A Turing Machine Contest for Introducing High School Students to Computer Science

Antonio Brogi
Department of Computer Science
University of Pisa, Italy
brogi@di.unipi.it

ABSTRACT: *This report briefly describes a Computer Science contest for high school students which was recently organized by the Department of Computer Science at the University of Pisa. The goal of the contest was to stimulate the interest of high school students for Computer Science, and to give them a chance of demonstrating and developing their problem solving abilities. A distinguishing aspect of the contest was that no background in Computer Science was required to participate.*

I. Introduction and Motivations

The Department of Computer Science at the University of Pisa offers two undergraduate programs ("Diploma in Informatica", 3 years and "Laurea in Informatica", 5 years) and a graduate program ("Dottorato in Informatica", 4 years).

Every year, the Department organizes a number of events to introduce high school students to Computer Science (CS). Indeed, while computer applications are nowadays present in almost all branches of human activities, the role and specificity of CS is generally only partially perceived by non-experts of the field. For instance, while high school students are generally familiar with Internet applications and video-games, they often under-estimate the amount of mathematics and logic underlying CS.

In order to try to give a more general and faithful view of CS, our Department organizes introductory lectures to illustrate CS curricula and research activities. This year we decided to accompany the seminars with another introductory activity, which could be more enjoyable (though still formative) for high school students: A Computer Science contest.

II. Designing the Contest

The goal of the contest was to stimulate the interest of high school students in CS, and to give them a chance of demonstrating and developing their problem solving abilities.

The contest was designed by following two main criteria:

(1) No background in CS is required to be admitted to the undergraduate programs offered by the Department. Actually recent statistics showed that most of the students who successfully completed their undergraduate studies in CS had no knowledge of CS before starting their University courses. The first criterion was hence to allow all high school students to participate in the contest. No background in CS was required to participate.

(2) Even though the ability of using or programming a computer is only one of the aspects studied in CS, it is natural to expect a CS contest to be run on computers. The second criterion was therefore to run the contest on computers, also for satisfying the curiosity of beginners.

In order to follow the two above criteria, we had to individuate a "language" very easy to understand and to use, and which at the same time could solve a significant class of problems. We found that the formalism of *Turing Machines* satisfies these requirements. On the one hand, the basic behavior of a Turing Machine can be easily understood by people with no knowledge of CS. On the other hand, Turing Machines can be used to solve the largest class of problems which can be solved with a computer language (viz., all computable functions).

After choosing Turing Machines as the "language" for the contest, the first problem to be faced was finding a convenient way of introducing Turing Machines to beginners.

III. Introducing Turing Machines

Many CS books contain a description of Turing Machines, generally presented in the context of computability, complexity or automata theory. From an educational point of view, Turing Machines are not usually employed as the first formalism to introduce CS. Indeed while Turing Machines can be used to code sophisticated programming techniques (like writing an interpreter for the formalism), solving non-trivial problems via Turing Machines can be hard. Moreover, the associated theories introduce non-trivial notions, like for instance the notions of decidability and semi-decidability in computability theory.

In spite of this, the basic behavior of Turing Machines can be explained to beginners in a simple way, as illustrated by the following description. We therefore decided to prepare a short and self-contained introduction to Turing Machines. This material was distributed to the schools together with the presentation of the contest. Moreover, to encourage participation, we organized an introductory seminar on Turing Machines one month before the contest, which was attended by more than 200 students.

What is a Turing Machine?

In 1936 Alan Turing, a British mathematician, proposed the idea of an imaginary machine able to perform all kinds of computations on numbers and symbols.

A Turing Machine (TM) is defined by a *set of rules* which define the behavior of the machine on an input/output *tape*. The tape is divided into (a potentially infinite number of) *cells*. Each cell contains a symbol or it is empty. A TM has a read/write *head* which moves along the tape by reading, writing and erasing symbols in the tape cells. The machine analyzes the tape, one cell at a time, starting from the cell which contains the left-most symbol in the tape.

At each step, the machine reads a symbol from the tape and according to its *internal state*:

(1) It modifies its internal state, and

(2) it writes a symbol on the tape *or* moves its head one cell left or right.

Like a state of the mind of a human being, the internal state of a TM defines the environment in which decisions are taken. A TM can only have a finite number of states.

The behavior of a TM can be programmed by defining a set of quadruples of the form:

(current-state, read-symbol,
new-state, write-symbol/direction).

For example the quadruple `(0, A, 1, B)` states that if the machine is in the internal state `0` and reads symbol `A` from the tape, then it changes its internal state into `1` and writes `B` on the tape. The quadruple `(1, B, 0, >)` instead states that if the machine is in state `1` and reads symbol `B` from the tape, then it changes its internal state into `0` and moves its head one position right.

Notice that a set of quadruples associates with each pair

current-state, read-symbol

at most one pair

new-state, write-symbol/direction.

