EC 504 – Spring 2023 – Homework 4

Due Nov. 9, 2023 on grayscope by 11:59PM
Reading Assignment: CLRS Chapters 25, 26 and Appendix D.1

1. (20 pts) Answer True or False to each of the questions below, and explain briefly why you think the answer is correct..

   (a) Bellman-Ford algorithm can be used for finding the longest path in a graph.

   (b) In Bellman-Ford with a negative if you keep iteration the outer loop all distances will go toward minus infinity.

   (c) In Floyd's algorithm for finding all-pairs shortest paths, after each major outer loop k, upper bounds are computed on shortest distances $D(i,j)$ between each pair of nodes i,j. These upper bounds correspond to the shortest distance among all paths between nodes i and j which use only intermediate nodes in $\{1,\ldots,k\}$.

   (d) The one to all distance algorithm use relaxation, iterating $d(j) = MIN[d(j), d(j) + w(i,j)]$. MIN and $+$ obey the disctributive law $(a+b)*c = a*c+b*c$ where we have replaced $+$ by MIN and $+$ by $*$.

   (e) Consider a directed, weighted graph where every arc in the graph has weight 100. For this graph, executing Dijkstra's algorithm to find a shortest path tree starting from a given node is equivalent to performing breadth first search.

   (f) In Dijkstra's algorithm, the temporary distance labels D assigned to nodes which have not yet been scanned can be interpreted as the minimum distance among all paths with intermediate nodes restricted to the nodes already scanned.

   (g) The worst case complexity of the fastest minimum spanning tree algorithm is $O(Nlog(N))$, where N is the number of nodes in the graph.

   (h) Consider an undirected graph where, for every pair of nodes i,j, there exists a unique simple path between them. This graph must be a tree.

   (i) Consider a minimum spanning tree T in a connected undirected graph (N, A) which has no two arcs with equal weights. Then, an arc i,j in T must be either the minimum weight arc connected to node i or the minimum weight arc connected to node j.

   (j) The following three algorithms are examples of greedy algorithms: Dijkstra's shortest path algorithm, Kruskal's minimum spanning tree algorithm, Prim's minimum spanning tree algorithm.

