Informatics in Education, 2006, Vol. 5, No. 1, 147–159
© 2006 *Institute of Mathematics and Informatics, Vilnius*

The IOI is (not) a Science Olympiad

Tom VERHOEFF
Technische Universiteit Eindhoven
Den Dolech 2, NL-5612 AZ, Eindhoven, The Netherlands
e-mail: t.verhoeff@tue.nl

Received: March 2006

Abstract. The International Olympiad in Informatics (IOI) aspires to be a science olympiad alongside such international olympiads in mathematics, physics, chemistry, and biology. Informatics as a discipline is well suited to a scientific approach and it offers numerous possibilities for competitions with a high scientific standing. We argue that, in its current form, the IOI fails to be scientific in the way it evaluates the work of the contestants.

In this paper, we describe the major ingredients of the IOI to guide further discussions. By presenting the results of an extensive analysis of two IOI competition tasks, we hope to create an awareness of the urgency to address the shortcomings. We offer some suggestions to raise the scientific quality of the IOI.

Key words: informatics competitions, programming contests, computer science.

1. Introduction

The International Olympiad in Informatics, abbreviated IOI, is an annual competition in the discipline of computer science (IOI, 2005). It was established in 1989 and modeled after the International Mathematical Olympiad. The IOI is aimed at pupils in secondary education, especially those talented in computer science. One of the main objectives of the IOI is “to bring the discipline of informatics to the attention of young people” (Statute S1.7 from the IOI Regulations).

The IOI competition has evolved over the years to its current format of a programming contest. We will show that this has led to a number of undesirable features, which conflict with the aspiration of being a member of the family of science olympiads and which interfere with its main objectives. As a consequence, the IOI may fail to attract the talented students that the discipline of informatics needs and it fails to promote the science of computing in the best way possible.

1.1. *Overview*

In Section 2, we describe the IOI competition in more detail, in particular how its informatics content has evolved. Section 3 concerns an extensive analysis of two IOI competition tasks and a re-evaluation of the submitted programs. In Section 4, we summarize the main difficulties in grading algorithms. Section 5 offers some ideas to improve future IOI competitions. Section 6 concludes the paper.

2. The IOI Competition

The current *IOI Regulations* (IOI, 2005) define the IOI as *an annual international informatics competition*. However, these regulations do not otherwise constrain the actual content of the competition. They do constrain the form and organization.

Each year, the Host of the IOI is obliged to produce *Competition Rules and Judging Procedures*. These rules and procedures offer some further insight into the content of the competition. They have evolved over the years, though there is an intent to strive for continuity. Further background information about the IOI competition can be gleaned from (ISC, 2000).

The regulations of the first IOI (IOI, 1989; p.5) are more explicit than the current regulations. We quote:

1.3 "The IOI consists in solving one problem by using personal computer (PC) and it is carried out in one day within four hours."

1.4 "The problem will be selected by the International Jury (see 2.1) on the basis of the problems, provided in advance by the participating countries."

1.5 "The problem will be independent of the PC hardware and high level programming languages can be used for its solution (e.g., BASIC, LOGO, PASCAL, etc.). The problem will be of *algorithmic nature* [emphasis added] and no specialized software packages etc. will be necessary to solve it."

1.6 "The complete solution of each problem includes
   (a) An informal and block-diagram description and *argumentation* [emphasis added] of the algorithm on the basis of which the program is written;
   (b) The source text and results of the program execution, obtained by the microcomputer (2 copies);
   (c) The floppy disk on which the final version of the program is stored, in the source language and eventually in executable code (2 copies)."

The Competition Rules for IOI 2005 state:

> "All of the tasks in IOI 2005 are designed to be algorithmic in nature. Efficiency plays an important role in some tasks. Whenever efficiency of an algorithm is important, certain correct but inefficient approaches can score points . . . "

"Algorithmic in nature" is still rather vague. Current practice is that (most of) the competition tasks present problems which have an algorithm as a solution. The algorithm has to be expressed in one of the approved programming languages (currently, Pascal, C, and C++). The historic motivation for requiring a machine-executable implementation is along the following lines:

- There are no standardized, well-known, abstract algorithm notations[1] (let alone notations within reach of the IOI competitors).
- IOI competitors can be expected to be familiar with at least one popular programming language.

[1]This can be viewed as a weak spot of informatics.

