CSE 5311 DESIGN AND ANALYSIS OF ALGORITHMS

REPORT ON
IMPLEMENTATION OF MINIMUM SPANNING TREE USING PRIM'S AND KRUSKAL'S ALGORITHM

VIVEK ARVIND BALAJI (1001108300)
ESHWAR RAVINDRAN (1001120227)

OBJECTIVE:

To implement the minimal spanning trees for Prim's and Kruskal's algorithms, and compare them on the basis of their running times based on different datasets of various sizes.

PROBLEM:

MINIMUM SPANNING TREE:

KRUSKAL'S ALGORITHM (Contributed by Eshwar Ravindran):

We implement Kruskal's algorithm to compute the minimum spanning tree of a given graph and identifying the impacts of data size to implement Kruskal's algorithm. There are three techniques available for the implementation of Kruskal's Algorithm. They are:

- The naive method where we use DFS to check if the two vertices are in same or different trees.
- Union-find without path compression and union by rank heuristic.
- Union find with path compression and union by rank heuristic.

But, we have used the second method, Union find without path compression and union by rank heuristic.

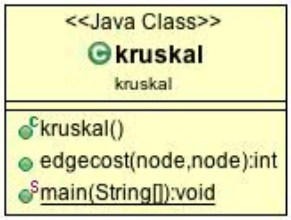
IMPLEMENTATION:

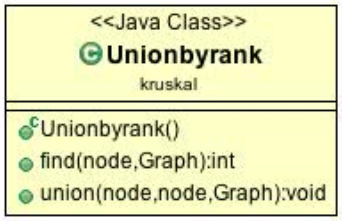
We have used a public class that defines constructors for necessary classes. Also the main function receives program arguments as input file text and output file text. The Euclidean distance is calculated for all the edges with help of member function edgecost (node a, node b). Each edge is initialized using the parameterized constructor for the class Edge defined in edge.java. The class Unionbyrank defines necessary methods and variables to execute the functionality as specified in the name. The program uses different data structures like hash to keep account of the different nodes and an array list to store all the visited nodes.

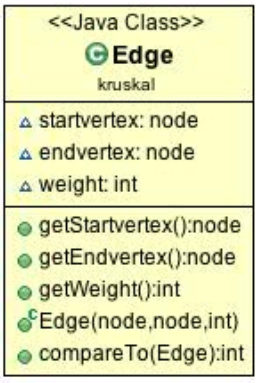
CLASSES:

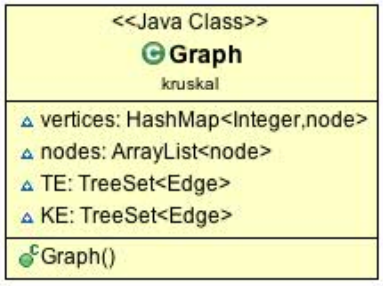
Edge, Node, Graph: To create the graph structure with edges and nodes.
Union by rank: To do operations like, Create-set, find and union.

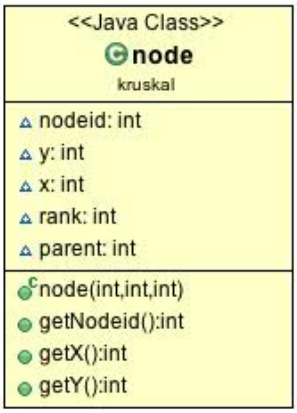
CLASS DIAGRAM:



