Due: 02.04.2012, Monday, 23:00

BLG 372E - Analysis of Algorithms, Spring 2012

Project 1 – Virus

(Please read this paper carefully)

Scenario:
You're helping some security analysts monitor a collection of networked computers, tracking the spread of an online virus. There are n computers in the system, labeled C1, C2, ...,Cn, and as input you're given a collection of *trace data* indicating the times at which pairs of computers communicated. Thus the data is a sequence of ordered triples *(Ci, Cj, tk*); such a triple indicates that *Ci* and *Cj* exchanged bits at time tk. There are *m* triples total.

We'll assume that the triples are presented to you in sorted order of time. For purposes of simplicity, we'll assume that each pair of computers communicates at most once during the interval you're observing.

The security analysts you're working with would like to be able to answer questions of the following form: If the virus was inserted into computer *Ca* at time *x*, could it possibly have infected computer *Cb* by time *y*? The mechanics of infection are simple: if an infected computer *Ci* communicates with an uninfected computer *Cj* at time *tk* (in other words, if one of the triples (*Ci, Cj, tk*) or (*Cj, Ci, tk)* appears in the trace data), then computer *Ci* becomes infected as well, starting at time *tk*. Infection can thus spread from one machine to another across a *sequence* of communications, provided that no step in this sequence involves a move backward in time. Thus, for example, If *Ci* is infected by time *tk*, and the trace data contains triples *(Ci, Cj, tk) and (Cj, Cq, tr)*, where *tk <= tr, then Cq will* become infected via *Cj*. (Note that it is okay for *tk* to be equal to *tr;* this would mean that *Cj* had open connections to both *Ci* and *Cq* at the same time, and so a virus could move from *Ci* to *Cq.)* For example, suppose n = 4, the trace data consists of the triples

(C1, C2, 4), (C2, C4, 8), (C3, C4, 8), (Cl, C4, 12)

and the virus was inserted into computer *C1* at time 2. Then *C3* would be infected at time 8 by a sequence of three steps: first *C2* becomes infected at time 4, then *C4* gets the virus from *C2* at time 8, and then *G3* gets the virus from *C4* at time 8. On the other hand, if the trace data were

(C2, C3, 8), (C1, C4, 12), (C1, C2, 14)

and again the virus was inserted into computer C1 at time 2, then C3 would not become infected during the period of observation: although *C2* becomes infected at time 14, we see that *C3* only communicates with *C2 before C2* was infected. There is no sequence of communications moving forward in time by which the virus could get from *C1* to *C3* in this second example.

Aim:
Design an algorithm that answers questions of this type: given a collection of trace data, the algorithm should decide whether a virus introduced at computer *Ca* at time *x* could have infected computer *Cb* by time *y*. If yes, give the trace of infection.

Hint for building graph:

For each triple (Ci, Cj, tk) we can see in our scan, we create nodes (Ci, tk) and (Cj, tk), we create directed edges joining these two nodes in both directions. If this is not the first node involving Cx (for all x from 1 to n), then we include a directed edge from (Cx, tx) to (Cx, tk), where tx, tk are the timestamps and tx < tk.

Implementation notes:

Use one of the algorithms that you have learned in the Analysis of Algorithms class. All your code must be written in C++ and compile and run on linux/unix using g++. You have to use standard libraries. Do not forget to add necessary headers. 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. Please obey the format introduced below.

A txt file is going to be given to you. Your code should be compiled and run by the statements below where **the running command takes 3 arguments** which are hold for respectively, "the input file name", "which computer is infected first", and "when it is infected". Please keep the .cpp filename as main.cpp with just one cpp file or give the compilation command in your reports briefly. The running command is **strict**, you should not change it.

Your compilation code will be: g++ main.cpp –o main
Running command will be: ./main file_name computer_no timestamp

Example run:

```
g++ main.cpp –o main
./main "computers.txt" 1 10
```

Format of input file:
Example "computers.txt" file is like below where first column holds first computer, second column holds for second computer, and the third column holds timestamp which shows the interaction of first and second computer.

1	2	4
2	4	8
3	4	8
1	4	12
2	3	14
4	5	15

Format of program:
The program shows a menu like below:

1. List the computers which can be infected?
2. Enter a computer no in order to test if it is infected?
3. Exit

Your choice:

Example run for the file given as an example above by the given running command (./main "computers.txt" 1 10**)**:

```
1. List the computers which can be infected?
2. Enter a computer no in order to test if it is infected?
3. Exit
 Your choice: 1

Computers can be infected by the trace (computer_no      timestamp):
1    10
4    12
5    15

1. List the computers which can be infected?
2. Enter a computer no in order to test if it is infected?
3. Exit
 Your choice: 2

Please enter the no of computer: 3
Is infected? No

1. List the computers which can be infected?
2. Enter a computer no in order to test if it is infected?
3. Exit
 Your choice: 2

Please enter the no of computer: 4
Is infected? Yes
The trace (computer_no      timestamp):
1    10
4    12
```

In your reports:
In your reports, 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 for your program.

Scoring:
70 points for your program:
- Data structure and building graph : 15 points,
- Algorithm(according to running time and suitability): 15 points,
- Correct run for several tests: 15 points, 15 points
- Presenting as explained: 10 points,

30 points for your reports.

Do not forget that your code will be tested with several different input files. The total point will constitute %10 of your overall grade.

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**.

Submission Instructions:
Please submit your assignment through Ninova. Please zip and upload all your files using filename Pr1_ studentID.zip. In the zipped file, you must include your report and your entire program.

If a question is not clear, please let the teaching assistant Meryem Uzun-Per know by e-mail (uzunper@itu.edu.tr).