- The syntax and semantics of standardized programming languages are well defined.
- Some qualities of machine-executable programs can be tested automatically.

Note that point 1.6(a) in the IOI'89 Regulations was soon dropped. The main reason being that competitors had little means to express their arguments, and grading of these arguments turned out to be labor intensive.[2] Remember that in those days informatics was not a regular topic in secondary education of most countries. Competitors were mainly self-taught.

Since IOI'94, grading of the programs submitted by the competitors has been automated. Initially, competitors created the executables themselves and no longer submitted source files (disadvantages: competitors had to know about compiler options etc.; no source files available for further analysis; advantage: competitors were in full control of compilation). Since IOI'01, competitors are required to submit their source files, and the organizers handle compilation.

The grading process nowadays is roughly as follows:

1. If the submitted source file does not compile, then the score is zero; otherwise, the resulting executable is evaluated further.
2. The executable is run on each of a number of inputs, while its execution is monitored and the output is captured.
3. If an execution raises an exception (run-time error; e.g., index out of bounds, division by zero, time-limit exceeded), then that particular run scores zero.
4. If an execution terminates normally, without violating any of the resource constraints, then the output is checked for correctness, resulting in a score for the run.
5. The final score is obtained as the sum of the scores for the individual cases. At IOI'05, test runs were clustered into test cases, where the score for a test case is the *minimum* of the scores for the constituting test runs.

The organizers determine the test cases (their number, contents, and maximum score) before the competition, and the same data is used for all competitors. These test cases are not known to the competitors during the competition. And they are not adjusted on the basis of the actual submissions.

The organizers also set resource limits in advance, in particular, on the total execution time and the total data memory. Other resource limitations typically are: single thread (no "forking"), single processor, no auxiliary files, no network access. It also turned out to be necessary to limit the source file size and the compilation time.[3]

The number of test cases and their design have evolved somewhat. At IOI'94, the number ranged from four to eight test runs per task. At IOI'05, it ranged from 20 to 24 cases, with many cases consisting of two runs. Since IOI'04, the design of the test inputs is specifically split into *testing for correctness* with "smaller" inputs and *testing for effi-*

[2]Compare this to the grading at the International Mathematics Olympiad (Verhoeff, 2002), where some 60 mathematicians grade the work on six problems by about 500 competitors over a period of two days.

[3]Submissions are done through a network and compilation is done on a server. These resources are shared by all competitors. Furthermore, through the use of macros, some of the computations could be diverted to the compiler, thereby bypassing the time accounting.

ciency with a variety of more demanding inputs to distinguish relevant efficiency classes (ISC, 2004).

2.1. *Summary of Constraints on the IOI Competition*

As an aid to understand how much freedom there is in setting up an IOI competition, we summarize various constraints. Some of them are non-negotiable, whereas others do leave some room for interpretation. We list in no particular order:

1. The subject matter is informatics, also known as computing science or computer science; in particular, it should include the areas of algorithms and data structures.
2. The competitors are in secondary education from all over the world.[4]
3. The competition tasks should focus on problem solving (and not, e.g., rote learning of encyclopedic knowledge).
4. The problem statements should be elementary to understand, i.e., require no specific prior knowledge, other than what is common at the level of secondary education. Also see (Verhoeff, 2004).
5. It should be possible to obtain solutions by ‘elementary’ means, i.e., this should not require special skills (e.g., touch typing) or specific prior knowledge (e.g., computer architecture).[5]
6. Work submitted by the competitors should be effectively gradable within the schedule and budget of the IOI.
7. The competition tasks should contain some innovative element.
8. The competition tasks should provide diversity in level of difficulty, problem domain, and applicable solution techniques.

Some further points to keep in mind are:

- The IOI competition has a responsibility to promote the discipline of informatics; it serves as a role model for national olympiads and secondary education; it is not intended to be recreational or entertainment only.
- There is a natural language barrier to be crossed twice: both from organizers to competitors and from competitors to organizers. We cannot rely on a common natural language.

3. Some Results of Analyzing Past IOIs

As a master’s thesis project, Wouter van Leeuwen analyzed two IOI competition tasks, together with the submitted work for these tasks and the scores obtained at the IOI (van Leeuwen, 2005). His research concerned the tasks[6] Median (IOI 2000) and Phidias (IOI 2004):

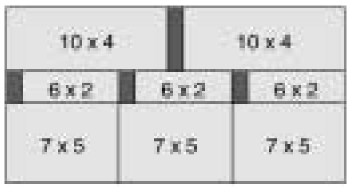
[4]The IOI Regulations provide a more precise definition.

