Due: Friday, March 10, 2017, 11am on MarkUs

You will receive 20% of the points for any (sub)problem for which you write "I do not know how to answer this question." You will receive 10% if you leave a question blank. If instead you submit irrelevant or erroneous answers you will receive 0 points. You may receive partial credit for the work that is clearly "on the right track."

1. (20 pts) Short answer questions. For each question answer TRUE/FALSE. If you answer TRUE provide a brief justification consisting of at most 3 short sentences (for majority of these questions, one short sentence suffices). If you answer FALSE provide a small counter-example. Long and complicated justifications as well as unnecessarily large and complicated counter-examples will lose marks. Guesses without justification receive 0 marks.

   Note: for some of the false statements below it might be hard to find a counter-example. You are encouraged to give it your best shot, but if you notice that you are spending disproportionate amount of time, you are encouraged to consult the web. If you find a counter-example on the web you must *understand it, internalize it, and write it up from memory.* In addition, you must cite the resource that you used (this won't cost you any mark deductions), and you are still responsible for correctness of the example (i.e., you can't blame the source if it ultimately turns out incorrect - there is plenty of wrong information on the web).

   (a) An undirected graph with n vertices and at most $n - k$ edges has at least k connected components.

   (b) The shortest-paths tree computed by Dijkstra is necessarily an MST.

   (c) Suppose that we have computed an MST. If the weight of each edge in the graph is increased by 1, the computed spanning tree remains minimum with respect to the new weights.

   (d) In a weighted directed graph with positive weights, Dijkstra might call the update() procedure (aka Relax() procedure, see CLRS 24.3) on the same edge more than once.

   (e) Maximum flow in a network with integral capacities is necessarily integral.

   (f) We are given a weighted graph and a shortest path from s to t. If all edge weights in the graph are multiplied by a positive constant, the given path remains shortest from s to t with respect to the new weights.

   (g) Suppose we run DFS on a directed graph $G = (V, E)$ and we find a vertex with discovery time 1 and finishing time $2|V|$. Then the entire graph must be strongly connected.

   (h) Ford-Fulkerson method runs in polynomial time assuming that capacities are positive integers.

   (i) Ford-Fulkerson method terminates on all input flow networks with capacities that are positive real numbers.

   (j) Undirected graph $G = (V, E)$ is called k-regular if degree of every vertex is exactly k. Girth of the graph is the length of a shortest cycle in G. For example, the simple cycle with 5 vertices is a 2-regular graph of girth 5. We claim that there is no 3-regular graph of girth 5 on 10 vertices.

2. (20 pts) Design an efficient algorithm for the following problem:

   Input: Undirected graph $G = (V, E)$ in adjacency lists representation with unit edge costs. Vertices $s, t \in V$.
   Output: The number of distinct shortest paths from s to t.

   (a) Briefly describe your algorithm in plain English.

   (b) Describe your algorithm in pseudocode.

   (c) Formally prove correctness of your algorithm.

   (d) State and *justify* the running time of your algorithm.

3. (20 pts) You have a server that can be attached to two electrical outlets simultaneously. The server runs uninterrupted as long as it is attached to at least one electrical outlet. The server is located on a rolling cart that can be moved freely within the rectangular room (the room has no obstacles). Looking at the room from above and seeing its plan, you know the coordinates of n electrical outlets given by (x_i, y_i). Currently the server is attached via a single power cable of length ℓ to the electrical outlet s. You would like to move the server without any interruptions to electrical outlet t. For that purpose you can purchase an additional power cable of length L and move the server from one electrical outlet to another until it reaches t. You would like to find out what is the minimum length L of additional power cable that you need to purchase to accomplish this task. Design an $O(n^2 \log n)$-time algorithm to compute the minimum value of L. Note this problem is assumed to be entirely 2-dimensional – length in this problem refers to the regular Euclidean 2D distance.

   Input: $\{(x_i, y_i)\}_{i=1}^{n}$ – positions of electrical outlets; ℓ – length of the given power cable; $s, t \in [n]$ – starting and terminal electrical outlets
   Output: minimum value of L such that we can move the server from s to t uninterrupted.

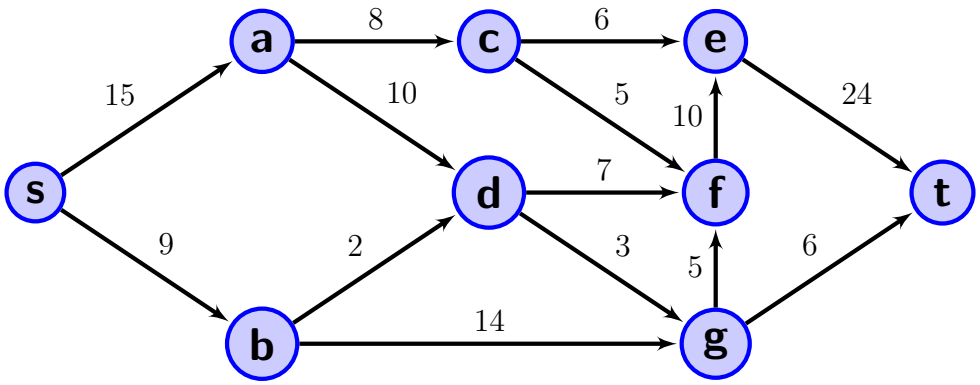
   (a) Briefly describe your algorithm in plain English.

   (b) Describe your algorithm in pseudocode.

   (c) Provide a concise argument of correctness of your algorithm. You may use results proven in class/textbook, but make sure to cite them accurately.

   (d) Justify that the runtime is $O(n^2 \log n)$.

4. (20 pts) Consider the following network graph.



(a) Compute maximum flow f and minimum cut (S, T).

(b) Draw the residual graph G_f - don't forget to state the capacities. Indicate the minimum cut (S, T) in the residual graph by circling S and T.

(c) An edge is called *constricting* if increasing its capacity leads to an increase in the value of maximum flow. List all constricting edges in the above network.

(d) Find a small (at most 4 nodes) example of a network graph that has no constricting edges.

(e) Describe in plain English an efficient algorithm to find all constricting edges. Argue correctness by using results from lectures/textbook. State the running time of your algorithm.

5. (20 pts) Consider a flow network $G = (V, E), s, t, c$ with integral capacities together with an additional edge-price function $p : E \to \mathbb{N}$. The value $p(e)$ denotes the price to increase the capacity of edge e by one unit. Suppose we have already computed maximum flow f in the network. Now, we would like to increase this maximum flow f by one unit by increasing capacities of some edges. The goal is to do this with the least possible cost. Design an efficient algorithm to compute which edge capacities to increase.

(a) Briefly describe your algorithm in plain English.

(b) Describe your algorithm in pseudocode.

(c) Provide a concise argument of correctness of your algorithm. You may use results proven in class/textbook, but make sure to cite them accurately.

(d) State and justify the runtime of your algorithm.