CS325 - Group Assignment 4
Group: 321gogogo
Group members:
Yu Chien, Lin
Huayue Sun
Xiaoru Chen
2020-11-24

Description of our algorithm:

We solve this problem by using prim algorithm to find the lightest minimum spinning tree first. Then we model into a Lowest Common Ancestor (LCA) problem to find the second minimum spinning tree, we remove a single edge from the lightest minimum spinning tree and replace it with another.
Lastly, same use of modeling for the third minimum spinning tree.

STEP 1:

Read from the input file, get variable ***vertices*** and put the matrix into a Two-Dimensional array ***primgraph***.

STEP 2:

Variables:
vertices: represents the number of vertices of the graph.
primgraph: holds the weights between every connected vertex.
charlist: a set that stores (parent, child), and the weight between them.
mstEdgelist: the edges that were chosen to walk.
mst_sum: represents the weight of the lightest MST.
sec_sum: represents the weight of the second MST.
third_sum: represents the weight of the third MST.

Prim Algorithm:
For the **lightest** minimum spinning tree:
Starting vertex is 0.
Push (null, 0), 0 into the ***charlist,*** since vertex 0 has no parent and the weight to itself is 0.
While ***charlist*** is not empty:
    Pop (parent, child) with the smallest weight**.**
        If child is not marked:
            Mark child
            Add (parent, child) to ***mstEdgelist***

            For all (child, child's child):
                Push (child, child's child) into ***charlist***

Return the sum of the weights stored in ***mstEdgelist.***

Find the **second** minimum spinning tree:

For each edge that is not in our lightest MST, add it to the lightest MST, then we create a cycle:
Find edge k with maximal weight in the cycle

Add the value of the weight difference between k and e to our light MST weight.

Return the best (smallest) weight we found.

For the **thrid** minimum spinning tree:

For each edge that is not in our second MST and the lightest MST, add it to the second MST, then we create a cycle:
Find edge k with maximal weight in the cycle

Add the value of the weight difference between k and e to our light MST weight.

Return the best (smallest) weight we found.

Note:
UnUsedEdge. Length: represents weight of unused edges
FindMaxNode.Length: represents the weight of the largest edge found

STEP 3:

Write out ***mst_sum, sec_mst,*** and ***third_mst*** into an output file.

Running time analysis:

Prim algorithm that found the lightest MST time complexity:

T(n) = O(VlogV + ElogV) = O(ElogV) since pushing vertices with weight into the charlist takes logarithmic time.

Algorithm that found the second and third MST time complexity:
The time complexity to iterate and find the smallest difference between edge k and e:
T(n) = O(ElogV)

Therefore, our running time complexity is:

T(n) = 2 * O(ElogV) = O(ElogV)

Correctness:
Correctness of computing lightest MST:

Base case:
At the starting vertex, it has no parent and the weight to itself is 0. Child vertex 0 is not marked. Therefore, we pop the (parent, child) with the smallest ***weight = 0*** and store the edge weight 0 in the ***mstEdgelist.*** It correctly computes the lightest path from null parent to vertex 0.
Induction step:
For k^{th} vertex, we know its children and the weights between the k^{th} vertex and children. We pop the (parent, child that has not been marked) with the smallest ***weight*** and store it in the ***mstEdgelist***. It correctly computes the lightest path from parent k to child k+1.

Correctness of computing second and third minimum spinning tree:
For the second MST:
When we add an edge that is not in the lightest minimum spinning tree, we created cycle in the graph.

Sppose the ***sec_mst*** we found is incorrect.
Then, there is a $weight'$ holds: ***mst_sum*** < $weight'$ < ***sec_mst***
Then $weight'$ - (***mst_sum***) < ***sec_mst*** – (***mst_sum***)
$weight'$ < ***sec_mst***
sec_mst = ***mst_sum*** – 1 edge weight + new edge weight = ***mst_sum*** + (new edge weight – 1 edge weight)
However, (new edge weight – 1 edge weight) is the minimum possible substitution we found.
Then, ***sec_mst*** $\not>$ $weight'$
Contradiction.
Therefore, ***sec_mst*** we found is correct.

For the third MST:
When we add an edge that is not in the lightest and the second minimum spinning tree, we created cycle in the graph.

Suppose the ***sec_mst*** we found is incorrect.
Then, there is a $weight'$ holds: ***mst_sum*** < $weight'$ < ***sec_mst***
Then $weight'$ - (***mst_sum***) < ***sec_mst*** – (***mst_sum***)
$weight'$ < ***sec_mst***
sec_mst = ***mst_sum*** – 1 edge weight + new edge weight = ***mst_sum*** + (new edge weight – 1 edge weight)
However, (new edge weight – 1 edge weight) is the minimum possible substitution we found.
Since the new edge is not is the lightest or the second minimum spinning tree, we will not able to create MST back to the lightest when we compute and add (new edge weight – 1 edge weight).
Then, ***sec_mst*** $\not>$ $weight'$
Contradiction.
Therefore, ***sec_mst*** we found is correct.