[5]Nowadays, many problems require such techniques as dynamic programming or branch&bound.

[6]The tasks can be found on the IOI website, but the details do not matter for the discussion.

Median Given an odd number of objects with distinct strengths, find the object of median strength using only the function *Med3*(a, b, c) that returns the object of median strength among *three* distinct objects a, b, c. It is cast as a reactive task, where the program to be designed communicates with an environment implemented as a library.

Phidias Determine the minimum wasted area, when cutting a marble slab to obtain plates from a given set of desired dimensions. The cuts are done sequentially and they run either horizontally or vertically, going all they way from one side to the opposite side (see the figure below).



3.1. *Analysis of Results for Median*

Table 1 shows how many competitors (of the 209 that did submit a source file)[7] obtained what score. Since there were ten test inputs, each with a maximum score of ten points, the scores range from 0 to 100 in steps of 10. The right-hand column gives the total number of points obtained by these competitors together. The task Median was a harder task with only 10% of the competitors obtaining a score of 90% or more.

A detailed analysis of the task Median and the design of the test inputs used at IOI'00 is given by (Horváth and Verhoeff, 2002). This task appears to have a rich solution space. The analysis of van Leeuwen, however, showed that the competitors were able to find additional interesting algorithms. This was not discovered during IOI'00, because of the automatic grading, which treats the submissions as black boxes. We will give an example in Section 5.

The re-evaluation done by van Leeuwen was aimed primarily at correctness. He prepared an algorithm taxonomy, investigated the source files, reran the programs on the IOI

Table 1
Frequency distribution of scores for task Median (IOI, 2000)

Score	100	90	80	70	60	50	40	30	20	10	0	**Total**
IOI, 2000	24	6	3	3	16	5	11	16	15	59	51	6410
v. Leeuwen	18	6	1	0	15	2	8	14	5	12	128	4380

[7]Of the 274 competitors, 226 submitted an executable. These numbers include the second team from the Chinese host.

test data, constructed new test cases, analyzed the log files for suspicious characteristics (e.g., by visualizing the queries and comparing this to the ‘finger prints’ of known correct algorithms; see Fig. 1).

Note that there are two ways in which competitors lose points:

- because of incorrectness, and
- because of inefficiency.

The difference cannot be seen in the statistics.[8] It turned out that under IOI grading, 99 competitors submitted programs that produced an incorrect output for at least one of the test cases. Furthermore, 67 of these had a non-zero score. From the IOI grading it is completely unclear what kind of correctness errors were made, and how they should be ranked with respect to each other. The research by van Leeuwen uncovered another 10 incorrect submissions, that were not identified as such by the IOI grading. Note that van Leeuwen decided to be quite strict in his re-evaluation, where he assigned a zero score to all submissions that were incorrect.[9]

From Table 1, we see that over 30% of the total score has not been properly justified. Why would one incorrect program obtain a higher score than another? The test cases hardly say a thing about the kind of correctness error. Even for efficiency, there are scores that one can doubt, because the 18 remaining solutions that score 100% are not sufficiently efficient in the worst case, that is, they cannot solve all allowed inputs within the limits.

Because of its reactive nature, the task Median allows for some additional automatic evaluation techniques:

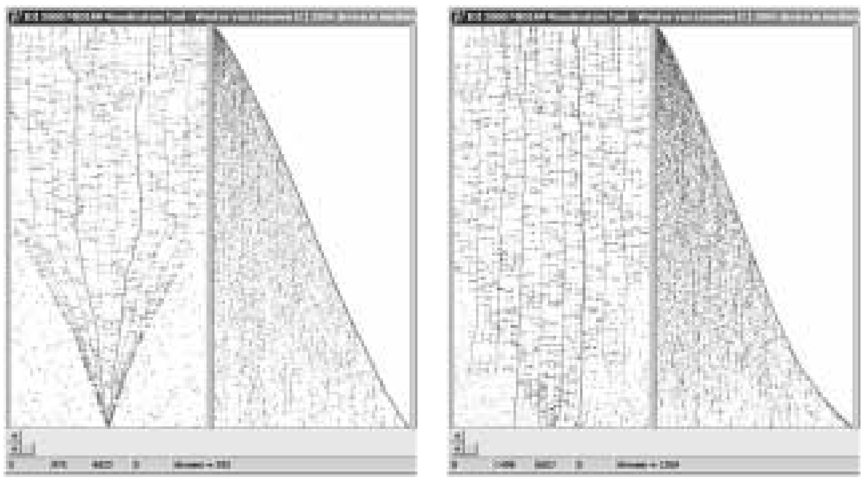


Fig. 1. Visualization of log files: Ternary Insertion Sort Zoom (left), incorrect implementation (right).

