiling" to scratch: can one language serve kids and computer scientists? In: *Proceedings Constructionism*, Paris.

Italiani, M. (2011). Italian Olympiad in informatics: 10 years of the selection and education. *Olympiads in Informatics*, 5, 140–146.

Iverson, K. (1980). Notation as a tool for thought. *Communcations of ACM*, 33(8), 444–465.

Lissoni, A., Lonati, V., Monga, M., Morpurgo, Torelli, M. (2008). Working for a leap in the general perception of computing. In: *Proceedings of Informatics Education Europe III*, Venice, Italy.

Lonati, V., Monga, M., Morpurgo, A., Torelli, M. (2011). What's the gun in informatics? Working to capture children and teachers into the pleasure of computing. In: *Proceedings of ISSEP 2011 – Informatics in Schools: Contributing to 21st Century Education*, Bratislava.

Martini, S. (2011). Lingua Universalis. *Annali della Pubblica Istruzione*, 4–5, 65–69.

Martina, A., Meo, A., R., Moro, C., Scovazzi, M. (2010). Passo dopo passo impariamo a programmare con Python. `http://linuxdidattica.org/polito/manuale-python-V2.pdf`.

Papert, S. (1997). Why school reform is impossible. *The Journal of the Learning Sciences*, 6(4), 417–427. `www.papert.org/articles/school_reform.html`.

Prensky, M. (2001). Digital natives, digital immigrants. *On the Horizon*, 9(5).

Resnick, M. *et al.* (2009). Scratch: programming for all. *CACM*, 52(11), 60–67.

Salanci, L. (2010). EasyLogo – discovering basic programming concepts in a constructive manner. In: *Proceedings Constructionism 2010*, Paris.

Scarabottolo, N. (2011). Olimpiadi di informatica bilancio di un decennio. *Mondo Digitale*, 38/39.

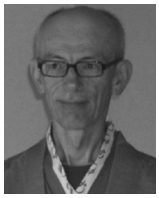
Sysło, M.M. (2011). Outreach to prospective informatics students. In: *Informatics in Schools: Contributing to the 21s Education*, *LNCS*, 7013, *Proceedings ISSEP 2011*, Bratislava.

Tomatis, M. (2010). Mathematics and informatics for crimes. In: *Seminar at the Department Computer Science, University of Torino*, October 2010, `www.marianotomatis.it/index.php?lang=en`.

Wirth, N. (1971). Program development by stepwise refinement. *Communications of the ACM*, 14(4), 221–227.

G. Audrito is a former IOI participant, bronze medal at the 2004 and 2005 editions. He has been involved as a staff member in the Italian IOI project since 2006, and in the regional IOI training courses held in Turin since 2010. He is currently a PhD student in mathematics at the University of Turin.

G.B. Demo is an associate professor of informatics at the University of Turin. She coordinates the working groups on cooperation with schools for informatics teaching both of her department and of GRIN, the National Group of Researchers and Teachers in Informatics of the Italian Universities.



E. Giovannetti is an associate professor of informatics at the University of Turin. He is responsible for the training stages of preparation to the Regional OII in West-Piedmont. He has been teacher in Italian secondary schools, and researcher at the former Italian Telecom Research Centre (CSELT).