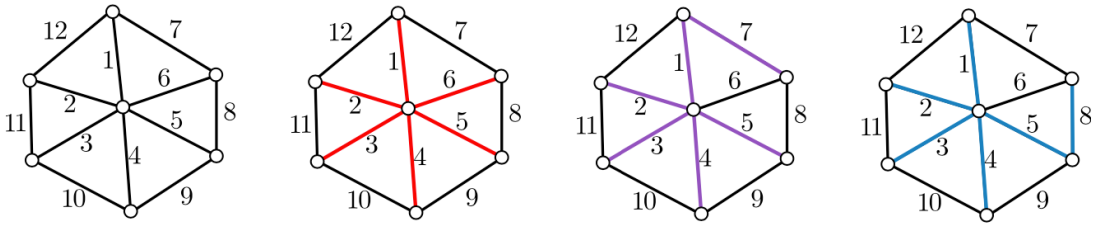
**Report**

## A. Problem Description

The following figure shows a graph, and its first, second, and third minimum spanning trees. Design algorithms for computing the second and third minimum spanning trees.




## B. Algorithm Description

- **First minimum spanning tree:**

For first minimum spanning tree, we use Kruskal Algorithm to find first minimum spanning tree for undirected graph G. The Kruskal algorithm is as follows:

| |
|---|
| Graph G as the number of V vertices and E edges;<br>Create new graph G_1 with V vertices same as graph G, and without edge;<br>Sort the E edges in graph G from to large with the value of edge weight;<br>Loop start:<br>Each edge is traversed from the edge with the smallest weight until all nodes in the Graph are in the same connected component.<br>If the start node and end node in edge_i in graph G_1 are not in same connected component, then:<br>Add edge_i into graph G_1<br>End |

**Correct proof: (mathematical induction)**

When n=1, we can find MST, because the graph just has one vertex.

If the Kruskal algorithm is suitable for n≤ K order graph, then, in K +1 order graph G. Let's merge the two ends of the shortest side, a and B, and merge u and V into one point, v prime. I'm going to connect the edges that were connected to u and v to v prime. In

this way, we can get a k order graph G'(u,v merge is k+1 less edge), G' minimum spanning tree T' can be obtained by Kruskal algorithm.

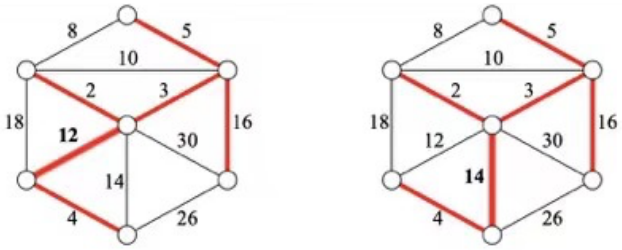
We show that T'+{< u, v >} is the minimum spanning tree of G.

By contradiction, if T'+{< u, v >} is not a minimum spanning tree, the minimum spanning tree is T. W (T) (T < W '+ {< u, v >}).

Obviously, T should include < u, v >, otherwise, it can be added to T with < u, v >, to form a ring, delete the original arbitrary edge on the ring, to form a smaller weight spanning tree. The T - {< u, v >}. It's the spanning tree of G prime. Thus, W (T - {< u, v >}) < = W (T '). So W(T)<=W(T')+W(< u, v >)=W(T'+{< u, v >}). Therefore, if it doesn't work, T'+{< u, v >} is the minimum spanning tree of G. Kruskal algorithm is also applicable to k+1 order graph. Consequently, Kruskal algorithm is correct.

- **Second minimum spanning tree:**

We design algorithm for second minimum spanning tree based on first minimum spanning tree. The example for change first minimum spanning tree to second minimum spanning tree is as follows:




The algorithm is as follows:

A: Set |T| = +infinite

Set E_new = -1 and E_old – 01

For each edge e that is not in first minimum spanning tree, do:

-Add the edge to the tree, creating cycle;

-find k the maximum weight edge in the cycle such that k is not equal to e.

-remove k

-compute the change in the tree weight alpha=weight(e) – weight(k).

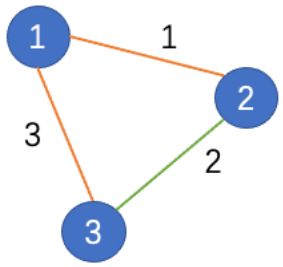
-if alpha < |T| then |T| = alpha and E_new = e and E_old = k.

The new tree is the one that results in from replacing E_old by E_new.

**Correct proof:（Based on first minimum spanning tree）**

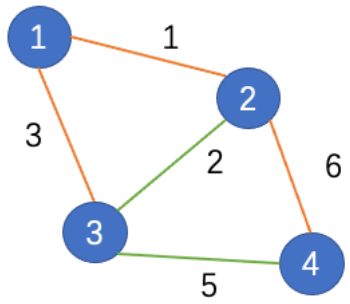
We have proved the algorithm for first minimum spanning tree is correct. The algorithm for second minimum spanning tree is based on first minimum spanning tree. From first MST to second MST just has one situation is as follow:

From the figure below, we assume two orange edges are in first MST, and green edge can also connect node 2 and node 3. Thus, we can have a circle with two orange edges and one green edge. From these three edges, we can remove one edge in first MST with minimize weight value and change it to green edge, then we can get a new MST, and the new MST for node 1, node 2, and node 3 is the second MST. Use this method we can iterate all edges which not in first MST, and get a list which store all second MST. And then, if we get the minimize value in this list, that has only one MST is the second MST. Consequently, the algorithm is correct.




- **Third minimum spanning tree:**

We design algorithm for third minimum spanning tree also based on first minimum spanning tree. The algorithm for third minimum spanning tree is like the algorithm for second MST. However, the third minimum spanning tree has two situations for each edge in first MST.




From the figure above, we assume orange edges are in first MST, and green edges are

in graph G. In order to build third MST, first situation is: remove the edge in first MST with minimum weight value in circle with node 1, node 2, and node 3, and change it to edge which connect node 2 and node 3, we can get MST_1. Second situation is: remove edge which connect node 2 and node 4, and change it to minimum weight value in circle with node 2, node 3, and node 4., we can get MST_2. And then, we can get min(MST_1, MST_2) to get only one MST, and this MST is the third MST, because for basic situation in first MST just have these two situations, and get the minimum situation is the final solution for third MST.

**C. Algorithm Complexity:**

- First MST: O(ElogE)
- Second MST: O(E)
- Third MST: O(V^2ElogV)
- All algorithm: O(ElogE) + O(E) + O(V^2ElogV) = O(V^2ElogV)

**Description:**

- First MST:

1. The edge set A initialized to generate the tree is empty set: O (1)

2. For each vertex in the set, initialize its set to itself: O (V)

4. Sort edges by weight: O (ElogE)

5. Sort the edge from small to large, to judge if this side even the two vertices is not in the same collection, will this side to join the edge set A spanning tree, and even by the edge of two vertices u and v is the set of A Union operator, so the number of edge set in circulation to the spanning tree of n - 1 stop: O (v + E) alpha (v).

Therefore, the time complexity for first MST is: O(ElogE)

- Second MST:

We build second MST based on first MST, and we need to iterate all edges which not in first MST and change edge in first MST to get final result. Thus, the algorithm for building second MST time complexity is O(E).

- Third MST:

We build second MST based on first MST, and we need to iterate all edges which not in first MST. Also, for each loop we need to do DFS for graph G to find all sub third

MST for each sub graph in Graph G. The DFS time complexity is V^2. Thus, the time complexity for third MST is: O(V^2ElogV).