CS 4349.400 Homework 6

Due Wednesday October 31st, in class

Please answer each of the following questions.

1. Inspired by Homework 5, you decide to organize a Snakes and Ladders competition with n participants. In this competition, each game of Snakes and Ladders involves three players. After the game is finished, they are ranked first, second, and third. Each player may be involved in any (non-negative) number of games, and the number need not be equal among all players.

   At the end of the competition, m games have been played. You realize that you forgot to implement a proper rating system, and therefore decide to produce the overall ranking of all n players as you see fit. However, to avoid being too suspicious, if player A ranked better than player B in any game, then A must rank better than B in the overall ranking.

   You are given the list of players and their ranking in each of the m games. Our goal is to design and analyze an algorithm that produces an overall ranking/ordering of the n players that is consistent with the individual game rankings, or correctly reports that no such ranking exists.

   (a) What vertices should we use for our graph? *[Hint: Remember, our goal is to order the players in some way. You should start thinking about what graph algorithm we'll ultimately end up calling.]*

   (b) What edges should we use in our graph? Are they directed or undirected edges? *[Hint: Remember, player A must rank better than player B overall if A ranked better than B in any one game.]*

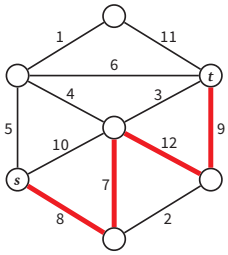
   (c) What standard algorithm should we call on our graph?

   (d) How do we translate the output of the algorithm into the final rankings of our players? (It's up to you exactly how you want to output the ranks, but it should be unambiguous who ranks better from any pair of players.)

   (e) What is the running time of the graph algorithm ***in terms of V and E***, the number of vertices and edges.

   (f) What is the running time of the whole algorithm in terms of ***the input parameters*** n, the number of players, and m, the number of games?

2. Consider a path between two vertices s and t in a connected undirected graph G. The ***width*** of the path is the minimum weight of any edge in the path. The ***bottleneck distance*** between s and t is the width of the widest path from s to t.



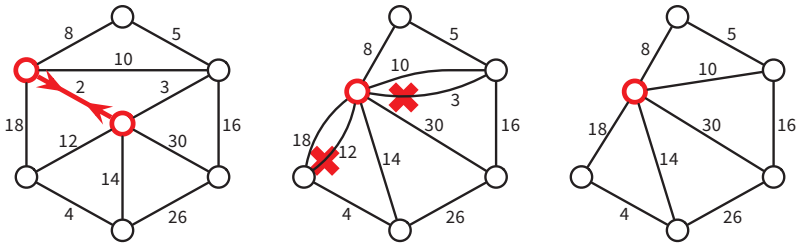
The bottleneck distance between s and t is 7.

(a) Prove that the *maximum* spanning tree of G contains the widest paths between *every* pair of vertices. You may assume all edge weights are distinct. (The maximum spanning tree is the spanning tree T maximizing $\sum_{e \in T} w(e)$.) *[Hint: Suppose the maximum spanning tree does not contain the widest path between some pair of vertices s and t. Perform an exchange argument similar to the one we used when discussing safe edges to show the spanning tree isn't maximum after all.]*

(b) Suppose B is the bottleneck distance between s and t. (You can write "I don't know" for either part i. or ii. individually if you need to.)

i. Prove that deleting any edge with weight less than B does not change the bottleneck distance between s and t. *[Hint: What, if anything, happens to the widest path from s to t? Does the deletion create any wider paths between s and t?]*

ii. To ***contract*** an edge uv, we insert a new node in G, redirect any edge incident to u or v (except uv) to this new node, and then delete u and v. After contraction, there may be multiple parallel edges between the new node and other nodes in the graph; we remove all but the *heaviest* edge between any two nodes.



Contracting an edge and removing redundant parallel edges.

Prove that contracting any edge with weight *greater* than B does not change the bottleneck distance between s and t. *[Hint: What, if anything, happens to the widest path from s to t? Does the contraction create any wider paths between s and t?]*

(c) Describe an algorithm to solve the following problem in $O(E)$ time: Given a connected undirected graph G, two vertices s and t, and a weight W, is the bottleneck distance between s and t at least W? *[Hint: 2(b)i. Modify the graph is some way and then run a basic algorithm you saw in class.]*

(d) Describe an algorithm to compute a widest path between s and t in $O(E)$ time. You may assume you have access to a procedure ContractBigger(G, W) that contracts *all* edges of weight greater than input parameter W in $O(E)$ time. *[Hint: Start by finding a median-weight edge in G in $O(E)$ time. You probably want to refer to parts (b) and (c).]*

3. We say that a graph $G = (V, E)$ is *dense* if $E = \Theta(V^2)$. Describe a modification of the Prim-Jarnìk minimum spanning tree algorithm that runs in $O(V^2)$ time (independent of E) when the input graph is dense, using only simple data structures (and in particular, *without* using a Fibonacci heap). *[Hint: Describe a very simple data structure for implementing the priority queue. How long can you afford an ExtractMin to take knowing you have $\Theta(V^2)$ time to run the whole algorithm?]*