Project Report

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

Submitted By:

1. K173647 Vishesh Kumar
2. K173636 Vimal Kumar

Contents

Abstract: 3
Introduction: 3
Proposed System: 3
Results and Discussion: 4
Experimental Setup: 5
Conclusion: 6
References: 6
1. *For GUI:* 6
2. *For Algorithms:* 6

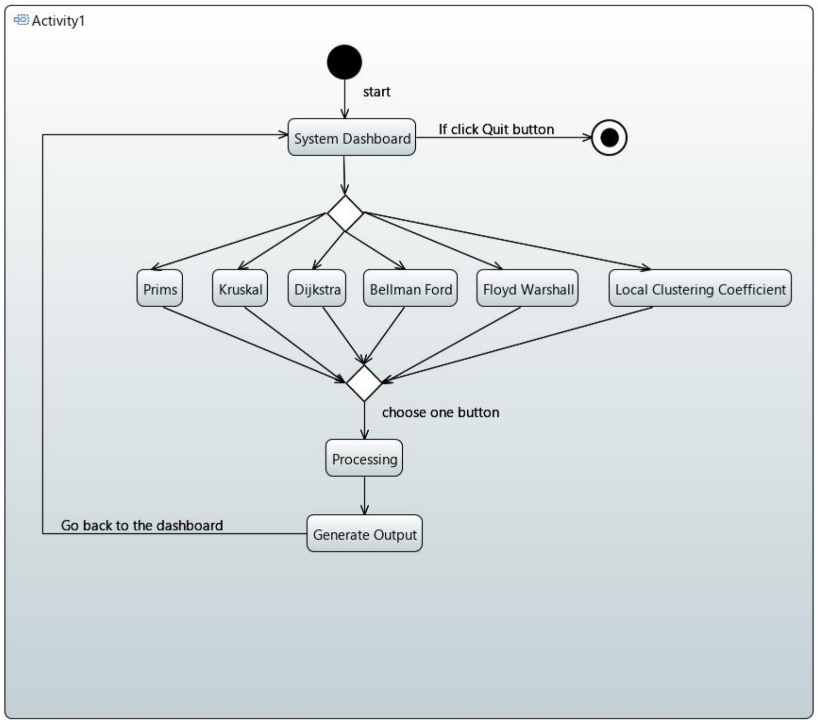
Abstract:

Graph theory algorithms are used to solve many computer science problems including obtaining different kind of paths between connected nodes we are given similar kind of problem as we have to obtain paths by applying 6 different kinds of algorithms, we have to find cost and distance of given inputs and show them in graphs having a good interface for user.

Introduction:

Graph theory algorithms are used in many daily life problems we face, graph theory algorithms are mainly based on finding different kinds of paths between connected nodes. Problems which are based on finding minimum cost, shortest distance, or networking are solved by these graph theory algorithms. One of the application of these algorithms which we use in daily life is GPS, by which we can find path from source to destination having minimum time and distance (Google Maps gives us the minimum distance route from available routes), modelling the social network (Dijkstra is used in finding Open shortest path first) is also an application where these algorithms are used, therefore almost in every field of life graph theory is used.

Proposed System:

