Minimum Spanning Trees (short answer)

(a) Given an undirected graph $G = (V, E)$ and a 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) Assume you are given a graph $G = (V, E)$ with positive and negative edge weights and an algorithm that can return a minimum spanning tree when given a graph with only positive edges. Describe a way to transform $G$ into a new graph $G'$ containing only positive edge weights so that the minimum spanning tree of $G$ can be easily found from the minimum spanning tree of $G'$.

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

Picking a Favorite MST

Consider an undirected, weighted graph for which multiple MSTs are possible (we know this means the edge weights cannot be unique). You have a favorite MST, $F$. Are you guaranteed that $F$ is a possible output of Kruskal’s algorithm on this graph? How about Prim’s? In other words, is it always possible to “force” the MST algorithms to output $F$? Justify your answer.

MST Variant

Give an undirected graph $G = (V, E \cup S)$ with edge weight $c(e)$. Note that $S$ is disjoint with $E$. Design an algorithm to find a minimum one among all spanning trees having at most one edge from $S$ and others from $E$.

Input: A graph $G = (V, E)$, set of potential superhighways $S$, and a cost function $c(e)$ defined for every $e \in E \cup S$.

Output: A tree $T = (V, E')$ such that $T$ is connected (there is a path in $T$ between any two vertices in $V$), $E' \subseteq E \cup S$, $\sum_{e \in E'} c(e)$ is minimized, and $|E' \cap S| \leq 1$.

Service scheduling

A server has $n$ customers waiting to be served. Customer $i$ requires $t_i$ minutes to be served. If, for example, the customers were served in the order $t_1, t_2, t_3, \ldots$, then the $i$th customer would wait for $t_1 + t_2 + \cdots + t_i$ minutes.

We want to minimize the total waiting time

$$T = \sum_{i=1}^{n} \text{(time spent waiting by customer } i)$$

Given the list of $t_i$, give an efficient algorithm for computing the optimal order in which to process the customers.