BLG 372E - Analysis of Algorithms, Spring 2013

Project 1

(Please read report and submission part carefully)

Total Worth : 10% of your grade

Handed Out : 07.03.2013 Thursday 10:30

Due : 21.03.2013 Thursday 23:00

Task 1-Residency Matching

The Residency Matching algorithm pairs medical school graduates(residents) and residency slots at teaching hospitals. Residents and hospitals submit their ordered preference list, and the stable pairing produced by matching residents with residency programs, i.e. a set of residents has to be assigned to a set of hospitals. The ingredients of a hospital admissions problem are as follows:

- a set of residents(*r*) ,
- a set of hospitals(*h*) each with a quota, i.e., the maximal number of residents it can admit,
- residents strict preferences over hospitals
- hospitals' responsive preferences over sets of residents

Matching S is unstable if there is a hospital *h* and resident *r* such that:

- *h* and *r* are acceptable to each other; and
- either *r* is unmatched, or *r* prefers *h* to her assigned hospital; and
- either *h* does not have all its places filled, or *h* prefers *r* to at least one

You are required to write a program to solve Residency Matching problem. Determine the running time of your algorithm with preference profiles for residents given in data1_r[1..4].txt and for hospitals given in data1_h[1..4].txt with hospital quotas 3, 5 and 10 respectively (Assume each hospital quota is same). Report stable matching *S* to output file output1_S[1..4].txt with indicating the number of quota at the beginning of each match list *S* .

output1_S1.txt

```
Quota= 3
1 3
2 9
3 4
..
..
Quota= 5
1 9
2 3
3 1
..
```

Report your running times in a table given in the following format:

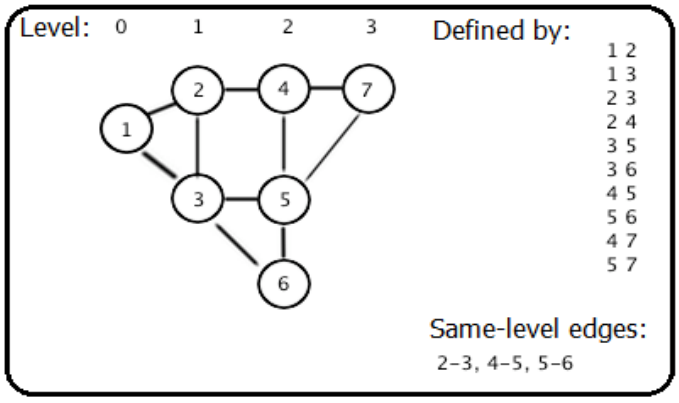
	3	5	10
data1_{h,r}1			
data1_{h,r}2			
data1_{h,r}3			
data1_{h,r}4			

Task 2-Not Quite Tree

In a graph, each node has a *level* representing distance to the root node. One of the structural property that could be extracted is cycle. If there is a connection between two nodes of the same level, then there is also a cycle of odd length in the graph, which gives us more properties about the structure.

The input file **data2.txt** will contain five sets of input. A single positive integer 1 <= N < 50, followed by N lines describing the graph. In each line there exist two integers, id's of nodes, separated by a space. Node id's are positive integers less than 100. The root node has id 1.

The output file **output2.txt** of the program should contain five lines, integer count of how many pairs of nodes have a connection, such that the shortest path from 1 to each node is equal.



Sample Input (first 2 sets shown):	Sample Output (first 2 sets shown):
3 1 2 3 2 1 3 10 1 2 1 3 2 3 2 4 3 5 3 6 4 5 5 6 4 7 5 7	1 3

Report, Submission

You should be aware that the Ninova system clock may not be synchronized with your computer, watch, or cell phone. Do not e-mail the teaching assistant or the instructors your submission after the Ninova site submission has closed. If you have submitted to Ninova once and want to make any changes to your report, you should do it before the Ninova submission system closes. Your changes will not be accepted by e-mail. Connectivity problems to the Internet or to Ninova in the last few minutes are not valid excuses for being unable to submit. You should not risk leaving your submission to the last few minutes. After uploading to Ninova, check to make sure that your project appears there.

Policy: You may discuss the problem addressed by the project at an abstract level with your classmates, but you should not share or copy code from your classmates or from the Internet. You should submit your own, individual project. Plagiarism and any other forms of cheating will have serious consequences, including failing the course.

If a question is not clear, please let the teaching assistant know by email (nilhan@itu.edu.tr).

Submission Instructions: Please submit your homework through Ninova.

Please zip and upload all your files using filename **HW1_ studentID.zip**. In the zipped file, you must include your completed **report_StudentId** file, a folder for Task 1 named as "**Task1**" and a folder for Task 2 named as "**Task2**". Test your programs with .txt files given to you (data1_r[1..4].txt, data1_h[1..4].txt, data2.txt).

Your compilation code will be:

Task 1: g++ program_name.cpp –o program_name
Task 2: g++ program_name.cpp –o program_name

Running command will be:

Task 1: ./program_name file_name1 filename2 quota_number
Task 2: ./program_name file_name

Example run for task1:
g++ task1.cpp –o task1
./task1 "data1_h1.txt" "data1_h1.txt" 3

In your report, please explain your approaches, your algorithms and then give the time complexity of your algorithms. Give the data structures you have built and explain them. Give the compilation command of your program. Be sure that your program can be run with the running command given above. Otherwise, you may get a grade of zero.

All your code must be written in **C++,** and we should be able to compile and run on it on Linux/Unix using **g++**. When you write your code, try to follow an object-oriented methodology with well-chosen variable, method, and class names and comments where necessary. Your code must compile without any errors; otherwise, you may get a grade of zero on the assignment.