CSCI 3100, Fall 2018

Homework 4

Problem 1 [25 points]

Consider the following problem, discussed in class: Given a connected graph *G*, with distinct edge weights, let *n* be the number of vertices in *G* and *m* be the number of edges. A particular edge *e = (v, w)* of *G* is specified. Determine if *e* is contained in a minimum spanning tree of *G.*

The following algorithm to this problem was presented in class:

1. Construct *G’* from *G* by deleting edges with weight greater than the weight of *e*. Delete *e* as well.
2. If there is a path from *v* to *w* in *G’*, then *e* is not included in any minimum spanning tree of *G*. Otherwise, *e* is included in a minimum spanning tree of *G*.

<u>Prove that</u> this algorithm is correct (prove that it produces the correct answer). Hint: you may want to use the "cycle property" in your proof.

Problem 2 [25 points]

Suppose you are given a connected graph *G*, with edge costs that are all distinct. <u>Prove that</u> *G* has a unique minimum spanning tree.

Problem 3 [25 points]

One of the basic motivations behind the Minimum Spanning Tree Problem is the goal of designing a spanning network for a set of nodes with the minimum total cost. Here we explore another type of objective: designing a spanning network for which the most expensive edge is as cheap as possible.

Specifically, let *G=(V, E)* be a connected graph with *n* vertices, *m* edges, and positive edge costs that you may assume are all distinct. Let *T=(V, E')* be a spanning three of *G;* we define the *bottleneck edge* of *T* to be the edge of *T* with the greatest cost.

A spanning tree *T* of *G is a* minimum-bottleneck spanning tree if there is no spanning tree *T'* of *G* with a cheaper bottleneck edge.

(a) <u>Provide an example</u> *G* and a minimum-bottleneck spanning tree of *T of G*, where *T* is NOT a minimum spanning tree of *G*.
(b) <u>Prove that</u> a minimum spanning tree of *G* is a minimum bottleneck tree of *G*.

Problem 4 [25 points]

A group of network designers at a communications company find themselves facing the following problem. They have a connected graph *G = (V, E)*, in which vertices represent sites that want to communicate. Each edge *e* is a communication link, with a given available bandwidth b_e.

For each pair of nodes *u, v* in *V*, they want to select a single *u-v* path *P* on which this pair will communicate. The *bottleneck rate b(P)* of a path *P* is the minimum bandwidth of any edge it contains; that is, $b(P) = min$ {b_e, for all *e* in *P*}. The *best achievable bottleneck rate* for the pair *u, v* in *G* is simply the maximum bottleneck, over all *u-v* paths *P* in *G.*

It's getting to be very complicated to keep track of a path for each pair of vertices, and one of the network designers makes a bold suggestion: Maybe one can find a spanning tree *T* of *G* so that for every pair of nodes *u, v*, the unique *u-v* path in the tree attains the best achievable bottleneck rate for *u, v* in *G. (In other words, even if you could choose any* u-v path in the whole graph, you couldn't do better than the *u-v* path in *T*).

We can find such tree *T*, by computing the minimum spanning tree of *G* with edge weight equal to the negative of its bandwidth. <u>Prove that</u> the bottleneck rate of any *u-v* path in *T* is equal to the best achievable bottleneck rate for the pair *u, v* in *G*.