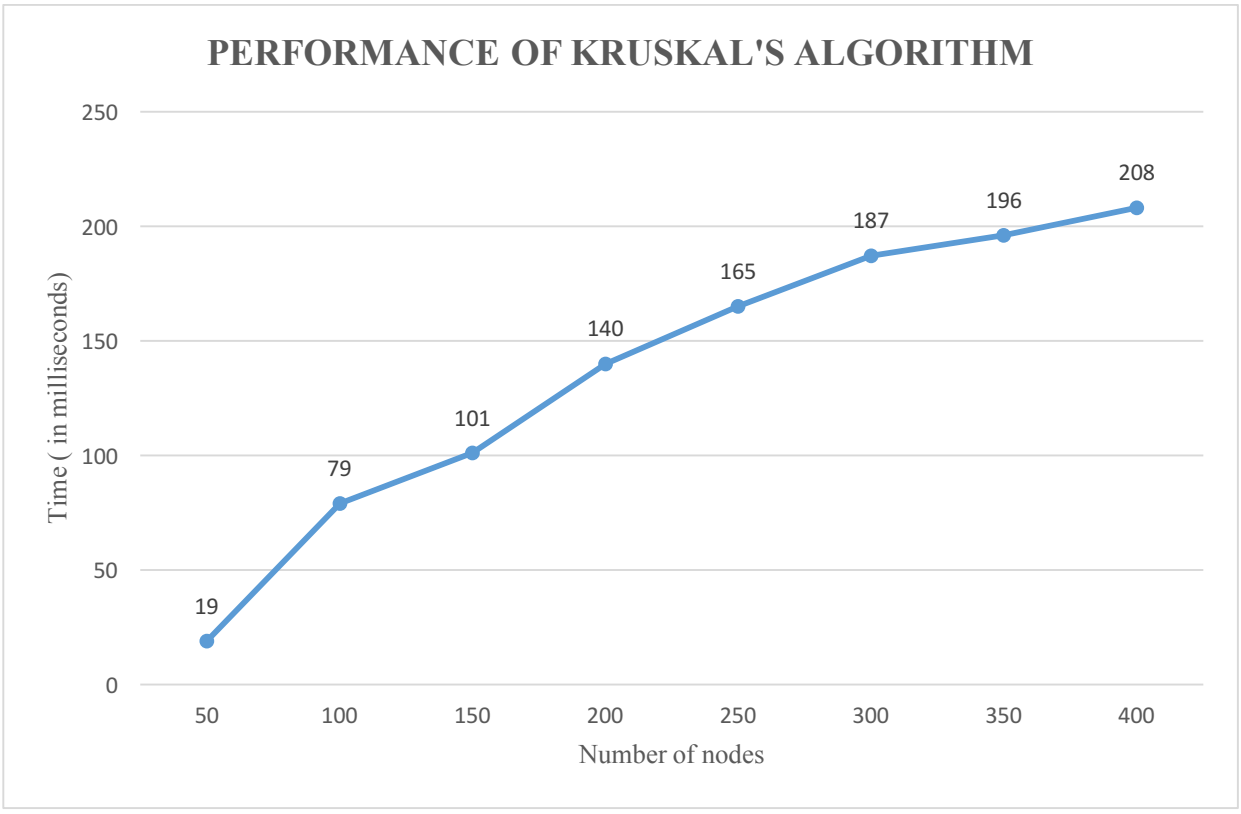








ANALYSIS:



PRIM'S ALGORITHM (Contributed by Vivek Arvind Balaji):

In this method we are implementing prim's algorithm to construct a minimum spanning tree for a given graph. There are three different methods for implementing prim's algorithm. They are,

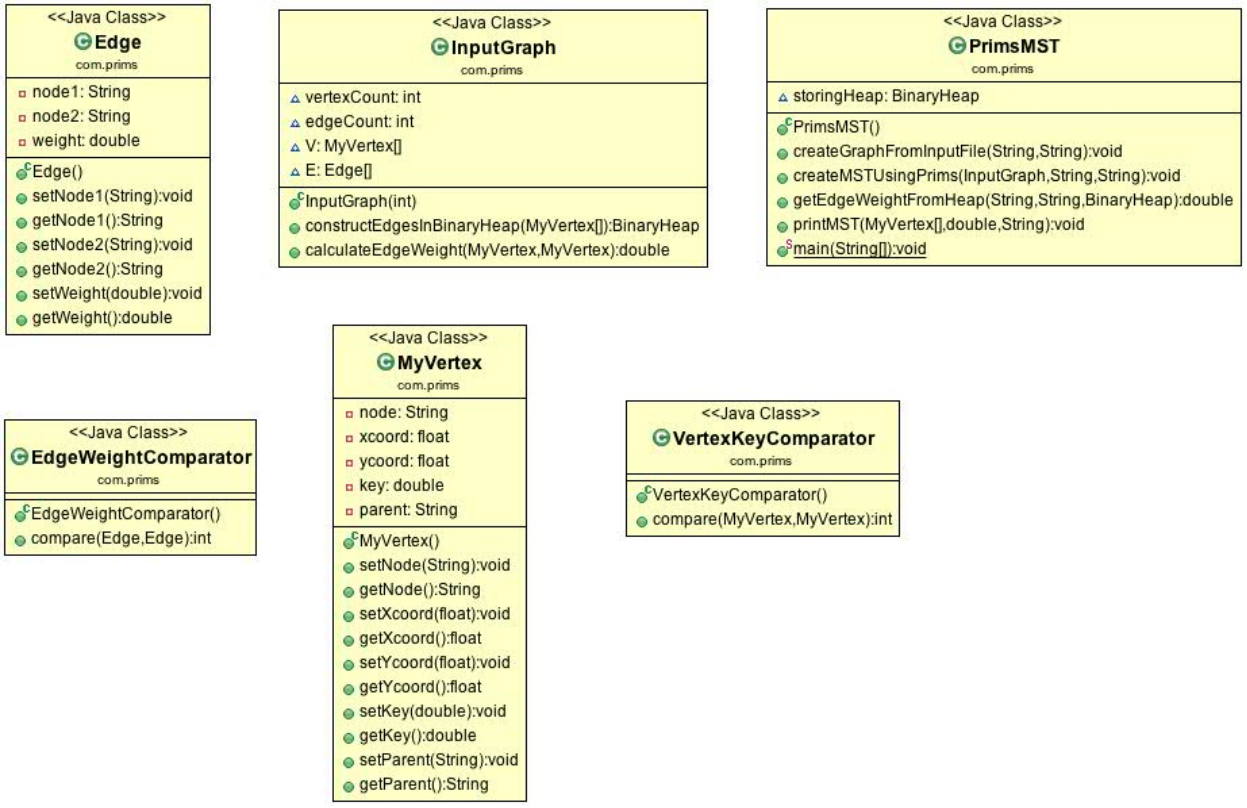
- Storing edges as an unsorted array
- Storing edges as a sorted array
- Storing edges as minimum heap**.**

Out of these three, we have used minimum heap data structure for the computation of minimum spanning tree.

