Data Structure and Algorithm I

Homework #4

Due: 5pm, Sunday, December 12, 2010

Submit the answers for problem 2-5 through the CEIBA system (electronic copy) or to the TA in R432 (hard copy). You also need to submit the answers of problem 1 through the CEIBA system. If you choose to submit your writing problems through CEIBA,

1. please combine the answers of all writing problems into only one file in the doc/docx or pdf format, e.g., hw4_writing.pdf; otherwise, you will only get the score of one problem (the one that the TA chooses);

2. please submit your programming report together with your source codes (in the zip archive); if the programming report is submitted with the answers of the writing problems, you will NOT get the score for the report; and

3. please write down your name and school ID in the header of both of the above two documents, e.g., b98902xxx Your_Name

Problem 1. (40%) In section 6.3 of the textbook, we introduce minimum cost spanning trees of an undirected graph and its algorithms. In this problem, you are required to write a program that implements Sollin's algorithm using the C programming language. Your program should take the input from the standard input device (stdin) and please use the following input/output format:

Input format:

The first input line is the number of vertices, n

The following input lines represent the edges with their costs.

For Figure 1, the input would be

```
7 --> the number of vertices, n
0 5 10 --> an edge between vertex 0 and vertex 5 with cost 10
0 1 28
5 4 25
```

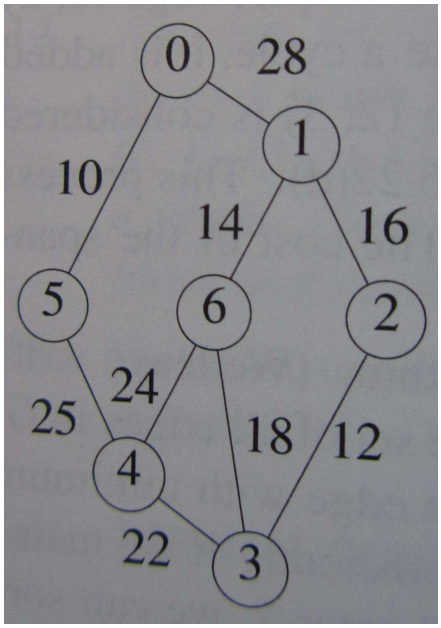

Figure 1: Graph

```
4 6 24
4 3 22 --> an edge between vertex 4 and vertex 3 with cost 22
3 2 12
3 6 18
6 1 14
1 2 16
```

Output format:

Two Parts. Please refer to Figure 2.

1. The first part outputs all stages when executing Sollin's algorithm. Each stage occupies a line.

2. The second part outputs the sequence of the Depth-first search of the minimum spanning tree using Sollin's algorithm starting from node 0.

Example:

```
(0,5,10)(1,6,14)(4,3,22)(3,2,12)  // The first line : added edge at stage(a) with cost
(5,4,25)(1,2,16)                  // the second line: added edge at stage(b) with cost
```

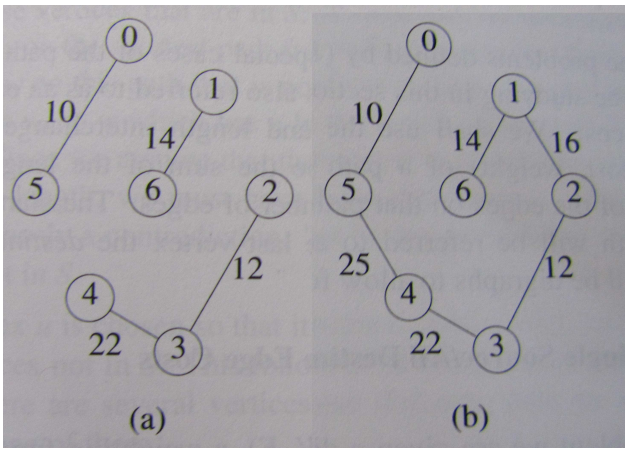

Figure 2: Stages of Sollin's Algorithm

```
0 5 4 3 2 1 6                    // Depth-first search of MST starts from node 0.
```

All numbers (the number of vertices, vertices id numbers, and edge costs) given in the input can be stored in 32-bit integer variables in the C programming language.

