Design and Analysis of Algorithms
University of California, Irvine
Prof. Aparna Chandramowlishwaran

October 26, 2016
EECS 215

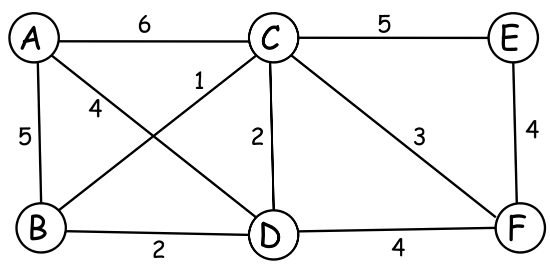
Homework 3

Due on **Monday, November 14, 2016 by 10am**

Reading: Chapters 22-26

(20 Points) Graphs

Consider the undirected graph shown in the following figure. It consists of six nodes A,B,C,D,E,F and nine edges with the shown edge costs.



1. *(5 Points)* Run Dijkstra's algorithm to find the shortest paths from node A to all other nodes. (Show the final answer and briefly describe the intermediate steps.)

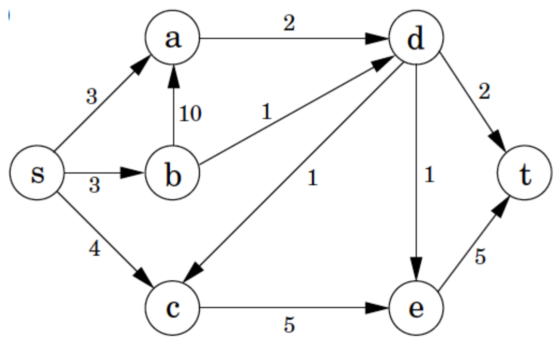
2. *(5 Points)* Run an algorithm of your choice (*e.g.*, Kruskal, Prim) and find a minimum spanning tree. (Show the final answer and briefly describe how you got there.)

3. *(5 Points)* Is the minimum spanning tree of this graph unique? Justify your answer, *i.e.*, if the answer is yes, provide a proof; if the answer is no, provide a counter-example and explain why this is the case.

4. *(5 Points)* Consider the average distance from A to all other nodes, first by following edges on the shortest path tree (a), let's call it d^{avg}_{SPT}; and then following edges on the minimum spanning tree found in (b), let's call it d^{avg}_{MST}. Which one is greater, d^{avg}_{SPT} or d^{avg}_{MST}? Does the same answer hold for any graph $G = (V, E)$ and node $A \in V$, or is it specific to this example?

(10 Points) Bellman-Ford Algorithm

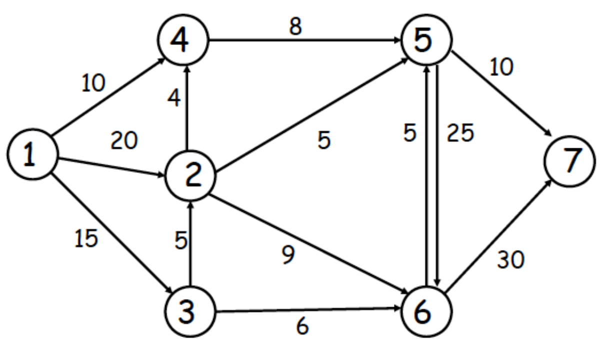
Consider the directed graph shown below. The numbers on the edges indicate lengths (or costs) of these edges.



1. *(5 Points)* Find the shortest paths from all nodes to destination t, using the Bellman-Ford algorithm. Show (some of) your intermediate steps and the final result in the following form: [next hop][distance to t] for every node.

2. *(5 Points)* After the algorithm reaches steady state, somebody cuts off edges d-t and b-d at the same time. Show the computations following those failures and the new [next hop][distance to t] for every node.

***(30 Points)* Max Flow**



1. *(5 Points)* Find the max-flow from the source (node 1) to the sink (node 7).

2. *(5 Points)* Identify the min-cut corresponding to the max-flow you found in (a).

3. *(20 Points)* Now assume that the capacity of edge 2-6 changes from 9 to 2. Find the max flow on this new graph *without* recomputing it from scratch, but starting from the solution you found in (a) and incrementally updating it.

   (a) Describe an algorithm that does that incremental update.

   (b) Argue that it indeed finds the optimal solution (max flow for the new graph).

   (c) Analyze its running time (it should be less than running Ford-Fulkerson from scratch).

   (d) Run your algorithm and report the new max flow.

(40 Points) **Prim's algorithm for MST**

The goal is to write a well-structured and well-documented program to implement Prim's MST algorithm in C/C++.

1. The input graph is to be read in from a file. The format of the file is as follows: The number of vertices, n, is the first line of the file. The vertices are numbered $0, 1, 2, ..., n-1$. Each subsequent line contains two integers, each between 0 and $n-1$, indicating the existence of an edge between these two vertices, followed by a number, indicating the weight on the edge. The graph is undirected. For example, a triangle graph in which all the weights are 0.1 would be represented as follows.

   3

   0 1 0.1

   1 2 0.1

   2 0 0.1

2. The output of Prim's algorithm is a list of edges (and their weights) that constitute a minimum spanning tree. Output the MST into a text file.

3. In your implementation of Prim's algorithm, implement the priority queue with a binary heap.

4. Your submission should contain a `Makefile` and `Readme` file with instructions to run your program. Failing these two files, your code will not be graded.

5. Evaluate your code and obtain the running times on the two test graphs – one dense and the other sparse, each with 100000 vertices given to you.

Submission

- When you've written up answers to all of the above questions, turn in your write-up by uploading it to eee.uci.edu dropbox or handing it to the instructors in person during class/office hours. LATE HOMEWORKS WILL NOT BE ACCEPTED.

- You may work in **teams of two**. Be sure to indicate your assignment partner in your submission.