[8]In some cases, it is even not so easy to classify the cause automatically; e.g., a memory exception may have been caused by an index going out of bounds (incorrectness), or insufficient memory in the machine (inefficiency).

[9]His motivation was that the results on the test cases were not a good basis for differentiating the various incorrect submissions.

- By analyzing the log file that records the dialog, it is possible to do stricter correctness testing (e.g., detect *bluffing*), and algorithms can be classified through their behavior *during* a test run.
- By using input values that depend on previous outputs, it is possible to provoke (near) worst-case behavior (using a so-called *adversary*).
- Resource limitations can be enforced more objectively (through counting, rather than through timing).

3.2. *Analysis of Results for Phidias*

Table 2 shows how many of the 295 competitors[10] obtained what score. Since there were 20 test inputs, each with a maximum score of five points, the scores range from 0 to 100 in steps of 5. The right-hand column gives the total number of points obtained by these competitors together. The task Phidias was a medium task with 35% of the competitors obtaining a score of 90% or more. By that standard, it was the easiest task at IOI'04, and that was also the reason for analyzing it in detail.

Under IOI grading, 104 competitors submitted programs that produced an incorrect output for at least one of the test cases. Furthermore, 89 of these had a non-zero score. The research by van Leeuwen uncovered another 57 incorrect submissions, that were not identified as such by the IOI grading. From Table 2, we see that over 50% of the total score has not been adequately justified.

For example, some competitors used an incorrect recurrence relation, where only vertical (horizontal) cuts at multiples of the widths (heights) of desired plates were considered. The smallest counterexample that we could find is depicted in Fig. 2. Cutting only at multiples of desired plates will result in a nonzero waste.

It also happened that programs obtained only 60 points, whereas in hindsight there are good reasons to agree that they qualified for 100 points. It turned out that most of the test data for efficiency was near the upper bound, and it was slightly too big for a

Table 2
Frequency distribution of scores for task Phidias (IOI, 2004)

Score	100	95	90	85	80	75	70	65	60	55	50
IOI, 2004	76	12	6	9	1	5	12	6	15	9	19
v. Leeuwen	48	10	0	8	0	3	5	3	8	3	0

Score	45	40	35	30	25	20	15	10	5	0	**Total**
IOI, 2004	4	8	7	22	5	2	3	7	7	60	15795
v. Leeuwen	0	0	0	0	0	0	0	0	0	207	7845

[10]These numbers include the second team from the Greek host.

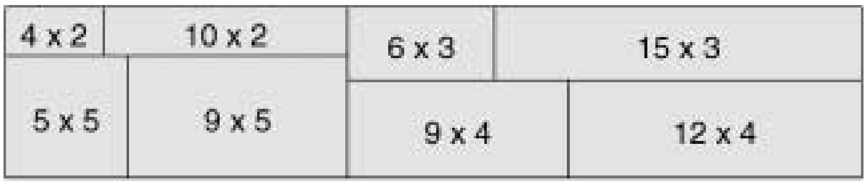


Fig. 2. Counterexample to incorrect recurrence relation based on multiples

particular combination of programming language and solution technique, which clearly was not anticipated in the design of the test data.

4. The Difficulties of Grading Submitted Programs

In the preceding section we have shown some undesirable consequences of the current IOI grading practices. (Forišek, 2006) shows more examples from other competitions and explains why a large class of tasks is not suitable for black-box testing. (Cormack, 2006) shows that at IOI 2005, the final rankings “involve a considerable degree of chance”, mostly due to awarding points to incorrect programs. Our in-depth analysis concerns only two tasks, and one might hope that these are exceptions. Unfortunately, the method of grading the submitted programs by black-box testing is fundamentally flawed in general. Let us first summarize the observed shortcomings:

- incorrect programs are frequently not identified as such;
- scores for known-incorrect programs often cannot be properly justified;
- detection and scoring of inefficient programs are similarly flawed.[11]

Computer scientists have known and acknowledged the limitations of black-box testing for a long time, as this famous statement from Edsger Dijkstra’s Turing Lecture (Dijkstra, 1972) shows:

> “[P]rogram testing can be a very effective way to show the presence of bugs, but it is hopelessly inadequate for showing their absence. The only effective way to raise the confidence level of a program significantly is to give a convincing proof of its correctness[12].”