You must upload your homework in the format of a compressed zip file to the CEIBA, and the zip file should include the following three files:

1. The source code (.c file),
2. A shell script to compile the source (.sh), and
3. A document in PDF format to describe how your program/algorithm works.

Your score of 40% is divided into two parts: part 1 (6%) and part 2 (2%) (with 4 test cases, 32% in total) and explanations in the document (8%).

Problem 2. (20%) The square of a directed graph $G = (V, E)$ is the graph $G^2 = (V, E^2)$ such that $(u, v) \in E^2$ *iff* G contains a path with at most two edges between u and v. Describe an efficient algorithm for computing G^2 from G when using the adjacency list representation of G. Analyze the running time of your algorithm with the big-oh notation.

Problem 3. (20%) In the class, we have discussed how to perform a depth first search or a breadth first search on an undirected graph. In this problem, we consider performing depth first search on a digraph.

We can define four edge types in terms of the depth first search forest produced by a depth first search on the digraph G:

(a) **Tree edges** are edges in the depth first forest.

(b) **Back edges** are those edges $< u, v >$ connecting a vertex u to an ancestor v in a depth first tree. (self-loops is considered to be back edges)

(c) **Forward edges** are those edges $< v, u >$ connecting a vertex u to a descendant v in a depth first tree.

(d) **Cross edges** are all other edges. They can go between vertices in the same depth first tree, as long as one vertex is not an ancestor of the other, or they can go between vertices in different depth first trees.

We use the following pseudo-code functions to perform a depth first search:

```
DFS(G)
  for each vertex u ∈ V(G)
      u.color=WHITE
  time=0
  for each vertex u ∈ V(G)
    if u.color==WHITE
       DFS-VISIT(G,u)


DFS-VISIT(G,u)
   time=time+1
   u.d=time
   u.color=GRAY
   for each edge < u, v >
      if v.color==WHITE
         DFS-VISIT(G,v)
   u.color=BLACK
   time=time+1
```

`u.f=time`

Explain why an edge $< u, v >$ is

a. a tree edge or forward edge *iff* $u.d < v.d < v.f < u.f$,

b. a back edge *iff* $v.d <= u.d < u.f <= v.f$, and

c. a cross edge *iff* $v.d < v.f < u.d < u.f$.

Problem 4. (20%) Consider digraph G shown in Figure 3.

1. Use the Dijkstra algorithm described in section 6.4.1 in the textbook to determine the shortest path from vertex a to all other vertices in G. Please generate a table similar to Figure 6.28 in the textbook in order to explain your answer in a step-by-step manner.

2. Use the Bellman-Ford algorithm described in section 6.4.2 in the textbook to determine the shortest path from vertex a to all other vertices in G. Please generate a table similar to Figure 6.31 in the textbook in order to explain your answer in a step-by-step manner.

Problem 5. (bonus problem: +20%) Prove that Prim's algorithm finds a minimum cost spanning tree for every undirected connected graph.

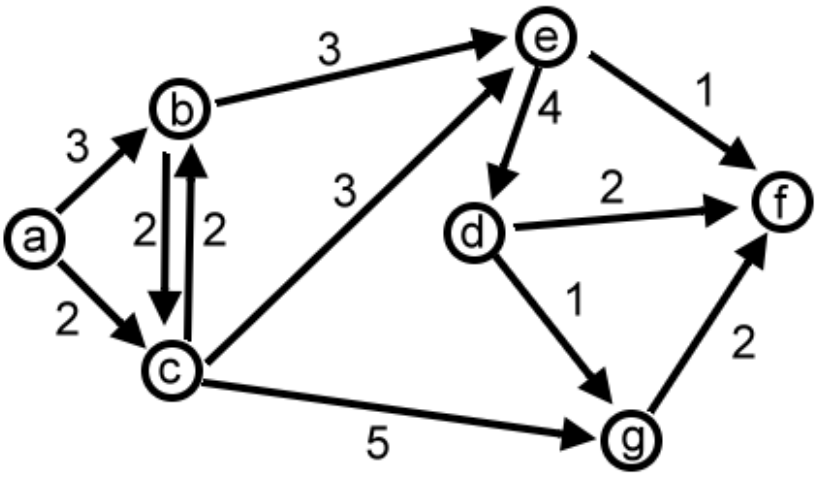

Figure 3: Digraph G