CS577 Homework4

William Shu, Haoyi Lu

1. (a) Spanning tree + edge 🡺 special graph

By definition: spanning tree 🡺 no cycle and for any two nodes, there exists a path

Then we add an edge <a, b>

Because there is already a path between a and b in the spanning tree. This new edge creates a new path, which forms a cycle.

Originally, there was no cycle in the spanning tree. After adding ONE edge, this operation can only add one cycle. Then the new graph has exactly one cycle.

Then we got add an edge to a spanning tree, the subgraph is special.

Special subgraph 🡺 spanning tree + edge

Special subgraph 🡺 connected, exactly one cycle

Choose any edge in the cycle, such as remove the edge <a, b>.

The node a and b are still connected because even after removing the edge, there is still a path from a to b. Meanwhile, all the other nodes are still connected.

In result, a subgraph is special iff it can be obtained by taking a spanning tree and adding one edge.

(b) findMinWeight(E, V):

Priority queue pq = {};

S = {};

T = {};

Add node (u, v), whose weight is the lowest, into pq

Add v to S

Add (u, v) to T

While S != E:

Remove the minimum-weight edge from pq;

If u in S and v is not in S

Add v to S

Add (u, v) to T

Add edges of v to pq

Add the least weight edge which is not in T into T

Return special subgraph

Correctness:

Because Prim’s algorithm can correctly return the minimum weight spanning tree and from part a, we proved that adding an edge to a spanning tree can get a special subgraph. Now we need to prove that adding a minimum edge to a minimum weight spanning tree can get a minimum special subgraph. Since we cannot find a subgraph whose weight is lower than the minimum weight spanning tree and we are adding the lowest weight edge, which is from T but not in the spanning tree. Then the sum is still minimum. Hence, our algorithm is correct.

Running time:

The step of removing the element from pq is O(logN), all the initialize steps are O(1), and we used a while loop, which contains removing step, then it becomes O(NlogN).