Informatics Olympiads: Challenges in Programming and Algorithm Design

Benjamin A. Burton

Department of Mathematics, SMGS
RMIT University,
GPO Box 2476V, Melbourne, VIC 3001,
Email: bab@debian.org

Author's self-archived version
Available from http://www.maths.uq.edu.au/~bab/papers/

Abstract

The International Olympiad in Informatics is a worldwide contest for high school students, with a strong focus on creativity and ingenuity in algorithm design. Here we describe the activities in Australia that support and complement this contest, including a range of programming competitions, more accessible pen-and-paper competitions, and other enrichment and training activities. Sample problems are included, along with suggestions for becoming involved.

1 Introduction

The International Olympiad in Informatics (IOI) is a prestigious international competition for high school students in programming and algorithm design. Created in 1989 in Bulgaria under the leadership of Petar Kenderov, it now boasts delegations and guests from around 90 different countries.

Students sit the IOI on an individual basis, and are given ten hours to solve six problems. The contest is a programming competition, in the sense that students submit programs which are then run through a variety of test scenarios and judged accordingly. However, the difficulty lies not so much in the programming but rather the design of the underlying algorithms.

Australia first entered the IOI in 1992, and became a regular participant in 1999. With a growing support base from academics, teachers and ex-students, a rich national programme is developing to support and complement the IOI.

The primary focus of this paper is to introduce the various activities that form the Australian programme. Section 2 presents an overview of these activities. In Section 3 we focus in detail on written competitions, a more accessible alternative that involves multiple choice and short answer problems, and in Section 4 we return to a detailed discussion of programming competitions. Broader activities within the Asia-Pacific region are discussed in Section 5, and Section 6 closes with suggestions for how teachers and students can become involved.

2 The Australian Programme

Like its sister programme in mathematics, the Australian informatics olympiad programme currently runs under the auspices of the Australian Mathematics Trust. Although the initial motivation for this programme was the annual selection and training of the Australian IOI team, it has since grown to provide enrichment for a wider range of high school students nationwide. Activities include:

- *The Australian Informatics Competition (AIC):* This is the most widely accessible activity, since it involves no programming at all. Held annually in May, it offers students a range of multiple choice and short answer questions that encourage algorithmic thinking in puzzle-like settings. The AIC and some sample problems are discussed in detail in Section 3.

  The AIC first ran in 2005, and has grown to over 3000 participants in 2007.

- *The Australian Informatics Olympiad (AIO):* The AIO is a true programming contest, and acts as the first round of selection in working towards an IOI team. Although the problems are necessarily simple, many of them retain a focus on algorithm design even at this early stage. See Section 4 for details and sample problems.

  The AIO has run since 1998. Numbers in this contest are typically much lower, with around 80 participants in 2007.

- *The School of Excellence:* The top twelve entrants from the AIO are invited to a live-in "programming boot camp" at the Australian National University in December, where they are given ten days of lectures, labs, contests and other activities. The school is intensive, and students typically emerge exhausted but full of ideas.

- *Invitational Contests:* The participants from the School of Excellence are invited to sit additional contests in February and March, including the French-Australian contest discussed in Section 5. These contests push the standard closer to IOI level, and (unlike the AIO) do not shy away from "required knowledge" such as graph theory and dynamic programming.

- *The Team Selection School:* Based on the invitational contests, a final eight students are invited to a second training school at Macquarie University. Where the focus of the December school is on teaching new material, the focus of this April school is on using this material in creative and unusual ways to solve problems of IOI difficulty. At the end of this school a final team of four members is chosen to represent Australia at the coming IOI.

- *The International Olympiad:* The four team members are individually mentored for the following 3–4 months. In August they meet for a final short but intense training school, after which they head directly overseas for the IOI.

Online materials are available through the national training site all year round (`http://orac.amt.edu.au/aioc/train/`), and a series of books to complement these materials is currently under development.

3 Written Competitions

It was noted in the introduction that the AIO—an entry level programming contest—has extremely low participation each year. Whilst there may be many reasons behind this, it is highly probable that the following factors contribute:

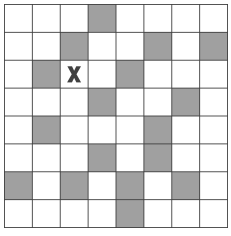
- Programming contests are difficult for schools to run. A typical mathematics contest requires nothing more than a desk and a pen. In contrast, a programming contest requires a computer for every student, appropriate software (compilers and debuggers), and a supervisor who can deal with technical problems if they arise.
- Programming contests require students who can write computer programs. In a mathematics contest, any student can follow their nose and scribble ideas down. In a programming contest—certainly the traditional type in which programs are scored according to their behaviour—a student cannot score any points (or even have their submissions judged) unless they can create a running program in a relatively short period of time.

