CS5800: Algorithms Spring 2018
Assignment 7.2

Saptaparna Das

April 10, 2018

(a) Algorithm: The underlying problem is to convert one minimum spanning tree to another such that all the intemediate steps also maintain spanning tree structure.
Let newedges and obsoleteedges are two lists to maintain edges to be added and to be removed in original spanning tree B from T. Let from and to are two edges to store result.

```
Step 1: Scan each edge of T and check if it is present in B. If not present,
          add it to newedges.
Step 2: Scan each edge of B and check if it is present in T. If not present,
          add it to obsoleteedges.
Step 3: For each edge (u,v) in obsoleteedge, do the following:
          Step 3.a: remove (u,v) from B and from obsoleteedge list.
                       Store (u,v) in from.
                       Mark all vertices unvisited in the graph.
          Step 3.b: Start DFS on any random vertex only once. Mark vertices
                           as visited in the process.
          Step 3.c: Once DFS ends, for each edge (u,v) in newedge, do the following:
                    Step 3.c.1: If u and v vertices are both visited or both unvisited,
                                  do nothing.
                    Step 3.c.2: Else if u and v are combination of visited and unvitied,
                                  add that edge to B.
                    Step 3.c.3: Remove (u,v) from newedge list.
                                  Store (u,v) in to.
                    Step 3.c.4: Print move wire "from" to "to"
```

(b) Since, total numbre of edges in a spanning tree is $|V|-1$ where V is number of vertices in graph, this number will always be same for any spanning tree. Hence, B and T will always have same number of edges and is finite. In worst case if all edges are different in T, it will take V-1 exchanges to transform B to T. So, the algorithm will always terminate.
Now lets say (u,v) be an edge in T which is not present in B. So, if we add (u,v) in B it will form a cycle since B is also spanning tree.
So, there must an edge in B which is not there in T, removing which will break the cycle. Say that edge is (u',v').
Hence aftr one exchange if the new tree is B', we can say, $B' = B + (u, v) - (u', v')$.
At this point if B' = T then the process ends otherwise we look for next edge to be exchanged. So the process terminates when B' transforms to T and takes maximum of V-1 steps to do so.

(c) Since, in every iteration the algorithm makes sure that no cycle is formed and all vertices are still connected, we can say the messaging quality won't go down temporarily.

Analysis and explaination: Total time taken to make newedge and obsoleteedge lists is O(E). Total time taken to do DFS is O(V+E). This is done for E times in worst case.
Hence, total time is O(E(V+E)). Now E = V-1 for spanning tree. So, total time is $O(V^2)$
Space complexity is O(V+E) in DFS which is O(V) for spanning tree.

In the given algorithm, it basically removes one edge from source spanning tree, which will always partition the vertices in two sets. Now doing DFS on any random node only covers either of the partitions. So, there must be an edge in T which connects both these partitions to a spanning tree. For that edge one side will be visited and one will be unvisited in earlier DFS process. Hence, we add that edge.