EC 504 – Spring 2023 – Homework 4

Due May 5, 2023 on gradescope 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) In the Bellman-Ford algorithm, the maximum number of times that the distance label of a given node i can be reduced is less than or equal to $n-1$, where n is the number of nodes in the graph.

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

   (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) 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.

   (e) Consider a directed, capacitated graph, with origin node O and destination node D. The maximum flow which can be sent from O to D is equal to the minimum of the following two quantities: the sum of the capacities of the arcs leaving O, and the sum of the capacities of the arcs entering D.

   (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.

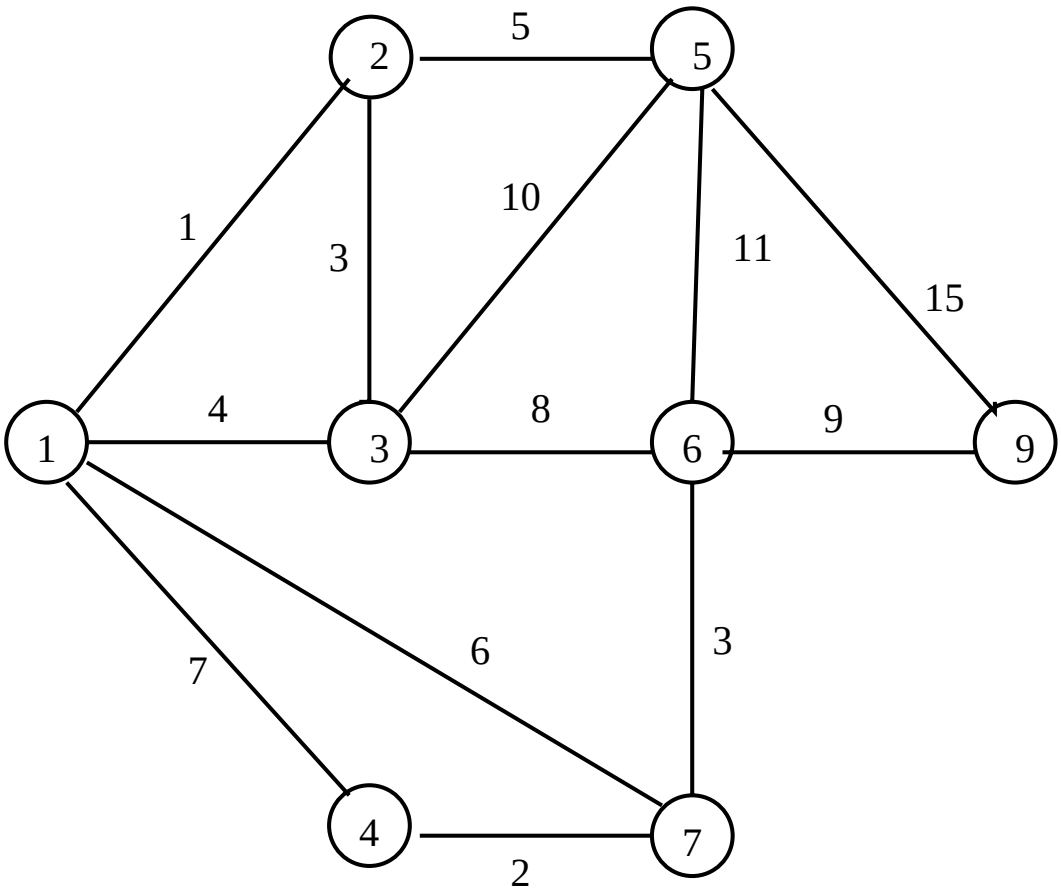

Figure 1:

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

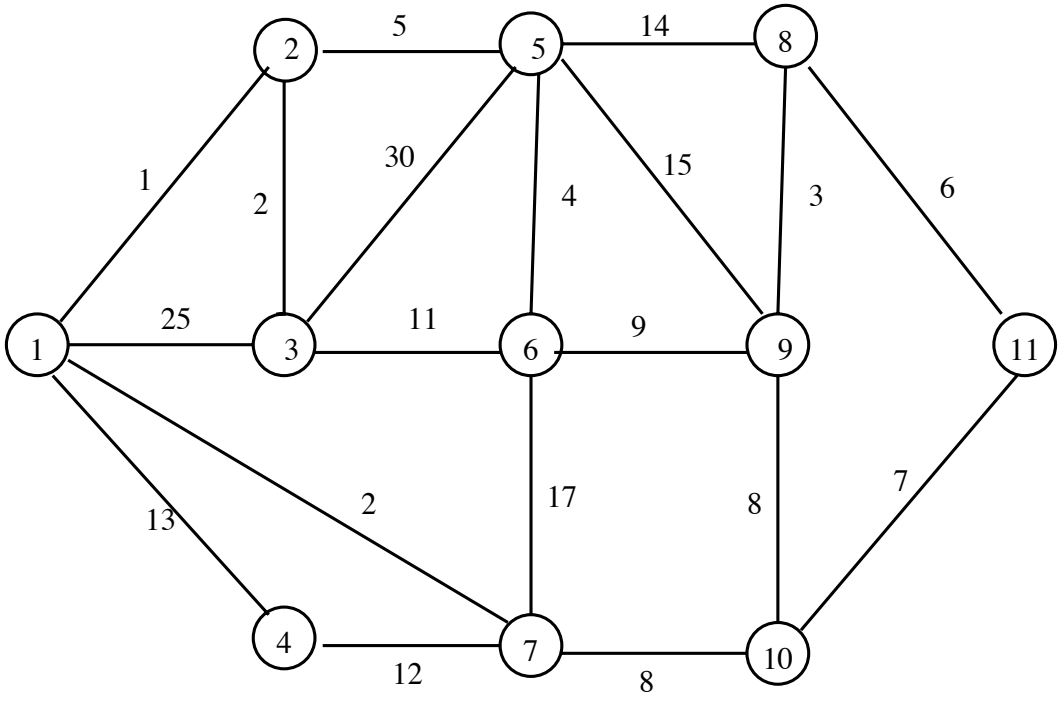

Figure 2:

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.

