Homework 7 (50 pts)

1. (a) (5 points) Give an example of a weighted directed graph G with negative-weight edges, but no negative-weight cycle, such that Dijkstra's algorithm incorrectly computes the shortest-path distances from some vertex v. Trace the execution of Dijkstra's algorithm to show where it goes awry.

   (b) (5 points) Consider the following greedy strategy for finding a shortest path from vertex *start* to vertex *goal* in a given connected graph.

   1. Initialize *path* to *start*
   2. Initialize $visitedVertices$ to $\{start\}$
   3. If $start = goal$, return *path* and exit. Otherwise, continue.
   4. Find the edge ($start$, v) of minimum weight such that v is adjacent to *start* and v is not in $visitedVertices$.
   5. Add v to *path*.
   6. Add v to $visitedVertices$.
   7. Set *start* equal to v and go to step 3.

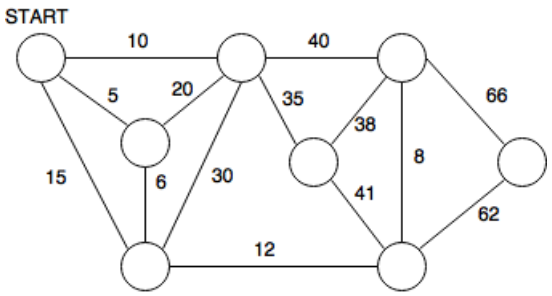
   Does this greedy strategy always find a shortest path from *start* to *goal*? Either explain intuitively why it works, or give a counter-example.

2. (10 points) There are n trading posts along a river, numbered 1 to n as you travel downstream. At any trading post i you can rent a canoe to be returned at any of the downstream trading posts j, where $j \geq i$. You are given a table $R[i, j]$ defining the cost of a canoe which is picked up at post i and dropped off at post j for $1 \leq i \leq j \leq n$. Assume that $R[i, i] = 0$ and that you can't take a canoe upriver (so perhaps $R[i, j] = \infty$ when $i > j$). However, it can happen that the cost of renting from i to j is higher than the total cost of a series of shorter rentals. In this case, you can return the first canoe at some post k between i and j and continue your journey in a second (and maybe third, fourth ...) canoe. There is no extra charge for changing canoes this way.

   Describe an efficient algorithm to determine the minimum cost of a trip by canoe from each possible departure point i to each possible arrival point j. Analyze the running time of your algorithm.

3. (10 points) Suppose you are given a diagram of a telephone network, which is a graph G whose vertices represent switching centers, and whose edges represent communications lines between two centers. The edges are marked by their bandwidth. The bandwidth of a path is the bandwidth of its lowest bandwidth edge. Give the **pseudocode** for an algorithm that, given a diagram and two switching centers a and b, will output the maximum bandwidth of a path between a and b. (Just report the maximum bandwidth; you do not have to give the actual path). Analyze the running time of your algorithm.

4. In this problem, you will show the execution of the minimum spanning tree algorithms that you studied in class on the following graph:



(a) (5 points) Trace the execution of Prim's algorithm to find the minimum spanning tree for this graph. At each step, you should show the vertex and the edge added to the tree and the resulting values of D after the relaxation operation. Use START vertex as the first vertex in your traversal.

(b) (5 points) Trace the execution of Kruskal's algorithm to find the minimum spanning tree for this graph. Give a list of edges in the order in which they are added to the MST.

5. (10 points) NASA wants to link n stations spread over the country using communication channels. Each pair of stations has a different bandwidth available, which is known a priori. NASA wants to select $n-1$ channels (the minimum possible) in such a way that all the stations are linked by the channels and the total bandwidth (defined as the sum of the individual bandwidths of the channels) is maximum. Give the **pseudocode** for an efficient algorithm for this problem and determine its worst-case time complexity. Consider the weighted graph $G = (V, E)$, where V is the set of stations and E is the set of channels between the stations. Define the weight $w(e)$ of an edge $e \in E$ as the bandwidth of the corresponding channel.

6. **[UNGRADED]** Trace the execution of the Edmonds-Karp maximum flow algorithm on the graph shown below. To break ties during BFS, visit the edges for each vertex in order. Show the augmenting path chosen in each step (and the flow of that path), as well as the final flow for each edge and the value of the maximum flow.