Group Assignment 4

Philip Warton, Brandon Lam, Jacob Cheney

November 24, 2020

1 Algorithm Description

Let MST_i denote the i-th best minimum spanning tree. The algorithm we use to solve this problem, is based on an existing algorithm for finding a minimum spanning tree. Using Kruskal's Algorithm, we can compute our first minimum spanning tree of course. For the second and third, these require more attention. Let E be our list of edges, for each $e \in MST_1$, we take the set of edges $E \setminus e$ and compute a minimum spanning tree based on this list of edges. We add its total weight to an array of weights, and take the two smallest values from this array. We claim that this gives the weights of each MST_1, MST_2, MST_3.

2 Running Time Analysis

Since our input comes in the form of an adjacency matrix, to convert this into an adjacency list, we must loop through the upper or lower triangle of our adjacency matrix, which we can do in $O(n^2)$ time at best. Then we sort our list of edges, where $|E| = \frac{n(n-1)}{2} - n$, since that is how many entries lie in our upper or lower triangle. The sorting occurs in of course $O(E \log E)$ time. Then, if $n > 3$, in our worst case scenario we must compute the minimum spanning tree from some edge list $|V| - 1$ times. This is because for each edge $e \in MST_1$ we create a new minimum spanning tree based on $E \setminus e$. Since our MST_1 contains $v - 1$ edges, this of course occurs $v - 1$ times. Since Kruskal's Algorithm takes $O(E \log V)$ excluding the sorting, we say that computing all $|V| - 1$ trees takes $O(V) \cdot O(E \log V) = O(VE \log V)$. Then since we have the sorting in addition to this, we get a total runtime of

$$O(E \log E) + O(VE \log V) = O(VE \log V)$$

3 Proof of Correctness

We assume without proof that for any undirected weighted graph G such that it has a minimum spanning tree, Kruskal's Algorithm produces a valid minimum spanning tree. Trivially, by using the algorithm our first minimum spanning tree will of course be correct. Then we claim that for the second and third MST's, changing one edge from MST_1 will produce the correct results for MST_2, MST_3. Since we compute all such trees, showing this will be sufficient to show that the algorithm is correct.

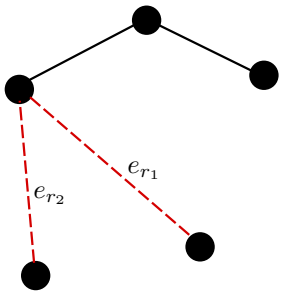
Proof. Suppose that changing two edges produced a second or third minimum spanning tree. Call our edges to remove e_{r_1}, e_{r_2}. Then we say that our tree is split into three disconnected compenents, and that we have three distinct root nodes on our disjoint set structure. We call these components C_0, C_1, C_2 such that e_{r_1} connected C_0, C_1, and e_{r_2} connected C_1, C_2. Then we of course must add in two more edges e_{a_1}, e_{a_2} such that all three components become connected, achieving a spanning tree once again. Our net change in weight will be

$$\Delta = e_{a_1} + e_{a_2} - e_{r_1} - e_{r_2}$$

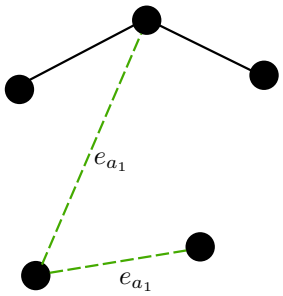
It should be clear that both $e_{a_1} \geqslant e_{r_1}$ and $e_{a_2} \geqslant e_{r_2}$, otherwise our original MST_1 is not valid since a smaller spanning tree would exist (simply replace one of these edges). Then we could produce two smaller spanning trees where only one edge is changed, by using only one of these r/a pairs, Figure 1. For these spanning trees the change will be

$$e_{a_1} - e_{r_1} \quad \text{or} \quad e_{a_2} - e_{r_2}$$

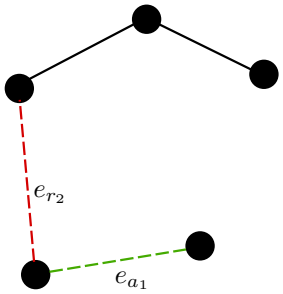
Both of these quantities are less than Δ. Thus, said tree cannot be smaller than the second or third minimum spanning trees. If they are equal, then the value will be found by exhausting all trees with only one edge changed. It follows that this argument will extend to changing $3, 4, 5, \cdots$ edges as well, since we can simply take one pair of changed edges that connect the components that become disconnected. □

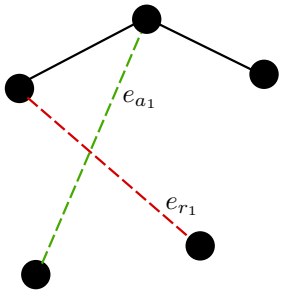


Two Edges Removed



Two New Edges Introduced





Two Smaller Possible MST's

Figure 1: Diagram of Other Possible Trees