EC 504 – Fall 2020 – Homework 7

Due Friday Dec 4, 2020 in the beginning of class. Coding problems submitted in the directory `/projectnb/alg504/yourname/HW7` on your SCC account by Dec 4, 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.

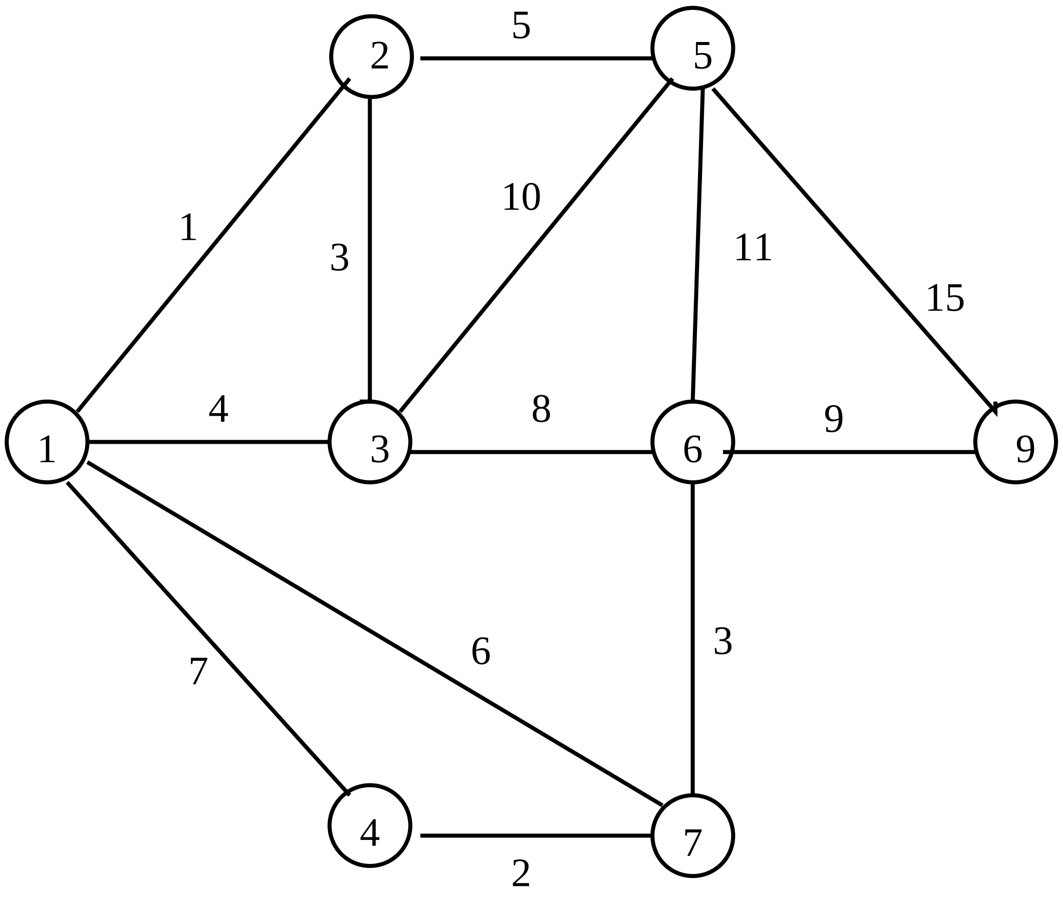

Figure 1:

