# CS6150 - Homework/Assignment-4

Arnab Das(u1014840)

November 3, 2016

## 1: Minimum Spanning Tree

**(a)** Let G be weighted directed graph with all edge weights being distinct. **Prove:** G has an unique spanning tree:
**Proof:** Consider G has two spanning trees, $T_1$ and $T_2$. Let $e_1$ be the edge with the minimum weight in the graph that is contained in $T_1$ and not in $T_2$. Then $T_2 \cup e_1$ results in a cycle, say C. Then one of the other edges of C, say $e_2$ is not in $T_1$, else $T_1$ would have had a cycle. Then, if we replace $e_2$ by $e_1$ in $T_2$, that would result in a spanning tree with a smaller weight than $T_2$. Then $T_2$ is not a spanning tree. Hence $T_1$ is a unique spanning tree. (Proved)

**(b)** Even if the weights of the graph are not distinct, it is still possible to get a unique spanning tree. We describe below an algorithm , that is a slight modification of the Kruskal algorithm. Given a weighted graph $G = (V, E, W)$, where V denotes the set of vertices, E denotes the set of edges and W denotes the edge weights. Consider initially a setting of all the nodes without any of the edges, each is a separate tree. Then we gradually build up the minimum spanning tree(mst) by union of these disjoint trees. We always start with the edge of minimal weight at every step, and if the end points of that edge lies in two different trees, then that edge acts as the union of these two trees resulting in a bigger tree. If , the end-points of the edge lie in the same tree, that edge, if added, will contribute to a cycle in that tree. In Kruskal's we ignore this edge. However, we can use this edge to detect non-uniqueness. Once a cycle is detected, then in that tree, we look for another edge that has the same weight as out current edge. If there exists another such an edge, that leads to the conclusion that it is not unique, since the other edge could have been replaced with the current edge, still yielding the minimum spanning tree.
**PseudoCode:**

```
KRUSKAL(G(V,E,W)):
    // Graph represented as triples(u,v,w), where u,v are the edge end-points and w the edge weight w(u,v).
    BuildHeap(G(V,E,W))
    foreach u ∈ V
        Make-Set(u)
    T = NULL // This corresponds to the spanning set.
    while (|F| = n-1) // if still spanning
        (u,v,w) = ExtractMin(H)
        Uset = Find(u)
        Vset = Find(v)
        if (Uset ≠ Vset)
            T = T ∪ (u, v)
            Merge(Uset, Vset)
        else // indicates part of same tree, hence contributes to a cycle
            bool x = search(w, Uset) // search for the same weight value in the Uset
            if (x=true) return MST_NOT_UNIQUE
    return MST_UNIQUE
```

**Correctness:** The correctness is same as Kruskal's algorithm. Consider that Kruskal's algorithm gives a minimum spanning tree, M=(V,F), where F is the set of the edges in the MST. Then, if there exists an edge, e, in E-F, such that adding e to the M yields a cycle, C, and if weight(e)=m is the lowest edge weight in $F \cap C$, then we can swap that edge of $F \cap C$ with e, yoielding another spanning tree. This is exactly the part executed by 'search' and hence guaranteed correctness. We searches for same weight edges in the Uset corresponding to the edge of the current edge when a cycle is detected, and if found we say the MST is not unique, else the MST is unique. Thus, we have modified the Kruskal's algorithm to dynamically find the MST uniqueness.