In fact, Dijkstra already made this observation over 35 years ago in (Dijkstra, 1970), where he also states:

> “[A]s long as we regard the mechanism as a black box, testing is the only thing we can do. The inescapable conclusion is that we cannot afford to regard the mechanism as a black box. . . ”

It is equally hopeless to construct a convincing proof for a given program after the fact, especially for someone other than the author. The burden of proof rests on the program’s de-

[11]Besides false positives, we have also seen false negatives in performance assessment.
[12]This holds not only for correctness, but also for performance, be it worst-case, average-case, or best-case.

signer. Scientists and engineers are expected to provide the reasoning behind their claims. (Petroski, 1992) formulates this nicely:

> "[I]t is the essence of modern engineering not only to be able to check one's own work, but also to have one's work checked and to be able to check the work of others. In order for this to be done, the work must follow certain conventions, conform to certain standards, and be an understandable piece of technical communication."

It is not the IOI organizers' duty to show why submissions are incorrect or inefficient. This can be nearly impossible, especially if competitors seek ways to make this even harder[13]. It is the competitors' duty to convince the organizers that their submissions are correct and efficient. This will involve *design arguments*, since that is the way science and engineering work. The current grading practice in the IOI helps perpetuate a non-scientific attitude.

Black-box testing is not adequate for determining program correctness. It is even less appropriate for assigning a (nonzero) score to incorrect programs. A failed test case does not reveal the reason behind the failure. It could have occurred in understanding the problem, analyzing the problem, designing the algorithm, or implementing the program. If we want a balanced judgment of incorrect programs, then we need to look into the black box even further.

The situation with respect to performance is more difficult still. We must face the distinction between asymptotic time and memory efficiency of abstract algorithms versus the ability of a specific program to handle a specific range of inputs within specific resource bounds. Both are hard to assess by doing some limited black-box testing.

5. Ideas for Future Directions

A first step toward improvement is to establish the *principles* behind the IOI competition and its grading process. The current regulations and rules are not very informative. What aspects of computing do we find important? Problem understanding, problem analysis, solution design, program implementation, functional correctness, efficiency, verifiability? The second step should be to select *task types* and *grading methods* in agreement with the principles.

One lesson from the past is that test data needs to be better designed and motivated. The motivation should be made available to the delegation leaders and competitors after the competition. However, as we have shown above, this will not be enough, even not throwing a thousand test cases at a program[14]:

> "[T]esting by random sampling is hopelessly inadequate as well, because even the most vigorous exercising possible will only cover a truly negligible

[13]Many submissions do not seem to strive for elegance and simplicity, and some are obfuscated on purpose (turning them into a secured black box).

[14]Also think of the effort to design them, to execute and evaluate them, and to interpret the results.

> fraction of the possible number of cases, and whole classes of in some sense critical cases can —and will!— be missed: only the most obvious blunders will show up." Quoted from (Dijkstra, 1970)

Furthermore, after an IOI, the submissions need to be analyzed thoroughly, because that is the only way to find out how well the grading was actually done and to learn how to improve the grading in the future.

Even when we stick to competition tasks that are "algorithmic in nature", there are alternatives to a programming contest. For instance, a task could involve one or more algorithms, but not as its solution. We give two examples in §5.1 and §5.2.

5.1. *Theoretical Analysis of Algorithms*

The first class of such tasks concerns *reasoning* about algorithms. Consider the following new algorithm for solving task Median of IOI'00, which van Leeuwen discovered among the submissions. We named it *Bubble Search* because of it resembles Bubble Sort.

```
1. Put all $N$ objects in a sequence, say $a[i]$ for $i \in \{ 1, \ldots, N \}$
2. Let $M = (N+1)$ $\mathbf{div}$ $2$
3. Repeat the following for-loop, until no swap has occurred in $a$
       For each $i \in \{ 1, \ldots, M-1 \}$ do
          i. Let $x$ be such that $a[x] = \mathit{Med3}(a[i], a[M], a[M+i])$
         ii. If $x \neq M$ then swap $a[x]$ and $a[M]$
```

This algorithm is obviously correct *if* it terminates, because in the final pass through the for-loop, $a[M]$ has not moved and all pairs $a[i], a[M+i]$ surround $a[M]$. But it is far from obvious *why* the algorithm terminates. Without termination *argument*, the algorithm is merely a bluff. It is a nice algorithmic problem to discover why it terminates. We posed the following questions to the candidates at the training camp for the German IOI'05 delegation.