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.**

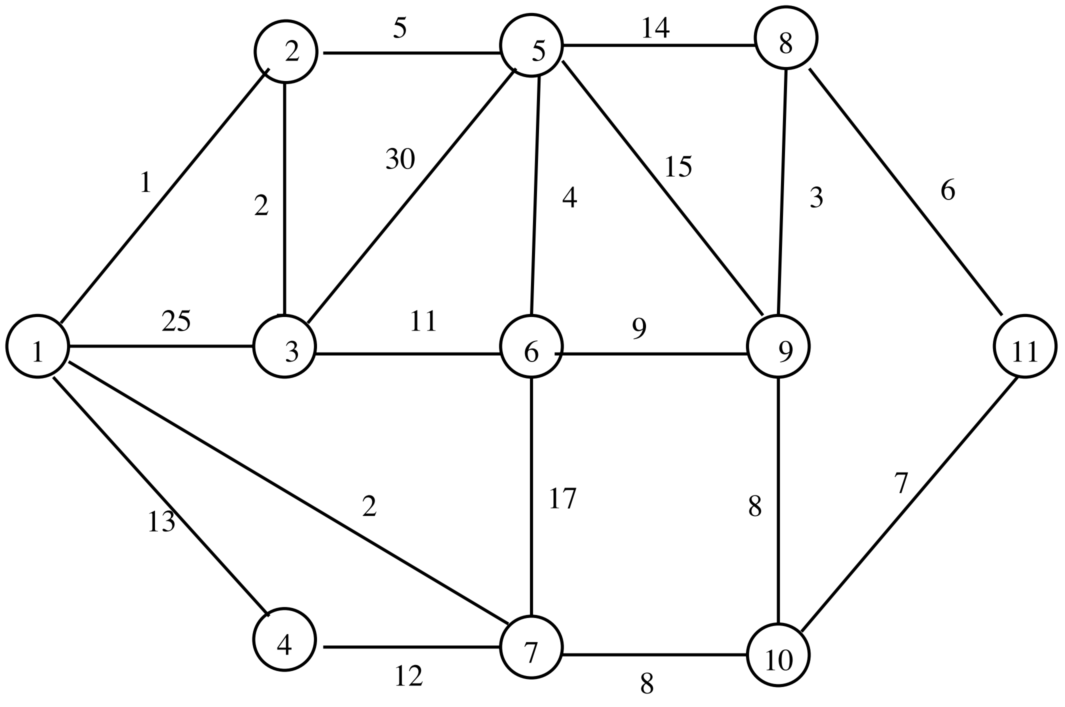

Figure 2:

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.

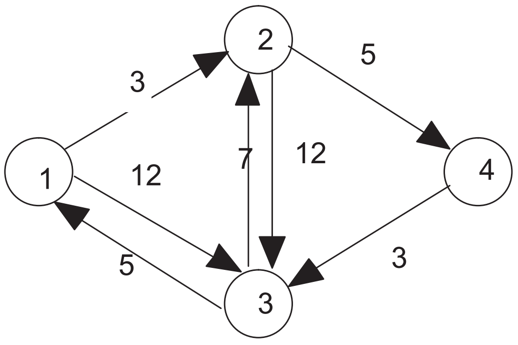

Figure 3:

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. (15 pts) In this problem, you are to implement an algorithm for the solution of shortest path problems using any variation you choose of the Bellman Ford and Dijkstra's algorithms for sparse graphs. You must implement both a variation of Bellman Ford and a variation of Dijkstra. You can choose to implement Dijkstra with either a priority heap, or without one. The objective of the project is to time these algorithms for solution on two large test graphs. The shell program `Shortest_paths.cpp` has timers already in it. The two test graphs are stored as `smallGraph.txt` and `Graph1.txt` with the progam on GitHub. The graphs are very sparse.

   Using the code provided or your own implementations, answer the questions below. Note that the nodes in the data (and in the code) will be indexed from 1 to number of nodes. The `Shortest_paths.cpp` is a fully functioning version of all code required with timers and output. You can use this or modify this as you see fit to report results for part (a) below.

   (a) The first part is to report the times required by each of the two algorithms to compute a full shortest path tree with origin node 5. Be very patient with the Bellman-Ford algorithm. It will converge some day on ACS, so you should bring along some reading material or use scripts. Print out as output the computation times and the shortest distances and the shortest paths from node 5 to nodes 573, 1021, 5783 and 9875 for the larger of the two graphs `Graph1.txt`for each of the algorithms. A hard copy of these result should be turned in with your source code on the SCC.

   (b) For extra credit, progam the $O(N^3)$ all to all distance finder. Sort the distances and plot the all distances to estimate the size *width* of graph in terms of the average distance in the distance matrx and the maximun distance. This program for the large graph will take a while so you should submit in batch queue.

   Comment: To debug and test your program you may want to make a new input file with very few node (4 or 5) that you can easily solve on scrap of paper. Trace the program with lots of output.