EECS 477. PROJECT

12/11/2002 is the new due date

DUE ~~12/9/2002~~ BY 11:59PM

Suppose that you have a successful tutoring business. At the beginning of the year you obtain a list of students and their schedules. Each student is assigned a single tutor, but a tutor may have multiple students under his/her supervision. Each tutoring session is one-on-one and runs for the whole day. You would like to determine how many tutors you should employ this year, in fact, you would like to minimize number of tutors.

The lesson schedules is given in a text file. Each line contains days on which a particular student has lessons. An example is given below:

```
1 2 6 9
2 3 4
4 8 9
6 7 8
```

Note that even though there are no more than two lessons on each particular day, we need three different tutors.

In the project, you will need to write a program that reads the schedule file of the type described above, and returns the minimal number of tutors needed to cover that schedule, together with the corresponding tutor-student assignments.

You may assume that every line contains days in increasing order. There is no a priori limits on the number of days or number of students in the schedule.

Before you start programming, formalize the task as a graph coloring problem. Thus, the minimal number of tutors will correspond to the chromatic number of the graph. Describe an exact algorithm based on exhaustive enumeration, and analyze its memory and time complexities. Describe an improved algorithm based on branch-and-bound and analyze its time and memory complexity (why is it improved?). Describe a further improvement for your exact algorithm that includes dynamic programming. Analyze its time and memory complexities. Describe a couple of heuristics for this problem and analyze their time and memory complexities.

All code must work with g++ 2.95.* or g++ 3.0.* on CAEN Solaris or Linux systems, or with Microsoft Visual Studio 6 (or .net).

Write a helper program that generates random graphs and random schedules. Produce examples that are good for testing and debugging.

Implement a front-end that extracts graph from a given schedule, and given a vertex coloring prints tutor-student assignment. It should verify that the generated assignment does not conflict with the schedule.

Implement a back-end that solves graph coloring problem according to the algorithms you designed.

Empirically evaluate your algorithms in terms of memory and runtime requirements on randomly generated inputs. Can you construct easy and difficult input yourself? Come up with difficult inputs to test your program.

How large are the instances that can be solved with your optimal algorithm within a minute? How does the runtime/memory grow with the number of lines? Is this consistent with your theoretical complexity analysis? If you apply your optimal algorithms to the same problem instance, do they always return solutions of the same cost? Do they always return the same solutions?

How much faster are your heuristics than your optimal algorithms? Does the advantage increase for larger inputs? How often do your heuristics return optimal solutions? What is the [geometric] average sub-optimality ratio?

Illustrate your experimental results with tables and plots.

How would you combine your optimal algorithms with heuristics? Demonstrate that this improves performance in practice.

Submission guidelines. Tar and zip (or winzip) your project files into a file called `<uniqname>477.tar.gz` (or `<uniqname>477.zip`). Post this file on the Web and email the URL to `guskov@umich.edu`. Please make sure that you have the file permissions set correctly and the file is downloadable.

If you do not know how to set up a Web page or how to use tar or gzip or winzip, contact CAEN help desk. Make sure to do this well before the deadline

Make sure that your .tar.gz file includes the following:

(1) A README file describing the files in your submission
(2) All the source code, it should compile within environments specified above (Makefile or .dsp files would be helpful)
(3) DO NOT include executables or object files
(4) A subdirectory with at least 10 examples on which program was tested
(5) A project report called report.pdf, report.ps, or report.txt (Word/Excel files are discouraged and will not be sufficient). The project report needs to contain tables and plots which describe the performance of your implementations. We should be able to reproduce all data points using the supplied code.

Sending graphics in separate files is not recommended. If you do not know how to create .pdf or .ps files, contact the CAEN help desk. Make sure to do this well before the deadline.