HW5: Find the MST

Describe your algorithm in words：

Step 1:

Once the edge e = (u, v) is deleted from G, remove the e = (u, v) from MST T as well to get two separate trees Tu and Tv.

Step 2:

Start a DFS or BFS from vertex u in Tu and mark all reachable vertices as part of Tu, and start a DFS or BFS from vertex v in Tv and mark all reachable vertices as part of Tv.

Step3:

Iterate all the edges in the graph G' to find the cut set that could connect Tu and Tv: for each edge (x, y), if one vertex is marked as part of Tu and the other vertex is marked as part of Tv, the edge (x, y) connects a vertex in Tu to a vertex in Tv, which means it should be added to the cut set.

Step4:

For all the edges in the cut set, iterate them to find the edge e' = (x', y') that has the minimum weight.

Step5:

Add the selected e' to the disjoint trees Tu and Tv to form a new MST for G'.

Justify its correctness ：

1) After removing the e = (u, v) from MST T to get two separate trees Tu and Tv, both Tu and Tv are still an MST for their vertices, so Tu and Tv need to be contained in the new MST for G'.
2) The cut property of MSTs states that for any cut of the graph, the minimum weight edge that crosses the cut and does not form a cycle with the already included edges in the MST must be part of the MST. So what we need to do is to find an edge with the minimum weight that can connect Tu and Tv.
3) By DFS or BFS, we can identify all the vertices that can be explored from in Tu and Tv, thus all the edges in the cut set can be found.
4) Once we find the edge e' = (x', y') in the cut set that has the minimum weight, it must be contained to form a new MST for G'. So by adding it to connect Tu and Tv, the new MST is formed.

Analyze and state its running time. ：

Given n = ∣ V ∣ and m = ∣ E |.

Step 2: Run DFS or BFS to identify all the vertices to determine which side they belong to: O(n + m).

Step 3: Form the cut set need to iterate all the edges in G': O(m).

Step 4: Selecting the edge with minimum weight in the cut set: O(m).

Therefore, the overall running time complexity for this algorithm takes O(n + m).