Time Spent: 1 hour

Collaborators and Resources:

Problem 1

For this problem I wanted to consider several things. Firstly, given an undirected graph of vertices and edges where the number of vertices $|V|$ is odd and the edges are partitioned into two disjoint sets A (above ground) and B (below ground), we're asked to find a spanning tree of G with exactly $\frac{(|V|-1)}{2}$ edges from each set A and B. Now, disjointness implies that the sets do not overlap. That being said, we don't know what edges are in A and B and so finding a spanning tree of G by analyzing whether the edge is from A or B at each step is going to be fairly difficult. However, what we can do is implement some of the algorithms that we have discussed in order to find a solution to the problem that won't take too much runtime given that there aren't too many vertices ...

1. Given an undirected graph G = (V, E = A ∪ B) where V is odd
2. For all edges in the graph, sort them in all possible distinct ways
    (a) For each sorting arrangement
        i. For each edge e in order, add e to T_k if it connects two disjoint components of T_k
        ii. Add T_k to a list of possible spanning trees
3. For all trees, check if exactly $\frac{(|V|-1)}{2}$ are from set A
4. If this is the case, return such a tree
5. If it is not, return none exists

Claim 1. For all valid graphs, the algorithm will find if a spanning tree G with exactly $\frac{(|V|-1)}{2}$ edges from set A and $\frac{(|V|-1)}{2}$ edges from set B exists in $O(|E|! \cdot log(|E|))$ time.

Proof. It's clear that the algorithm is a modified version of Kruskal's algorithm which we already know has $O(|E| \cdot log(|E|))$ time because it has to sort through all the edges by increasing order of weight before adding the first edge connecting two disjoint components (a greedy algorithm). Thus my algorithm is just a modification which, because we aren't trying to find a minimum spanning tree based on weights. The algorithm needs to sort the edges in all possible ways where each edge has the same weight. Then, it runs Kruskal's algorithm on all possible orderings of edges which takes more time. However, since we know Kruskal's algorithm returns the MST of any graph G, we're going to get a minimum spanning tree in each case.

We know that any MST is already going to have $|V|$ - 1 edges total because we start with one vertex (which requires $|V|$ - 1 = 0 edges) and each additional vertex v in the graph is just going to require one additional edge (between v and the rest of the minimum spanning tree) to be added to maintain the graph's connectivity. So all we need to do is check whether the number of edges from set A within a given MST is exactly $\frac{(|V|-1)}{2}$. If more edges are found from A then we're going to find less edges from B, for example, so checking whether exactly $\frac{(|V|-1)}{2}$ edges within any tree are from set A is sufficient to find if the number of edges from each set A and B are equal. Since we found all MSTs we know whether or not such a spanning tree exists and will return it if true. □