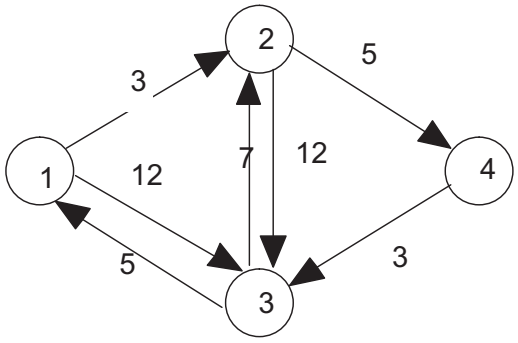

Figure 3:

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) Explain how you modify Dijkstra's shortest path algorithms on a directed graph with non-negative edge weights to count the number of shortest paths from a given origin n to a destination node d.

6. (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.

7. (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. 7. (HINT: You may want to do part b first.)

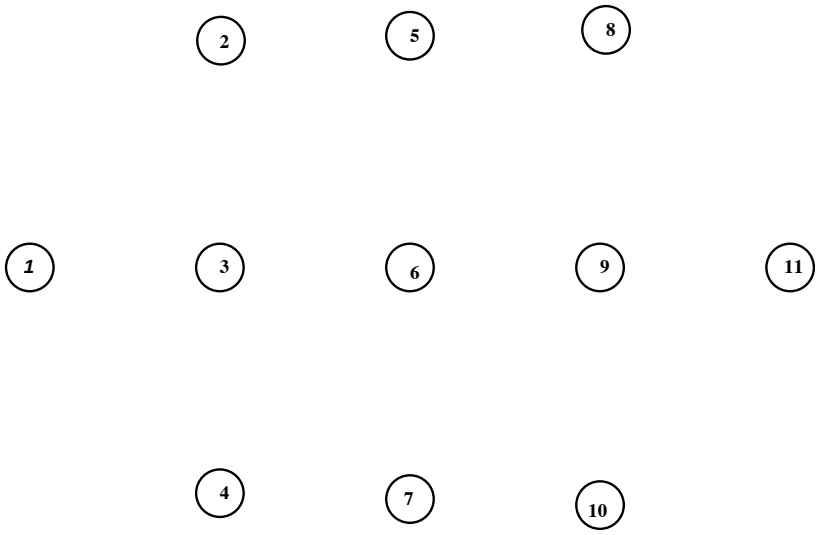

Figure 4:

(b) Represent this graph as an adjacency list.

(c) Is this graph a DAG?

8. (10 pts) An undirected bipartite graph $G(V, E)$ is one where its nodes can be partitioned into two disjoint sets $V = V_1 \cup V_2$, such that that every edge $e \in E$ is an arc $\{v_1, v_2\}$, where $v_1 \in V_1$ and $v_2 \in V_2$. Note that, in a bipartite graph, the length of every cycle must be an even number.

   Define a pseudo code for an algorithm to determine whether an undirected graph is bipartite with worst case complexity $O(m)$, where m is the number of edges in the graph.

9. (10 pts) While Euler's theorem gives a characterization of planar graphs in terms of numbers of vertices, edges and faces, it is hard to establish whether a graph is planar or not if it is difficult to count faces. There are a couple of other properties of simple, connected planar graphs that derive from Euler's theorem:

   - A simple, connected planar graph with $n \geq 3$ vertices and e edges must satisfy $e \leq 3n - 6$
   - A simple, connected planar graph with $n \geq 3$ vertices, e edges and no cycles of length 3 must satisfy $e \leq 2n - 4$

   A popular architecture for parallel computers is a hypercube. A hypercube of dimension k, denoted by Q_k, has 2^k nodes, and each node is connected to k other nodes. The nodes can be embedded into a k-dimensional boolean vector, and nodes are connected to other nodes that differ along one of its coordinates. Thus, Q_2 has nodes $(0, 0), (0, 1), (1, 0), (1, 1)$, and has 4 edges. The node (0,0) is connected to (0,1) and (1,0). Q_3 has nodes $(0, 0, 0), (0, 0, 1), (0, 1, 0), (0, 1, 1), (1, 0, 0), (1, 0, 0), (1, 1, 0), (1, 1, 1)$. Node (1,1,1) is connected to nodes (0,1,1), (1,0,1) and (1,1,0). Note that the number of edges in a hypercube of dimension k is $k * 2^{k-1}$, since each node has k edges, and we divide by 2 so as not to count the number of arcs twice. Other important facts about hypercubes is that every hypercube is a bipartite graph.

   (a) Using the above facts, verify that Q_3 is a planar graph.

   (b) Using the above facts, show that Q_4 cannot be a planar graph.