Jordan Allard
CSCI 264-01
Homework 6

HW6 Problem 1 Writeup

a) Verbal Description

This algorithm determines the minimum cost of an F-containing spanning tree of a graph G. The program uses an implementation of Kruskal's algorithm that starts with all the edges of F already added to the final set of edges. This version of Kruskal's algorithm utilizes Union-Find implementation.

b) Pseudocode

```
Init(V):
      For every element v in V:
            boss[v] = v
            size[v] = 1
            set[v] = [v]

Union(u, v):
      If size[boss[u]] > size[boss[v]]:
            temp = u
            u = v
            v = temp
      Append set[boss[v]] to set[boss[u]]
      size[boss[u]] += size[boss[v]]
      For every z in set[boss[v]]:
            boss[z] = boss[u]

Kruskal(V, E, F, w):
      treeWeight = 0
      Init(V)
      For every edge (u, v) in F:
            Union(u, v)
            treeWeight += w[u, v]
            Remove (u, v) from E
      Sort the edges in E in increasing order of weight
      For every edge (u, v) in E:
            If boss[u] != boss[v]:
                  treeWeight += w[u, v]
                  Union(u, v)
      Return treeWeight
```

c) Proof of Correctness

We know that Kruskal's algorithm will always find the minimum spanning tree if there is one. The algorithm always takes the edge with the smallest weight that doesn't form a cycle, so no shorter path can exist to that vertex than the one that was already taken. The algorithm checks the shortest edges first and will check every edge until every vertex is visited, so it is guaranteed to find a spanning tree if one exists. By starting with the edges of F already taken in the tree, the algorithm can determine if a spanning tree can be formed without creating a cycle while containing F. Since all edges of F must be included in an F-contained spanning tree, including the weights of F will still find the minimum possible spanning tree.

d) Running Time Estimate

O(m log m)

e) Running Time Estimate Reasoning

The running time of this algorithm can be broken down as follows:

Initialization --> O(n) + O(m)
Getting input --> O(m)
Kruskal's --> O(m log n)

The base of this algorithm is Kruskal's algorithm, which is a for loop that runs up to m times which contains a chance to run Union. The technical running time of this loop is O(n^2), but since unions can only occur until the union reaches the number of vertices in the set, the for loop has an actual running time of only O(n log n). Thus, the overall runing time of Kruskal's algorithm is O(n log n + m log n) from the sorting, which is O(m log n). This variation does not include any changes that increase the running time, since whether the unions occur in the initialization with edges in F or whether they occur later with the edges in E, there can still only be a total of n - 1 unions and the lemma still holds.