Time Spent: 2 hours

Collaborators and Resources:

Problem 2

To start off with we're given a problem in which we want to find the minimum spanning worker tree. This is essentially a tree T = (V, E' $\subseteq$ E) (where vertices are buildings and edges are tunnels) such that we minimize $max_{e \in E'} w(e)$ where w is a function from the set of edges E to the set of non-negative integers $\mathbb{Z}$. In the context of the problem w denotes how many workers w(e) are at each edge e, but I chose to approach this problem exclusively in the context of the weight w(e) of each edge e as an alternative. The key difference between a MSWT and a MST is that we want to minimize the maximum weight over all edges instead of minimizing the sum of all edge weights. In other words, we want to be able to fix up all tunnels connecting buildings with the least total number of workers at any time. In our MSWT we need to be able to say that the maximum edge weight across T is the absolute minimum ...

(a) Prove that an MST is also an MWT.

Claim 1. A MST is also a MWT.

Proof. All minimum spanning trees as defined have the least number of edges possible to keep all vertices connected.

This is because we're given a weighted connected graph G = (V,E), W:E $\to \mathbb{R}_{>0}$. By definition, a MST minimizes the sum of edge weights. Algorithms like Prim's (and Kruskal's, by checking there aren't cycles) always select the cheapest edge connecting each additional vertex to (an indeterminate vertex in) the already existing tree and in doing so find the MST. That is, for each additional vertex added to the initial tree S = [x], exactly one additional edge is added. If we could remove an edge from the MST and it's still spanning, we already did because otherwise the sum of edge weights isn't minimized. So, if we remove any edge from an MST then the set V of vertices becomes disconnected (disjoint).

These things being said, let any minimum spanning tree MST = (V, E' $\subseteq$ E) be given. Suppose it is not also a minimum spanning worker tree.

Then, $\text{sum}_{e \in E'} w(e)$ is minimized and there exists a different minimum spanning worker tree MWT = (V, E" $\subseteq$ E) such that $\max_{e \in E''} w(e) < \max_{e \in E'} w(e)$. That is, the maximum edge weight in MWT is less than the max edge weight in MST $\implies$ there exists a weighted edge (call it e_o) $\in$ MST which is not in MWT because it weighs too much.

Removing e_o from MST disconnects MST into two disjoint intra-connected minimum spanning subtrees T_1 = (S, $E_1 \subseteq$ E) and T_2 = (V\S, $E_2 \subseteq$ E).

MWT excludes e_o because it weighs too much, but it still has to span every vertex in V. So it includes an alternative route e_A which connects S to V\S such that w(e_A) < w(e_o).

So we've established that there's an alternative edge e_A from S to V\S with less weight than that of the edge which MST already found connecting T_1 = (S, $E_1 \subseteq$ E) to T_2 = (V\S, $E_2 \subseteq$ E). What this means is the MST-finding algorithm would have already exchanged e_o with e_A because e_A is clearly cheaper and V would still be connected. Therefore, $e_o = e_A \implies$ all such highest-weight edges in question have to be in both MST and MWT.

A minimum spanning worker tree can't contain an alternative edge with less weight because this alternative edge would have already been included by any minimum spanning tree finding algorithm. So all edges in a MST do not exceed the maximum edge weight in a MWT. Then by definition, a MST is a MWT. □

(b) Part (a) tells us that we can find an MWT in O($m$log$n$)-time, but President Poskanzer only needs to know if an MWT exists with a maximum number of workers required less than k.

Give a O($n + m$)-time algorithm that, given a graph G = (V, E), weights $w : E \to \mathbb{Z}_{\geq 0}$, and an integer k, reports whether there is an MWT whose maximum number of workers is less than or equal to k.

This algorithm which will be expanded upon in the next problem is essentially a modification of the pre-existing BFS algorithm. We talked about the Breadth-first search algorithm having running time O({V} + {E}) a few weeks ago (3.11 K-T, true as long as the graph will be given by the adjacency list representation) and as a result we will be able to find whether there is an MWT with maximum worker number (edge weight) k in O($n + m$) time

(where n represents vertices and m represents edges). The whole point of this algorithm which will be iterated below is to try to find out if we can't find a MST (a MWT) without including some edges of weight greater than k ...

1. Given graph G = (V,E) and vertex s ∈ V:
2. Run BFS (without including edges of weight greater than k) to get D of distances from the given starting vertex s.
   # This takes O(n+m) time
3. Iterate over all the nodes in the resulting tree and check if there are $|V|$ nodes in total.
4. If so, return True

Claim 2. The algorithm is indeed $O(n + m)$ and reports whether there is a MWT whose maximum number of workers is less than or equal to k.

Proof. The reason that the algorithm is the sum of edges and vertices is that the breadth-first search algorithm goes through the graph in layers, one at a time. We've already established this fact, and the resulting algorithm is just an additional step which involves making sure that the weight of the edges (the number of workers) does not exceed k for any one edge before allowing the breadth-first search algorithm to determine the shortest distance to each level at a time. So as a result, all additional operations are in linear time so we know that the total runtime is $O(n + m)$ and additionally that it is going to determine whether or not the entire graph can be connected without the components which require more than k workers. □