Let us now see how a TM *computes*. Initially the tape contains a finite sequence of symbols, called *input sequence*. The TM is in its initial internal state, `0` say, with its head on the left-most symbol on the tape. Starting from this initial configuration, the TM performs a sequence of actions by strictly following its set of rules. If the machine reaches an internal state such that there is no quadruple for the pair:

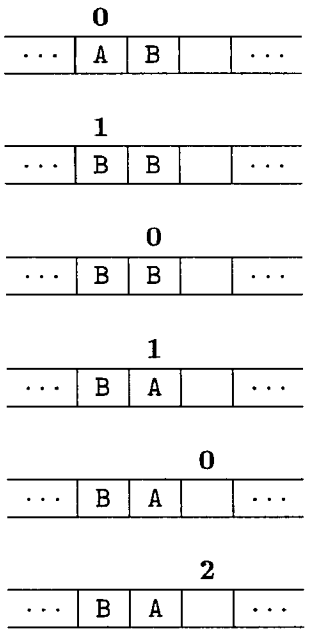
current-state, read-symbol

then the TM halts and terminates its *computation*.

Consider for instance a TM which modifies a sequence of `A` and `B` by replacing each `A` with a `B` and vice-versa. Such a TM can be defined by the following set of rules:

0	A	1	B
0	B	1	A
0	-	2	-
1	A	0	>
1	B	0	>

where `-` denotes the empty cell. If the input sequence is `AB` then the TM performs the following computation:



Notice that a TM may *not* terminate its computation on some input. For instance, the TM:

0	A	1	B
1	B	0	A

will never halt on any sequence of A and B, as it will keep replacing A with B and vice-versa in the left-most non-empty cell.

IV. Turing Machine Simulator

After introducing Turing Machines to beginners, the second problem was to find a Turing Machine simulator for running the contest. We looked for a simulator which was extremely easy to use, and which had some kind of nice interface.

Among several Turing Machine simulators available on the net, we found that the simulator developed by the *Buena Vista University Java Team* [1] offered both these features, its graphical interface being particularly appealing. Indeed the BVU simulator won a USD 75,000 prize in SUN MicroSystems JAVA programming contest last year.

The main obstacle for employing the BVU simulator for our purposes was that it did not allow users to permanently save their programs (i.e., their Turing Machines). With the agreement of BVU, we modified the simulator so as to overcome this limitation and to support multi-user functioning. We also simplified the interface of the simulator by eliminating several functionalities supported in the original version.

The interface of the simulator used for the contest displays a Turing Machine running on the tape and presents:

- Two windows where the set of rules of the Turing Machine and the initial tape can be directly typed in,
- a `LOAD` button for loading the set of rules of a Turing Machine from a list of given machines, and a `SAVE` button for saving changes,
- `RUN`, `RUN-FAST` and `STOP` buttons to put the machine at work.

V. The Contest

The contest was designed as a competition among teams of high school students. Each team consisted of two students, who participated in the contest representing their school. The teams were given a number of problems to be solved by programming Turing Machines on a computer.

The response of the high schools to the call for participation was more than positive. 38 high schools participated in the contest, from eight different provinces of Italy. For the first edition of the contest, we limited participation to 50 teams, and each school was hence represented by one or two teams of students. It is worth observing that only half of the schools participating were schools specializing in technical disciplines. The other half was formed by "Liceo" schools, specializing in classical and scientific studies.

The contest was run on March 17, 1997 at the Computing Labs of the Department of CS of the University of Pisa. The 100 participants were given nine problems to be solved in three hours. The contestants worked on 50 workstations connected by a local network. After the contest, the Judges classified the teams on the basis of the number of problems correctly solved and of the time spent to produce the correct solutions. The solution of a problem was considered correct if the corresponding Turing Machine behaved correctly on a set of test tapes.

A number of companies and institutions sponsored the contest: *ACM (Association for Computing Machinery)*, *Apple Computer*, *the Department of Computer Science at the University of*

Pisa, *DocLine Internet Provider*, *the Province di Pisa*, *Punto Video*, *TEKNOService*, *the University of Pisa*, and *3Com*. The sponsors offered a number of prizes for the schools of the first teams, including personal computers, printers, and funds for developing computing labs. The ACM offered 20 free ACM student memberships to the students of the first 10 teams. Smaller prices and gadgets were offered to all the participants.

VI. Assessment of the Experiment

Organizing a CS contest for high school students was a considerable effort. This experiment suggested a number of observations which we would like to share with other people interested in the issues of Computer Science education.

Choice of the Language

The first observation concerns the choice of Turing Machines as the "language" for the contest.

The ambitious objective of running a CS contest on computers without assuming participants to have some CS knowledge obviously posed severe constraints on the choice of the language. Before choosing Turing Machines, we analyzed several other possibilities such as using some existing educational programming language, like LOGO, or even designing an over-simplified language just for the contest. The main problem was how to give a short and self-contained introduction to these languages, which could be quickly understood by people with no knowledge of CS.

This difficulty led us to choosing Turing Machines, which have the additional advantage of being a really important and powerful formalism in CS. A potential risk of showing Turing Machines as the first example of executable language is that it may give beginners the impression that programming is basically a low-level coding process, far from being an intellectually rewarding activity. On the other hand, teaching beginners how to solve simple problems by programming Turing Machines offers several important benefits from an educational point of view. Students learn to formalize their ideas in an unambiguous language, understand the notion of an "executor" of commands, and autonomously discover the notion of algorithm.