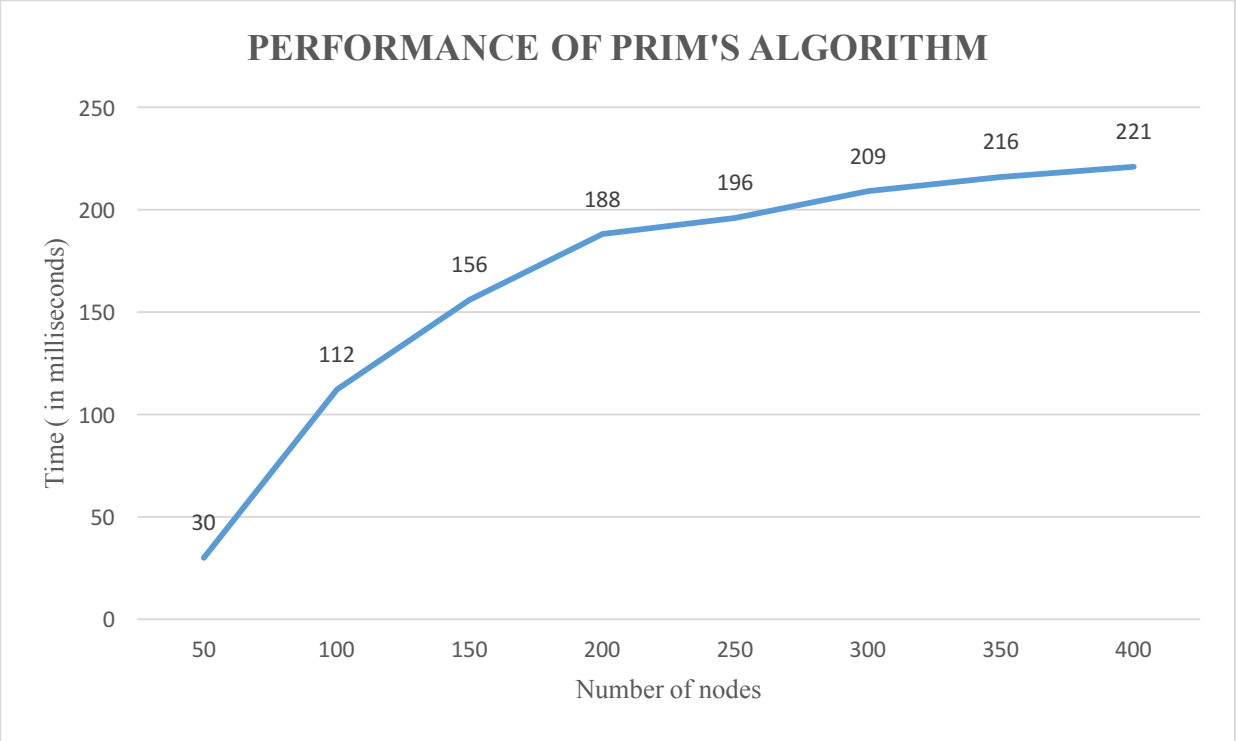
IMPLEMENTATION:

A main class PRIMSMST is used to construct an MST. Another class named EdgeWeightComparator is created to compare the edge weights and VertexKeyComparator is used to compare the vertex objects based on the weights between each nodes. The class MyVertex stores the information of different nodes such as the coordinates. The method createMSTUsingPrims () is used to design the minimum spanning tree with the help of the methods defined in the other classes. The program uses heap for the implementation of binary min heap.

CLASS DIAGRAM:



ANALYSIS:



PERFORMANCE COMPARISON OF PRIM'S AND KRUSKAL'S ALGORITHMS:

For a graph with V Vertices with E edges, Kruskal's algorithm runs in **O (E log E)** time and Prim's algorithm can run in **O (E + V log V)** amortized time, if we use a Heap. Prim's algorithm is significantly faster in the limit when there is a really dense graph with many more edges than vertices. Kruskal's performs better in typical situations (sparse graphs) because it uses simpler data structure.

Thus, it can be observed in the graphs that Kruskal's algorithm performs better in most of the cases, but the curve of Prim's gets towards a dip when the number of nodes gets to 400. However, a graph with 400 is not considered so dense. Therefore, Prim's algorithm is expected to perform better for denser graphs.

CONCLUSION:

After our analysis, we have observed that Prim's algorithm performs better when comes to real world data. For instance, graph search in Facebook would contain millions of nodes. If Prim's and Kruskal's algorithms are applied for such a graph, Prim's algorithm would perform better. This project was done in parts, in which Eshwar Ravindran contributed towards the implementation of Kruskal's Algorithm while Vivek Arvind Balaji contributed towards the implementation of Prim's Algorithm.