Report for Problem 4

a) Analysis:

Choice of algorithm: Kruskal's

Data structure:

Use a disjoint-set data structure (Union&Find) to keep track of which vertices are in which components. Perform O(V) operations, as in each iteration connect a vertex to the spanning tree, two 'find' operations and possibly one union for each edge. Even a simple disjoint-set data structure such as disjoint-set forests with union by rank can perform O(V) operations in O (V log V) time.

Time Complexity:

1) ComputeMST ():

   Step1: sorting edges using function sorted () in python.

   $O(|E|\log|E|) = O(|E|\log|V|^2) = O(2|E|\log|V|) = O(|E|\log|V|)$

   Step2: processing edges using disjoint sets.

   $2|E|*T(Find)+|V|*T(Union)=O((|E|+|V|)\log|V|) = O(|E|\log|V|)$

   Total running time: $O(|E|\log|V|)$

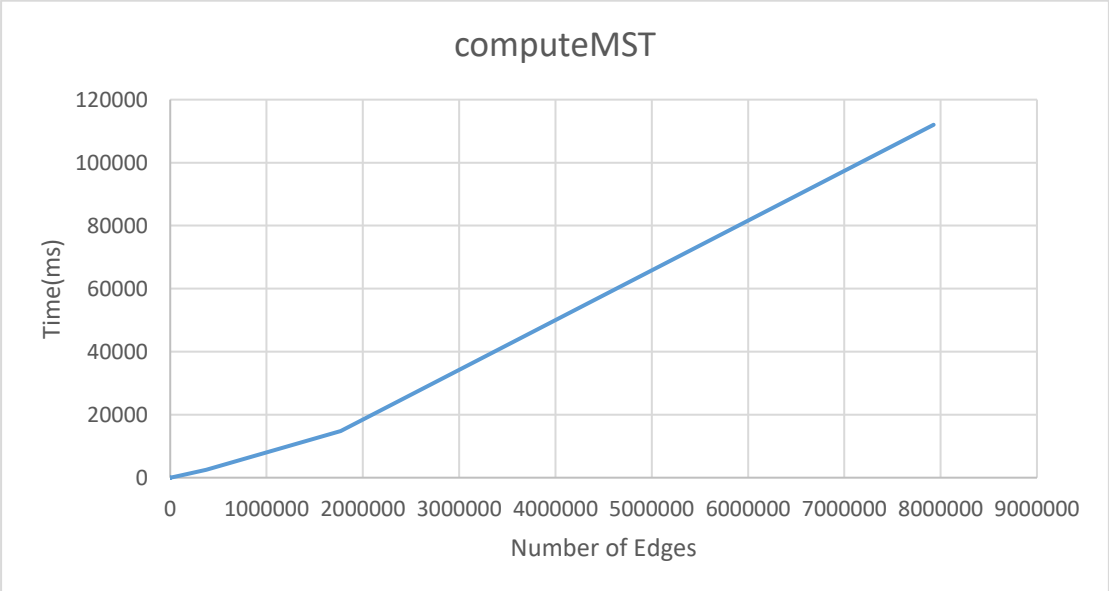
2) RecomputeMST ():

   using DFS to find a path from node a to node b in the existing MST. There is only one path for a node to another in a MST, so the time complexity is O(|E|).

b) Plots:

Static MST calculation:

The theoretical complexity is O(|E|log|V|). The number of V in the graphs is relative small(no more than 100000), so the running time is approximate O(e). The plot therefore shows a linear relationship between number of edges and time.



Dynamic MST calculation:

Theoretical complexity of this function in O(|E|). The graph shows several lines with different slopes, not a simple linear relationship. When original edges are less than 10000, inserted edges have great impact on the number of total edges. When we insert 1000 edges and edges in the original graph are more than 100000, 1000 is relatively very small to the number of original edges so number of total edges in the new MST graph is not influence apparently. Thus, time is in a linear relationship to number of original edges.