Note: Your TA probably will not cover all the problems. This is totally fine, the discussion worksheets are not designed to be finished in an hour. They are deliberately made long so they can serve as a resource you can use to practice, reinforce, and build upon concepts discussed in lecture, readings, and the homework.

1 The Discussion is Yours

Since this discussion is right after a midterm, there'll be less content covered than usual (just MSTs this time) and so our time is more flexible. Other than MSTs, is there anything else you'd like to review or clarify today? Do you have any requests regarding how future discussions should be run? Please let your TA know if you have any questions or concerns!

2 MST Basics

For each of the following statements, either prove it or give a counterexample. Always assume the graph $G = (V, E)$ is undirected and connected. Do not assume the edge weights are distinct unless specifically stated.

(a) Let e be any edge of minimum weight in G. Then e must be part of some MST.

(b) If e is part of some MST of G, then it must be a lightest edge across some cut of G.

(c) If G has a cycle with a unique lightest edge e, then e must be part of every MST.

(d) For any $r > 0$, define an r-path to be a path whose edges all have weight less than r. If G contains an r-path from s to t, then every MST of G must also contain an r-path from s to t.

3 Minimum Spanning Trees

(a) Given an undirected graph $G = (V, E)$ and an edge set $E' \subset E$, briefly describe how to update Kruskal's algorithm to find the minimum spanning tree that includes all edges from E'.

(b) Suppose we want to find the minimum cost set of edges that suffices to connect a given weighted graph $G = (V, E)$; if the weights are non-negative then we know that the optimum will be a MST. What about the case when the weights are allowed to be negative? Does it have to be a tree if the weights are allowed to be negative? If not, how would you find this minimum-cost connected subgraph?

(c) Describe an algorithm to find a maximum spanning tree of a given graph.

4 Updating a MST

You are given a graph $G = (V, E)$ with positive edge weights, and a minimum spanning tree $T = (V, E')$ with respect to these weights; you may assume G and T are given as adjacency lists. Now suppose the weight of a particular edge $e \in E$ is modified from $w(e)$ to a new value $\hat{w}(e)$. You wish to quickly update the minimum spanning tree T to reflect this change, without recomputing the entire tree from scratch.

There are four cases. **In each, give a description of an algorithm for updating T and a runtime analysis for the algorithm.** Note that for some of the cases these may be quite brief. For simplicity, you may assume that no two edges have the same weight (this applies to both w and $\hat{w}$).

(a) $e \in E'$ and $\hat{w}(e) < w(e)$

(b) $e \notin E'$ and $\hat{w}(e) < w(e)$

(c) $e \in E'$ and $\hat{w}(e) > w(e)$

(d) $e \notin E'$ and $\hat{w}(e) > w(e)$

5 A Divide and Conquer Algorithm for MST

Is the following algorithm correct? If so, prove it. Otherwise, give a counterexample and explain why it doesn't work.

procedure FindMST(G: graph on n vertices)
  if $n = 1$ **then**
    return empty set
  $T_1 \leftarrow$ FindMST(G_1: subgraph of G induced on vertices $\{1, \ldots, \frac{n}{2}\}$)
  $T_2 \leftarrow$ FindMST(G_2: subgraph of G induced on vertices $\{\frac{n}{2}+1, \ldots, n\}$)
  $e \leftarrow$ cheapest edge across the cut $\{1, \ldots, \frac{n}{2}\}$ and $\{\frac{n}{2}+1, \ldots, n\}$.
  return $T_1 \cup T_2 \cup \{e\}$