For these reasons it was decided to complement the programming contests with a written contest, in the hope that this written contest might have broader appeal. The result was the Australian Informatics Competition, which has run annually since 2005.

Dungeon

(Australian Informatics Competition 2005, Intermediate)

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



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

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

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

Figure 1: The problem "Dungeon"

Lost

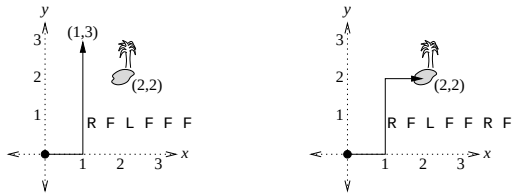
(Australian Informatics Competition 2007, Intermediate)

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

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

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

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



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

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

1. `RFFLFLFFFRFF` ⟶ $(3, 3)$

2. `RFFLFRFFLFFLFLFRFFR`
`FFLFLFRFRFFFRFLFFLF` ⟶ $(5, 5)$

3. `RFFFLFFRFFFFRFRFFRFFFF`
`FFLFFLFFFLFLFFFFFLFF` ⟶ $(8, 8)$

Figure 2: The problem "Lost"

Although the AIC is styled as an informatics competition, AIC questions almost never use any code or pseudocode, and only a minority describe any explicit algorithm. Most problems pose some form of puzzle which, in order to be solved correctly, requires students to devise some type of informal algorithm in their heads.

An example from the first AIC is *Dungeon*, described in Figure 1. Although the problem can be solved by ad-hoc trials and guesses, it is faster and more reliable to work systematically outwards from the token, identifying all the squares that are one move away, then two moves away, and so on. Essentially the student who has never seen programming is encouraged to informally conduct a breadth-first search.

As well as multiple choice problems, the AIC contains a number of "algorithmic problems". An example is *Lost*, seen in Figure 2. Each algorithmic problem contains a task description followed by three scenarios, each of which can be solved with an integer in the range 0–999. The first scenario is typically small and easy to solve in an ad-hoc fashion, whereas the third is typically large and requires a systematic

algorithm to solve quickly. The hope is that, as students attempt the simpler scenarios, they develop a feel for the problem and a systematic method that will allow them to tackle the larger cases.

Although written contests in computer science are relatively rare in comparison to programming contests, examples can be certainly be found elsewhere. One prominent example is the Lithuanian *Beaver* contest, which like the AIC encourages algorithmic thinking without explicitly requiring an understanding of computer programming (Dagienė 2006).

For a more detailed discussion of the AIC and additional sample problems, the reader is referred to Clark (2006).

4 Programming Competitions

Whilst written competitions can offer a highly accessible introduction to algorithms, the core activities of the Australian olympiad programme revolve around programming competitions.

The competitions offered in the Australian programme typically follow the model of the international olympiad. Students are given a large amount of time to solve a small number of tasks, each of which requires them to write a computer program. Each task describes the precise problem to solve, offers a simple text format for reading input scenarios and writing corresponding solutions, and sets time and/or memory limits within which the program must run.

A typical task of this type is *Mansion*, illustrated in Figure 3. Problems in the international olympiad are of course more difficult; worked examples are discussed by Horváth et al. (2002) and Burton (2007), and a comprehensive list of past IOI problems can be found at the IOI secretariat (`http://olympiads.win.tue.nl/ioi/`).

Readers might be familiar with tasks of this type from university programming contests, such as the ACM International Collegiate Programming Contest or the TopCoder contests. The International Olympiad in Informatics differs from these contests in the following ways:

- IOI tasks are graded on a sliding scale from 0 to 100, instead of an all-or-nothing pass or fail. This allows a range of scores for solutions of varying sophistication and efficiency.
- IOI tasks are extremely difficult to solve completely, since the running time and memory constraints for programs are often very tight. Whilst it might be straightforward to write a correct but inefficient solution that scores partial marks, it is often a significant achievement to score full marks for an IOI task.
- Students do not race each other. What matters is not when they submit each program, but only how it performs. This encourages stronger students to take their time in implementing sophisticated algorithms, in the hope of passing even the most difficult test scenarios.

Of course different styles of contest have different strengths. For instance, the all-or-nothing scoring for the ACM and TopCoder contests places a strong emphasis on rigour, whereas the extreme running time and memory constraints of IOI problems place the focus squarely on creative algorithm design. A more detailed comparison of different programming contests is given by Cormack et al. (2006).

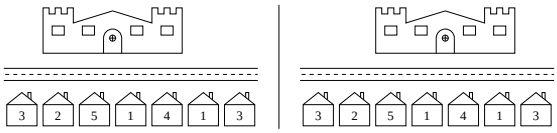
At the national level, the Australian Informatics Olympiad is intended as an entry-level programming contest for students with little or no formal training.