- Why does the algorithm terminate for all allowed inputs?
- What is the worst-case number of *Med3* calls as a function of N?
- What is a witness input for worst-case performance?
- What is the average-case performance?

Somewhat to our surprise, they were very capable of formulating (either in German or English) proper arguments for termination. They also solved the two problems concerning worst-case performance. The last problem was not really solved, though they did carry out some programmed experiments and did some curve fitting (it is still an open problem).

5.2. *Experimental Analysis of Algorithms*

Another class of tasks concerns *experimentation* with algorithms. As an example, consider $N \times N$-matrices of floating-point numbers. The sparsity of such a matrix is the percentage of zero entries. The task concerns a comparison of two representations for such matrices. The *array* representation is simply as a two-dimensional array; the *list* representation is as a singly-linked list of singly-linked lists of pairs (non-zero entry, column index). The latter representation is especially suited for sparse matrices. If you know the memory usage of floating-point numbers, pointers, and indices, then you can calculate at what sparsity the list representation is more memory efficient than the array representation. This comparison is less obvious for the time efficiency of operations on such matrices, such as for instance matrix squaring. The task is to *set up, carry out, and interpret the results of an experiment to determine the turnover sparsity, where matrix squaring becomes more time efficient under the list representation than under the array representation*.[15] Randomized algorithms are also a nice source for experimentation.

5.3. *Grading*

Grading such tasks, where competitors deliver arguments or experiments rather than machine-executable algorithms, requires a new approach. But in my opinion even proper grading of tasks with algorithms as solutions requires another approach from the traditional execution-only method that treats the algorithms as black boxes. Is there any university-level algorithms course where students are examined solely on the basis of black-box testing?

We propose to investigate the following approach. Besides program code, the competitors also deliver the key ingredients of their reasoning. The task formulation could be augmented to ask for such things as the following.

- The output for a simple input case, to show their understanding of the task.
- A formulation (model) of the task that abstracts from the story.
- Key observations about this model that motivate the presented solution.
- Main design arguments concerning algorithms and data structures.

The organizers need to prepare a motivated *grading scheme*, that summarizes relevant parts of the solution space (for example, on the basis of an algorithm taxonomy), common errors, and their associated scores. The mathematics, physics, and chemistry olympiads use similar grading schemes; see (Verhoeff, 2002) for an explanation of the IMO grading scheme. Note that the decision on the scores in the grading schemes is not a scientific issue but a policy issue.

The actual IOI grading should be tool supported, including measurements obtained from the execution of programs. The delegation leaders should have a larger role in the evaluation. This is not only needed to overcome the language barrier, but also because it would foster a better attitude towards algorithms.

6. Conclusion

We have shown that the current grading practice of the IOI has severe shortcomings. It is a (well-known) mistake to evaluate the correctness and efficiency of an algorithm solely

[15]Of course, some further details need to be supplied. I found the result surprising: the turnover is at approx. 25% zeros.

on the basis of executing the implementation on a set of test cases. It is reasonable to doubt the validity of the results of the IOI from a scientific point of view. Analysis of two IOI tasks and the submitted work confirms these doubts.[16] Although our analysis concerns only two tasks, we now understand better why it is actually a more fundamental problem that should not be viewed as something specific to these tasks.

The computer science goals of the IOI should be stated more explicitly. In particular, the meaning and value of correctness and efficiency should be better established. Is there a reasonable way to rank two incorrect algorithms? What about ranking algorithms by worst-case, average-case, or best-case efficiency?

Within the area of algorithmics, there are various other types of competition tasks that could be posed. It might not always be feasible to grade the submitted work automatically. But the desire for automation as carried out in the past has blinded the IOI community: the real merits of the work of the competitors have remained invisible. Even in the case of "traditional" IOI tasks, it would be advisable to base the result not only on the opinion of a preprogrammed grading automaton. A thoroughly prepared and motivated grading scheme, supported by measurements, is recommended as a better basis for evaluating the submissions.

