# Project Report

Implementation and Evaluation of Graph Theory Algorithms Design and Analysis of Algorithms

Submitted By

Danish Abdullah 17K-3720

Huzaifa Abid 17K-3807

**Abstract**

We are provided with the bandwidth speed between two points and we have implemented some algorithms of graph theory for minimum spanning tree, shortest path and clustering coefficient. The input is taken from the file and the file is parsed to get the useful information about the graph. The user can then select which algorithm to apply on the graph depending on what result he wants.

**Introduction**

In graph theory, the two algorithms used to find the minimum spanning tree are, Prims and Kruskal, and for finding the shortest path there are three algorithms namely, Dijkstra, Bellman Ford and Floyd Warshall. To find the clustering or cliquishness of a graph, local clustering algorithm we is used. We have implemented all these algorithms on weighted directed graphs. The weights on the edges represent the bandwidth of internet connectivity in Mbps. We have to consider the minimum values between the edges to find the minimum spanning tree, shortest path and clustering coefficient.

**Proposed System**

Run main.py

Main Screen

Select File and click OK

Play With Graph Screen

Click on Initial Graph to see the graph of input

Choose any option to view its graph

Upon running the file, main screen will be displayed which will allow the user to select from the given 10 input files using a dropdown menu. When the user clicks OK, the file will be parsed and the Play with Graph screen will be displayed which will allow the user to plot the initial graph, which is of the input file, or select an algorithm to run on the graph and plot the resultant graph.

**Experimental Setup**

Our system has 10 input files with vertices ranging from 10 to 100. The files contain the x and y coordinates of each vertex and the detail of the edges i.e. the vertices between which the edge is and the weight of the edge. There are three weights given in the file but we are only concerned with the bandwidth, hence the file is parsed to filter out the values we need to calculate the results and plot graphs.

**Results and Discussion**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Benchmark | Prims | Kruskal | Dijsktra | Bellman Ford | Floyd Warshall | Clustering Coefficient (Local Clustering) |
| Input 10 | 29.4 | 29.4 | 79.2 | 79.2 | 79.2 | 0.714 |
| Input 20 | 69.75 | 69.75 | 182.4 | 182.4 | 182.4 | 0.208 |
| Input 30 | 102.3 | 102.3 | 234.0 | 234.0 | 234.0 | 0.21233 |
| Input 40 | 161.55 | 161.55 | 441.9 | 441.9 | 441.9 | 0.1375 |
| Input 50 | 163.65 | 163.65 | 539.55 | 539.55 | 539.55 | 0.0894 |
| Input 60 | 231.0 | 231.0 | 1144.95 | 1144.95 | 1144.95 | 0.1185 |
| Input 70 | 247.65 | 247.65 | 950.55 | 950.55 | 950.55 | 0.04714 |
| Input 80 | 294.15 | 294.15 | 807.6 | 807.6 | 807.6 | 0.059 |
| Input 90 | 342.6 | 342.6 | 1284.9 | 1284.9 | 1284.9 | 0.08611 |
| Input 100 | 364.2 | 364.2 | 1261.2 | 1261.2 | 1261.2 | 0.0518 |

**Conclusion**

Prims and Kruskal are the algorithms used to find the minimum spanning tree. Complexity of Prims is O(ElogV) while of Kruskal is O(ElogV + ElogE).

Both of them provide the same output and the complexity of Prims is lesser hence it is much better.

For shortest path, Dijkstra, Bellman Ford and Floyd Warshall are used. Dijkstra does not work on negative weights while Bellman Ford and Floyd Warshall do. Floyd Warshall is used to find the shortest path from all sources to all destinations. Complexity of each is represented as,

Dijkstra O(ElogV)

Bellman Ford O(VE)

Floyd Warshall O(V3)

For graphs with positive weights Dijkstra is the most efficient.

Complexity of Local Clustering Coefficient is O(V3).

**References**

<https://www.one-tab.com/page/cQKvrD_KQGCA8FvPWZptWg>

<https://networkx.github.io/documentation/networkx-1.9/reference/generated/networkx.drawing.nx_pylab.draw_networkx.html>

<https://www.geeksforgeeks.org/python-gui-tkinter/>

<https://stackoverflow.com/questions/28372127/add-edge-weights-to-plot-output-in-networkx>