Mansion

(Australian Informatics Olympiad 2007, Intermediate Q2)

You wish to build a mansion beside a long road. The far side of the road is filled with n houses, each containing a given number of people. Your mansion is as long as w houses combined. Your task is to position the mansion so that as many people as possible live across the road from you.

For instance, consider the road illustrated below, with $n = 7$ and $w = 4$. Here the seven houses contain 3, 2, 5, 1, 4, 1 and 3 people respectively. The first diagram places the mansion across from $2+5+1+4 = 12$ people, whereas the second diagram places it across from $5+1+4+1 = 11$ people. Indeed, of all the possible locations for the mansion, the largest possible number of people living across the road is 12.



Input: Your program must read its input from the file `manin.txt`. The first line of this file will give the integers n and w, and the following n lines will give the number of people living in each house.

Output: Your program must write its output to the file `manout.txt`. This file must give the largest possible number of people living across from the mansion.

Limits: Your program must run within 1 second. The input integers are guaranteed to lie in the range $1 \leq w \leq n \leq 100\,000$.

Sample Input and Output: The sample input and output files below correspond to the example given above.

`manin.txt`:

```
7 4
3
2
5
1
4
1
3
```

`manout.txt`:

```
12
```

Figure 3: The problem "Mansion"

At this level the scores are more likely to reflect the nuts and bolts of computer programming; as mentioned in Section 3, merely asking for correct running code is a relatively high bar for entry at the high school level.

Nevertheless, most problems in the AIO retain a focus on algorithm design. The challenge for the problem setters is to find *accessible* problems, where (i) the "obvious" algorithm is not necessarily the best, but (ii) good students with no formal training can be expected to find optimal algorithms through insight and creative thinking.

The problem *Mansion* (Figure 3) offers an example from the 2007 AIO. Essentially the problems asks, given an array of length n, for the continuous subarray of length w with the largest possible sum.

A simple algorithm might loop through all possible starting points, and for each starting point loop again to sum the w elements of the sub-array. Whilst correct, this algorithm runs in quadratic time and is too slow to score 100% (in the AIO such a solution scored 70%).

This algorithm can be improved as follows. We retain the outer loop through all possible starting

points, but we avoid the inner loop by using a *sliding window*. As we move from one starting point to the next, we adjust the sum by subtracting the one element that has been lost and adding the one element that has been gained. The resulting algorithm runs in linear time, and is fast enough to score 100%.

It is pleasing to note that, of the AIO entrants who obtained correct or almost correct solutions to this problem, 18 used the linear algorithm and 21 used the quadratic algorithm. This suggests that the optimal solution was indeed non-obvious but nevertheless accessible, as the problem setters had hoped.

Figure 4 describes *Restaurants*, a more difficult problem from the 2007 AIO. Whereas the challenge in Mansion is efficiency, the challenge in Restaurants is correctness.

Restaurants

(Australian Informatics Olympiad 2007, Senior Q3)

You are faced with the unenviable task of organising dinner for an international conference. Several countries are represented at the conference, each with a given number of delegates. You have also identified several restaurants in the neighbourhood, each with a different number of seats.

In order to break down international barriers, you cannot seat two people from the same country at the same restaurant. Your task is to find an arrangement that seats as many people as possible.

As an example. suppose there are three countries with 4, 3 and 3 delegates respectively, and three restaurants with 5, 2 and 3 seats respectively. You can seat most of the delegates by placing one delegate from the second country and one delegate from the third country in every restaurant, and by placing two delegates from the first country in the first and third restaurants. This leaves two delegates without dinner, which is the best you can do.

Input: Your program must read its input from the file `restin.txt`. The first line of this file will give the number of countries, and the second line will give the number of delegates from each country. Likewise, the third line will give the number of restaurants, and the fourth (and final) line will give the number of seats in each restaurant.

Output: Your program must write its output to the file `restout.txt`. This file must give the smallest possible number of delegates who cannot be seated.

Limits: Your program must run within 1 second. There will be at most 5000 countries and 5000 restaurants.

Sample Input and Output: The sample input and output files below correspond to the example described above.

`restin.txt`:

```
3
4 3 3
3
5 2 3
```

`restout.txt`:

```
2
```

Figure 4: The problem "Restaurants"

Most of the students who attempted this problem adopted some type of greedy approach, and indeed the official solution is greedy (for each country, seat its delegates in order from the restaurant with the most empty seats to the restaurant with the fewest).

The difficulty is that not all greedy approaches are correct. For instance, some students adopted a similar approach but began with the largest restaurant instead of the emptiest; this works with the sample input and output, but does not work for more complex scenarios. In the end, 16 of the 27 students who attempted this problem scored 100%.

