CSE 6140

Assignment 1

Bhanu Verma - 903151012

**Question 4** - Implementation of Kruskal’s Algorithm for finding minimum spanning tree in a multi-graph.

Here is a list of data structures that I have used for my implementation:

* *Union-Find* - Best time complexity for Union-Find can be reduced to by using weighted quick-union with path compression and when done for *m* nodes, we get as *m\*.*
* *Dictionary* - I have used this for avoiding any computation for those edges which have been added to graph but exist multiple times with different weights. But as we have already sorted the edges in the order of increasing weight, we can be assured that if an edge with same *source* and *destination* is observed again, then it’s weight will be more than the the edge that was observed for the first time and hence we can just ignore it. Dictionary helps us in detecting collisions in *(1)* time.

**Time Complexity**:

Before we start calculating time complexities for both the methods, let’s define few variables.

- total number of edges

- total number nodes

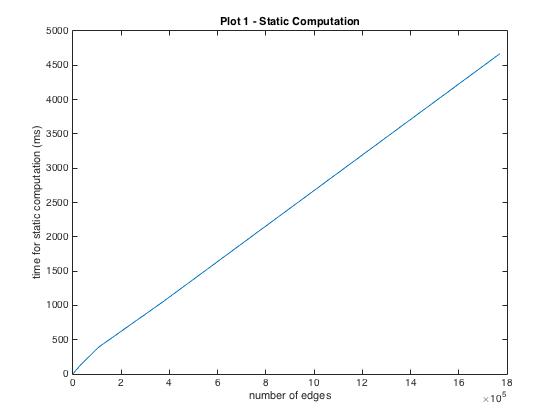
where 🡪 Eq. 1

* *computeMST( )* - First step is sorting the edges as per their weights, time complexity for this step is , but from Eq. 1 it can be easily deduced that , so time complexity for this becomes . Now, we just need to find time complexity for union find. Starting from an empty data structure, any sequence of m unions (total number of edges) and find operations on n objects (total number of nodes) takes . Moreover, for this case can be considered as constant hence for union-find becomes linear i.e. . So, for computeMST( ), .
* *recomputeMST( )* - Let’s list the algorithm for recomputing MST:
  + (u, v) is the new edge e\*, find shortest path, p, between u and v in T, MST.
  + find the edge in p that has the maximum weight .
  + If , then  the MST for , with
  + If not, then  is the MST of  (graph + edge)

Only step that can take some significant time in this algorithm is find shortest path, which takes linear time i.e. for a minimum spanning tree.

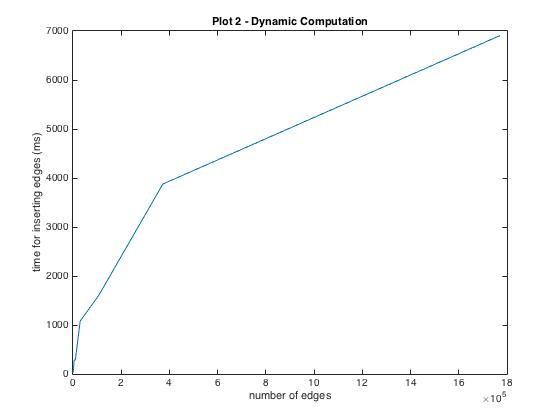
**Static Computation Plot**

Our forstatic computation is but we see a linear graph for our plot. As number of nodes are not too much for the given graph, can be considered a constant and does not really change the linearity and hence we get a straight line.



**Dynamic Computation Plot**

Our for dynamic computation is **n i.e. linear** and we should again expect a linear graph. But see a small curve in the beginning of the graph and that can be explained by the fact that for the first 4 graphs we are adding more number of edges in comparison to the initial number of edges. But as the initial number of edges get more than 1000, we see that our plot becomes linear.



**References**:

1. <https://en.wikipedia.org/wiki/Boyer%E2%80%93Moore_majority_vote_algorithm>
2. <https://www.cs.princeton.edu/~rs/AlgsDS07/01UnionFind.pdf>
3. Code Reference for UnionFind: <https://www.ics.uci.edu/~eppstein/PADS/UnionFind.py>