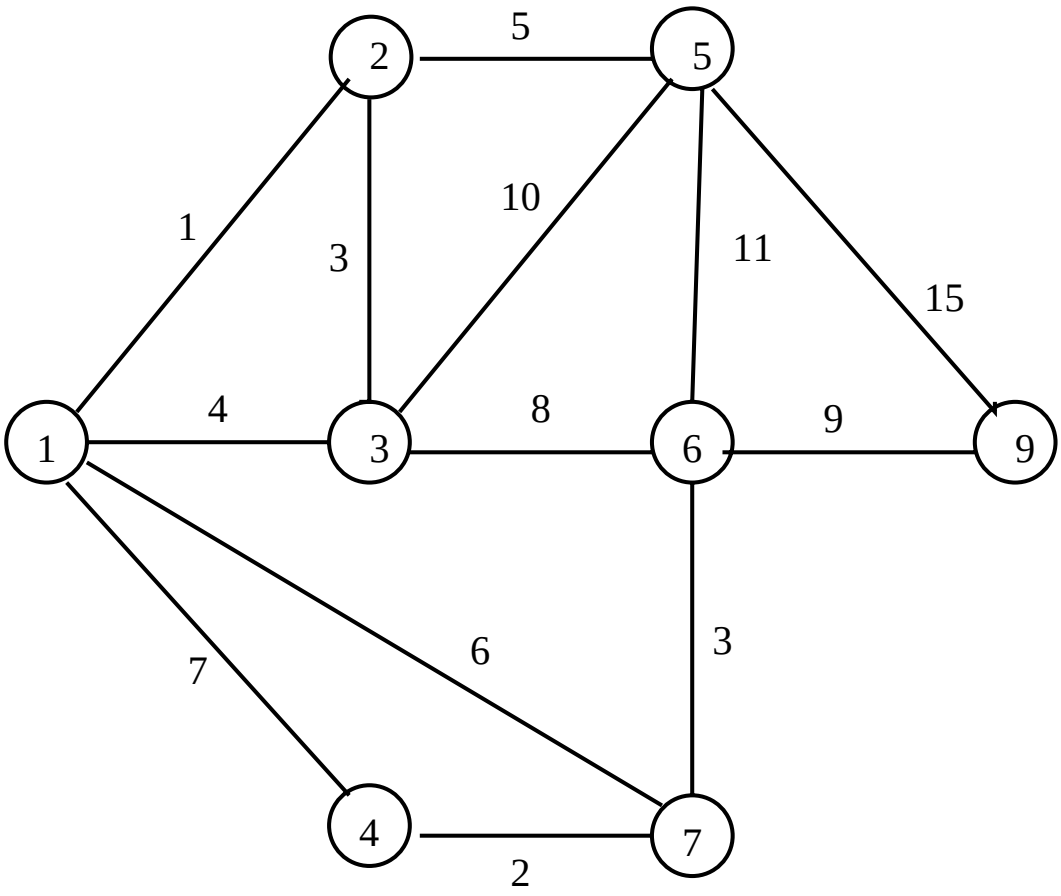

Figure 1:

2. (15 pts) Consider the undirected weighted graph in Figure 1. Show the distance estimates computed by each step in the outer loop of the Bellman-Ford algorithm for finding a shortest path tree in the graph, starting from node 1 to all other nodes. **Note the other loop in the Bellman-Ford algorithm adds an arc so you should give a table recording each new distance $d(j)$ and new predecessor $p(j)$ for each node.**

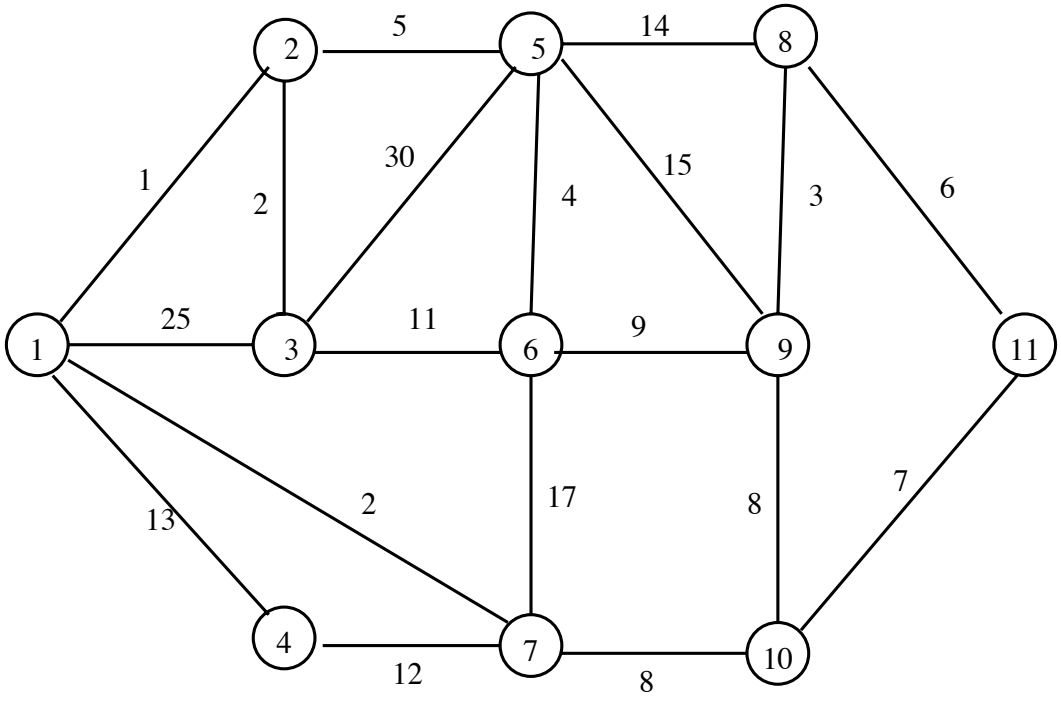

Figure 2:

3. (15 pts) Consider the weighted, undirected graph in Figure 2. Assume that the arcs can be traveled in both directions. Illustrate the steps of Dijkstra's algorithm for finding a shortest paths from node 1 to all others.