It is worth pausing to consider the ways in which this and other programming contests are judged. In particular, because solutions are judged entirely by their behaviour, students with partial or buggy implementations can score zero, even if they have derived the correct algorithm. In some cases (particularly in the IOI), good students may deliberately choose to submit an inefficient solution to avoid the risk of bugs that comes with more complex code.

This style of judging also raises pedagogical issues. Judging purely by behaviour does little to encourage good programming habits, and does not develop the communication skills that are crucial for teamwork and research in later life. This latter issue is explicitly addressed at the Australian training schools, where participants regularly present their algorithms to the other students and analyse them in a group setting.

The limitations of the current judging style are well understood, and the international community is actively engaged in finding ways to address them. Cormack et al. (2006) and Opmanis (2006) discuss the issues in depth and offer some concrete suggestions for improvement, and Burton (2007) examines them in the context of human-evaluated mathematics competitions. The IOI itself is actively evolving to find the right balance between competition, education and encouragement.

5 Regional Activities

To complement the national programme, it is valuable for students to engage in international competition and cooperation. Not only does this give them stronger experience in competition, but it also enhances the sense of camaraderie and helps them feel part of a wider community.

The IOI itself is a pinnacle of international competition, but with teams of four it can only be offered to a handful of students. To complement this, the regional and international communities have developed several smaller events that allow a greater depth of students to participate.

The first such event to appear on the Australian calendar was the French-Australian Regional Informatics Olympiad (`http://www.fario.org/`). This began in 2004 as a collaborative effort between Australia and France, and works well because the students of both countries have comparable skills. The contest has broader interest however, and each year a handful of students from other countries enter as unofficial participants.

More recently, the Asia-Pacific region has formed a new contest in the lead-up to the IOI. The inaugural Asia-Pacific Informatics Olympiad (`http://www.apio.olympiad.org/`) was hosted by Australia in 2007, with over 350 participants from 14 delegations. With Thailand and India lined up to host in 2008 and 2009, the contest is set to become a regular event on the regional calendar.

In addition to competitions, there is also collaboration in training between different countries. The Australian team met with the French in 2007 for a final week of joint training before the IOI, and two New Zealand students joined the Australians for the 2007 School of Excellence. At the teaching level, France and Australia regularly share problems and discuss training methods, and at IOI 2007 there was a mini-conference at which a diverse group of team leaders shared ideas and experiences.

6 Becoming Involved

For anyone eager to become involved in the programme as a teacher or a student, there are several excellent resources for learning more about programming contests.

Skiena et al. (2003) have written a superb book that focuses specifically on programming contests such as the IOI. It is very readable, contains a wealth of problems, and covers not only algorithms but also the practical issues of writing code in a contest environment.

Many countries have their own training sites, through which students can teach themselves in their own time. An excellent example is the USACO site (`http://www.usaco.org/`), which offers problems, reading notes and contest advice. The Australian site (`http://orac.amt.edu.au/aioc/train/`) includes all past AIO, French-Australian and Asia-Pacific papers. Both sites allow students to submit solutions with instant feedback, and are open to participants worldwide.

At the level of the IOI, Verhoeff et al. (2006) have proposed a "syllabus" of topics that might be covered. This list is currently under active discussion within the IOI community.

Interested people are also encouraged to contact their national organisation for information on local contests and training materials. The IOI secretariat (`http://olympiads.win.tue.nl/ioi/`) maintains a list of these organisations, alongside a wealth of archival material on the IOI and related competitions.

The Australian organisation can be reached through the author of this paper, who currently holds the role of Director of Training. The Australian Mathematics Trust, which oversees and administers the programme, can be reached through its Executive Director Peter Taylor at `pjt@olympiad.org`.

References

Burton, B. (2007), 'Informatics olympiads: Approaching mathematics through code', *Mathematics Competitions* **20**(2) 29–51.

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

Cormack, G., Munro, I., Vasiga, T. & Kemkes, G. (2006), 'Structure, Scoring and Purpose of Computing Competitions', *Informatics in Education* **5**(1) 15–36.

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

Horváth, G. & Verhoeff, T. (2002), 'Finding the median under IOI conditions', *Informatics in Education* **1** 73–92.

Opmanis, M. (2006), 'Some Ways to Improve Olympiads in Informatics', *Informatics in Education* **5**(1) 113–124.

Skiena, S. S. & Revilla, M. A. (2003), *Programming challenges: The programming contest training manual*, Springer.

Verhoeff, T., Horváth, G., Diks, K. & Cormack, G. (2006), 'A proposal for an IOI syllabus', *Teaching Mathematics and Computer Science* **4**(1) 193–216.