Hands-on-keyboard

Letting contestants use computers distinguishes a CS contest from other "paper-and-pencil" contests, like the *International Mathematics Olympiads* [2].

Indeed, giving the contestants the possibility of writing and running their programs on a computer turned out to be very stimulating for the students, and the importance of this should not be under-estimated. The "reactiveness" of computers is probably the most fascinating aspect of using computers. Getting immediate feedback for what you write is an important aspect of computer uses in education in general.

School Involvement

We believe that one of the main reasons for the large participation of schools was the choice of centering the contest around a topic which could be briefly described in a self-contained way.

This allowed us not to worry about the differences among the CS programmes taught in different types of (some of the) high schools. The self-containedness of the topic also encouraged the involvement of high school teachers in the preparation of the contest. Many teachers devoted several class hours to train their students with Turing Machines. Some schools run local contests to select the school representatives. The enthusiasm and freshmindness of the contestants completed the picture.

Overall assessment

We think that, in general, the first edition of the CS contest for high school students may be considered a successful event. The participation of the schools was higher than expected. Moreover, as each school participated with only one or two teams, the level of the contestants was very high. Many teams correctly solved in three hours most of the posed problems. The problems varied in difficulty, an average-difficulty problem being: *Write a Turing Machine which, given an input sequence S consisting of* `A`*'s and* `B`*'s, replaces S with the sequence* `YES` *or with the sequence* `NO` *depending on whether S is palindrome or not.*

It is interesting to observe that the majority of the best ranked teams (including the first one) were from "Liceo" schools, specializing in classical and scientific studies, where (almost) no CS is taught. The results therefore somehow showed that the initial objective of allowing all students to participate in the contest, independently of their knowledge of CS, had been obtained.

Other Contests

The idea of organizing a CS contest for high school was inspired by the *ACM International Collegiate Programming Contest* [3], the oldest and largest programming competition for universities and colleges. The main difference between the two contests (besides obviously the size) is that the ACM contest relies on the educational experience of the participants, which must be able to program in Pascal, C or C++.

Many other programming contests exist. For instance, the *International Olympiads in Informatics* [4] is another international programming contest for high school students. Some other programming contests feature different categories to accommodate different computing abilities and interests of students. One example is the computer programming contest for junior and senior high schools students organized by the *American Computer Science League* [5]. Another example is the *Calgary Region Computer Programming Competition* [6], which includes creative programming, problem solving and computer art competitions.

All these contests differ from our Turing Machine contest in that they are *programming* contests which rely on the educational experience in CS of the participants.

Acknowledgements

I would like to thank here all the people who helped me in organizing and running the contest. Thanks go to the heads of the high schools who promoted the event, to the teachers who prepared the contestants, and obviously to the students who participated with enthusiasm in the contest. Thanks are due to the sponsors who decided to substain and offer the prizes for the contest. I would like to thank the Head of our Department, F. Turini, and the Head of our Computing Labs, F. Romani, for having encouraged and supported the contest since the very beginning. Thanks also to K. Schweller, for letting us modify the BVU Simulator and use it for the contest, and to E. Bernardini, A. Bertanza, A. Cisternino, and G. La Vecchia for their technical support. Finally I would like to thank the members of the Judging Committee I had the honor to chair, R. Barbuti, P. Mancarella, E. Occhiuto, and L. Pagli, who greatly contributed to preparing and running the contest.

WWW References

[1] BVU Turing Machine simulator
`http://www.bvu.edu/faculty/schweller`
[2] International Mathematics Olympiads
`http://www.win.tue.nl/ioi/imo`
[3] ACM Int.nal Collegiate Programming Contest
`http://www.acm.org/contest`
[4] International Olympiads in Informatics
`http://www.win.tue.nl/ioi`
[5] American Computer Science League
`http://www.acsl.org`
[6] Calgary Computer Programming Competition
`http://www.cps.enel.ucalgary.ca`

***********Software From Page 16***********

to an improved understanding of the software development process, and improved student satisfaction. The primary costs are associated with acquiring the necessary hardware and software to support the proposed curriculum changes.

References

[1] R.J. Daigle, Michael V. Doran, J. Harold Pardue. "Integrating Collaborative Problem Solving Throughout the Curriculum". Proc. SIGCSE '96, ACM, pp. 237-241, 1996.

[2] Peter Denning, et.al., Computing as a Discipline. Communications of the ACM, 32/1, pp. 9-23, 1989.

[3] Alan Fekete, Antony Geening. "Designing Closed Laboratories for a Computer Science Course". Proc. SIGCSE '96, ACM, pp. 295-299, 1996.

[4] Joseph E. Lang, Barbara A. Smith. "Scheduled Supervised Laboratories in CS1: A Comparative Analysis". SIGCSE Bulletin 25/1, pp. 185-188, 1993.

[5] Mei-Ling Liu, Lori Blanc. "On the Retention of Female Computer Science Students". Proc. SIGCSE '96, ACM, pp. 32-36, 1996.