References

Cormack, G (2006). Random factors in IOI test case scoring. *Informatics in Education*, **5**(1), 5–14.

Dijkstra, E.W. (1970). Concern for correctness as a guiding principle for program composition. *EWD288*, July.
`http://www.cs.utexas.edu/users/EWD/transcriptions/EWD02xx/EWD288.html`

Dijkstra, E.W. (1972). The humble programmer. *Communications of the ACM*, **15**(10), 859–866 (ACM Turing Lecture 1972), also EWD340.
`http://www.cs.utexas.edu/users/EWD/transcriptions/EWD02xx/EWD340.html`

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

Horváth, G., and T. Verhoeff (2002). Finding the median under IOI conditions. *Informatics in Education*, **1**(1), 73–92.

Kenderov, P.S., and N.M. Maneva (Eds.) (1989). *Proceedings of the International Olympiad in Informatics*. Pravetz, Bulgaria, May 16–19, 1989. Union of the Mathematicians in Bulgaria, Sofia.

IOI, International Olympiad in Informatics.
`http://www.IOInformatics.org/` (accessed October 2005).

ISC (IOI Scientific Committee). *Guidelines for IOI Competitions*. Version 0.3, Jan./Feb. 2000.
`http://scienceolympiads.org/ioi/sc/documents/ioi-sc-guide-03-nr.txt`

ISC (IOI Scientific Committee). *Principles behind IOI 2004 Competition Tasks and their Grading*.
`http://scienceolympiads.org/ioi/sc/documents/principles-2004.txt`

van Leeuwen, W.T. (2005). *A Critical Analysis of the IOI Grading Process with an Application of Algorithm Taxonomies*. Master's Thesis, Technische Universiteit Eindhoven, Faculty of Mathematics and Computing Science. `http://www.win.tue.nl/ wstomv/misc/ioi-analysis/thesis-final.pdf`

Petroski, H. (1992). *To Engineer is Human: The Role of Failure in Successful Design*. Vintage.

Verhoeff, T. (2002). *The 43rd International Mathematical Olympiad: A Reflective Report on IMO 2002*. Computing Science Report 02-11, Faculty of Mathematics and Computing Science, Technische Universiteit Eindhoven. `http://www.win.tue.nl/ wstomv/publications/imo2002report.pdf`

Verhoeff, T. (2004). *Concepts, Terminology, and Notations for IOI Competition Tasks*, document presented at IOI 2004 in Athens.
`http://scienceolympiads.org/ioi/sc/documents/terminology.pdf`

[16](Cormack, 2006) and (Forišek, 2006) support this from a different perspective.

T. Verhoeff is an assistant professor in computing science at the Technische Universiteit Eindhoven (TU/e), The Netherlands. He obtained his PhD (*A Theory of Delay-Insensitive Systems*, 1994) from TU/e. His current research area is software engineering technology. He chaired the Host Scientific Committee of IOI 1995 in The Netherlands and contributed tasks to various competitions. In 1999, he was Finals Directors of the ACM ICPC World Finals held in Eindhoven, The Netherlands. He chairs the IOI Scientific Committee since 1999.

Tarptautinė informatikos olimpiada – tai (ne)mokslinė olimpiada

Tom VERHOEFF

Tarptautinė informatikos olimpiada, kaip ir matematikos, fizikos, chemijos ar biologijos, siekia būti moksline olimpiada. Informatikos disciplina puikiai atitinka moksliškumo kriterijus, šiam dalykui nesunkiai pritaikomos įvairaus pobūdžio varžybos, pasižyminčios aukštomis moksliškumo normomis. Straipsnyje iškeliama mintis, jog Tarptautinė informatikos olimpiada, – tokia, kokia ji yra dabar, – negali būti laikoma moksline dėl joje taikomos varžybų dalyvių darbų vertinimo praktikos. Straipsnyje aptariami pagrindiniai informatikos olimpiados komponentai, siekiama tolesnių diskusijų atgarsio. Supažindinant su rezultatais, pasiektais atliekant išsamią dviejų Tarptautinės informatikos olimpiados užduočių analizę, siekiama pabrėžti esamų trūkumų pašalinimo neatidėliotinumą. Taip pat pateikiama siūlymų, skirtų pagerinti Tarptautinės informatikos olimpiados mokslinę kokybę.