Department of Computer Science
Rhodes University
Honours Project Proposal

A Taxonomy of Algorithms used in the ACM Programming Competition

Student: Douglas Hobson (04h2708)
Supervisor: A.J. Ebden

1. Project Specification/Objective of Research

It has been observed that the problems given each year in the ACM Programming Competition seem to have been drawn from the same set of algorithms. The purpose of this research is to derive all of or part of this set of algorithms, solve as many of them as possible and then derive a classification of the algorithms.

2. Background:

Every year students from Rhodes University take part in the ACM Programming Competition. Last year's team, had they won on a regional level, would be getting ready to leave or have left already for the international final in Tokyo Bay on 12 – 16 March. The ACM Programming Competition is a large event on calendars all over the world. [ACM ICPC, 2007]

Algorithmics is central to computer science. According to David Harel [Harel, 1987] the name of this subject would be more accurate if it were changed to 'process science' or something else to more closely reflect that which he believes underpins it: algorithmics.

An algorithm is a recipe or process describing how to perform a certain task. The recipe for baking a cake is an algorithm, even though one would not normally call it that. Humans naturally use algorithms (some of them are very simple but they are algorithms none the less) for many processes without thinking about it, multiplication of large numbers, tying shoelaces or replacing a car tire. The algorithms used in computer science are usually a lot more abstract and complicated than these

everyday examples but ultimately they are of similar nature, just a recipe for how to accomplish a certain task.

3. Resources:

3.1 Literature Review

The book "*Algorithmics: The Spirit of Computing*" by David Harel will be one of my principal texts. It deals with many of the basic principles of algorithms that will be highly relevant to this project.

Another book by Michael Garey and David Johnson called "*Computers and Intractability: Guide to the Theory of NP-Completeness*" will also be useful. It deals with analysis of algorithms, looking for various qualities that determine whether or not the algorithm is solvable.

The ACM website and the ACM digital library will be of some use if I can find papers pertinent to my work. URL: http://www.acm.org

3.2 Other Resources

The local and international ACM Programming Competition websites will be very useful to me as both have many problem sets (more specifically, they have the problem sets that I will be looking at) and statistics about the competition.
Local URL: http://acm.cs.up.ac.za
International URL: http://icpc.baylor.edu/icpc/

4. Design and Implementation:

41. Timeline

Activity	Time Required
Analyse all ACM problems from the last 6 years. This will be done to try and find patterns and similarities among the problems.	1 month
Select problems to start formulating and verifying algorithms.	1 month
Program solutions to selected problems in C, C++ or Java in order to validate/verify the algorithm type/suitability of algorithm.	4 months
Analyse results of programming attempts.	2-3 weeks
Write up the project.	1 month

4.2 Example of an ACM problem

Third Southern African Regional ACM Collegiate Programming Competition (2001)
Problem 4 – Blue Balloon
Alien Security

You are in charge of security at a top-secret government research facility. Recently your government has captured a live extra-terrestrial (ET) life form, and is hosting an open day for fellow researchers. Of course, not all the guests can be trusted, so they are assigned different security clearance levels. Only guests with a level 5 rating will be allowed into the lab where the extra-terrestrial is being held; other than that, everyone is free to roam throughout the rest of the facility. Each room in the facility is connected via one-way airlocks, so that you can pass through the door in only one direction.

To protect your precious ET you will put in place enhanced security measures (in the form of armed guards) on the route leading to the room containing the ET, but not in the room itself – the guards do not have sufficient clearance to enter the room containing the ET.

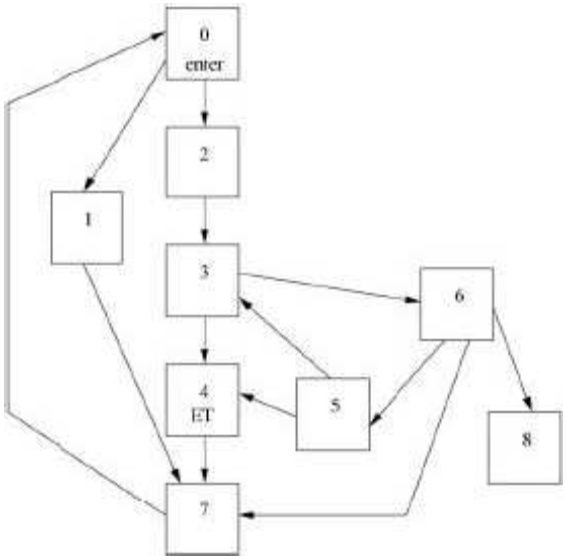
The guards will check the identity and the security rating of all guests trying to pass through the room in which they are stationed, so you would like to place the guards where they will cause the

minimum amount of irritation to the guests who have no intention of visiting the ET. The room where the guards must be placed thus satisfies the following two conditions:

1. In order to get to the room containing the ET, the guests must pass through the room containing the guards;

2. There is no other room with this property that is closer to the room containing the ET – remember, the guards cannot be placed in the room containing the ET itself.

The diagram below illustrates one possible map of your facility:



Note that placing the guards in room 2 would satisfy the first condition, but room 3 is closer to the ET, so the guards must be placed in room 3.

All guests enter through room 0, the entrance to your facility. Your program accepts a sequence of lines containing integers. The first line consists of two integers: the number of rooms, and the room in which the ET is being held (out of his own free will, of course).
The rest of the input is a sequence of lines consisting of only two integers, specifying where the airlock-doors are located. The first number on these lines specifies the source room, the second the destination room. Remember: you can pass only from the source room to the destination room.

The output of your program consists only of a single line:
Put guards in room N.
where N is the room you've decided to place the guards.

SAMPLE INPUT

This input sequence specifies the map shown above.

9 4

0 2

2 3

3 4

5 3

5 4

3 6

6 5

6 7

6 8

4 7

0 1

1 7

7 0

SAMPLE OUTPUT

Put guards in room 3.

5. Deliverables:

The final product of this project is to be a list of algorithms appearing in the ACM Programming Competition along with their frequency of appearance and level of difficulty for humans and machines. (i.e. how difficult is it to recognise and apply the algorithm and what is its big O notation.)

6. Possible Extensions:

a) A tutorial could be developed on how to approach the ACM Programming Competition.

b) A method of ranking algorithms for their level of difficulty for humans.

c) Move on to analyse and formulate algorithms from the board game Go [Sensei's Library, 2007]. The very best go playing programs are no match for a reasonably competent human. [Sensei's Library, 2007] This presents an interesting problem from an algorithmic and AI point of view that could easily become the basis of a masters or doctoral thesis.

7. References

Harel, David, *Algorithmics: The Spirit of Computing*, Addison-Wesley, Great Britain, 1987, pp 6-7

ACM International Collegiate Programming Contest website, Date of access: 2007-03-06, Date of publication: unknown, URL: http://icpc.baylor.edu/icpc/

Sensei's Library, Date of access: 2007-03-05, Date of publication: unknown. URL: http://senseis.xmp.net/?WhatIsGo