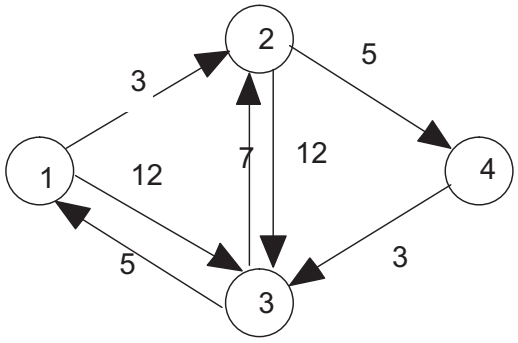

Figure 3:

4. (15 pts) Consider the graph in Figure 3. This is a directed graph. Use the Floyd-Warshall algorithm to find the shortest distance for all pairs of nodes. Show your work as a sequence of 4 by 4 tables.

5. (15 pts)In city streets, the length of an arc often depends on the time of day. Suppose you have a directed graph of streets connecting nodes that represent intermediate destinations, and you are given the travel time on the arc as a function of the time at which you start to travel that arc. Thus, for arc e, you are given $d_e(t)$, the time it takes to travel arc e if you start at time t. These travel times must satisfy an interesting ordering property: You can't arrive earlier if you started later. That is, if $s < t$, then $s + d_e(s) <= t + d_e(t)$. Suppose that you start at time 0 at the origin node 1. Describe an algorithm for computing the minimum time path to all nodes when travel times on arcs are time dependent and satisfy the ordering property.

6. (20 pts) This is a forward star representation for a directed graph with $|V| = 11$ vertices and $|E| = 16$ edges.

```
Vertex Number:   1   2   3   4   5   6   7   8   9  10  11  12
Array First:   { 1,  3,  4,  5,  7,  8, 12, 12, 14, 14, 15, 17 }


Edge Number:  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
Array Edge: { 2,  6,  6,  7,  3,  7,  8,  5,  8,  9, 10,  9, 11,  9,  9, 10, -1 }
```

(a) Draw the graph on the template in Fig. 6. (HINT: You may want to do part b first.)

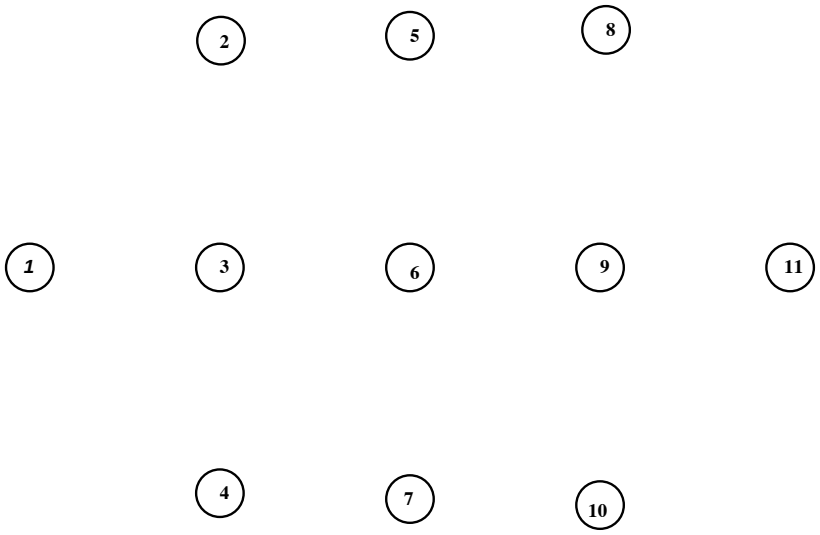

Figure 4:

(b) Represent this graph as an adjacency list.

(c) Is this graph a DAG?