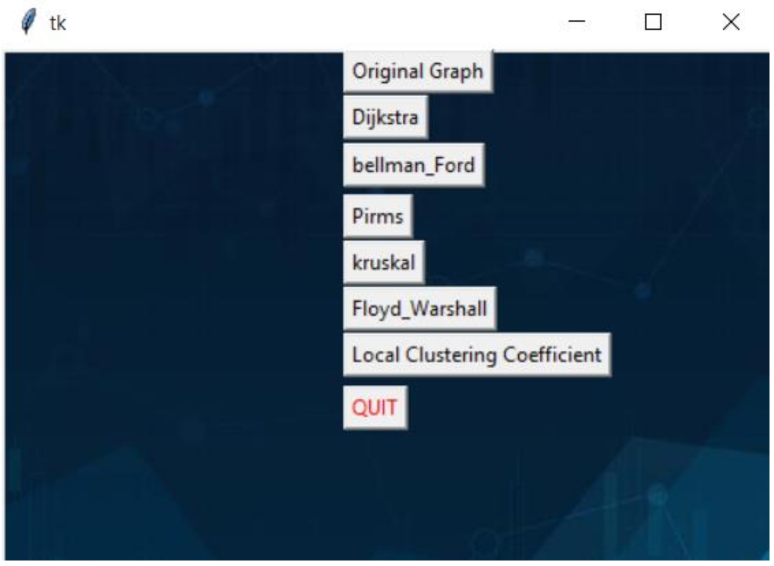


The above diagram shows the overall activity of the system. Firstly, a system dashboard shown to the user. There are some buttons there. User just have to click the button and enter the input file name. Then system will processing the input and return the output as "Graph". Then it goes back to the dashboard screen and wait for next command.

Results and Discussion:

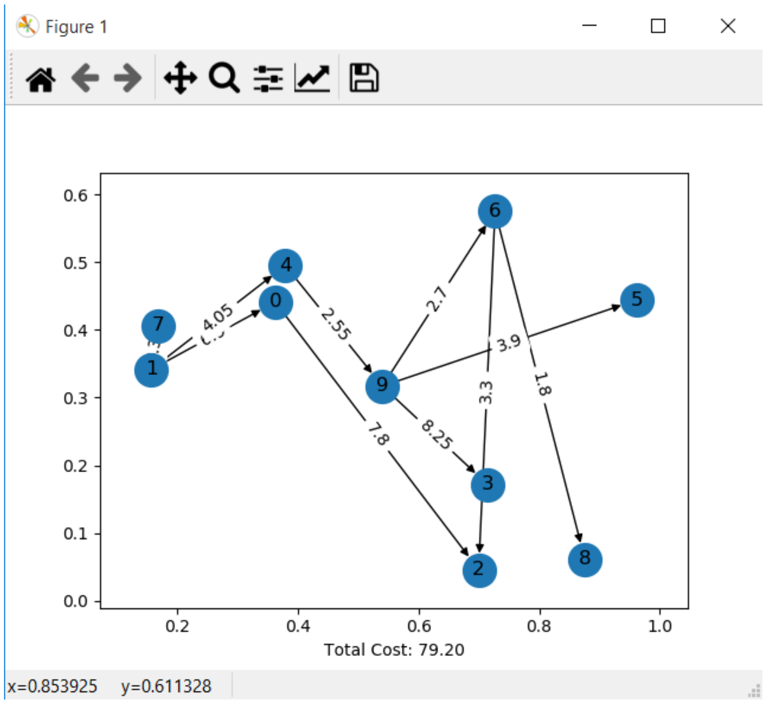
The table below contains the cost of every algorithm using a source vertex.

Benchmark	Prims	Kruskal	Dijkstra Algorithmat Node No 1	Bellman Ford Algorithm At Node No 1	Floyd Warshall Algorithm at Node No 1	Clustering Coefficient (Local Clustering)
Input 10	29.4	29.4	79.2	79.2	79.2	0.713
Input 20	69.75	69.75	182.4	182.4	182.4	0.208
Input 30	102.3	102.3	234	234	234	0.213
Input 40	161.55	161.55	441.9	441.9	441.9	0.138
Input 50	163.65	163.65	539.55	539.55	539.55	0.089
Input 60	231	231	1144.95	1144.95	1144.95	0..0119
Input 70	247.65	247.65	950.55	950.55	950.55	0.047
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	0.086
Input 100	364.2	364.2	1261.2	1261.2	1261.2	0.052



This is the snapshot of the system GUI where we have a list of the following algorithms. You just have to click the button, it asks you for an input file and then return the graph of that particular algorithm.

Here, is an example of Dijkstra algorithm at an input10.txt.



Experimental Setup:

In this project we have to implement 6 different graph theory algorithms and paths through them, there are benchmarks given to us to run and check, there are total 10 input files given, each input files contains nodes with x and y ordinate respectively, and their link between them with weight on every edge. We have to find cost and distance based on weights given, first we have to clean (file parsing) the input files given, according to conditions in input files, then we have to take these files as inputs in all the algorithms, based on the result we have to provide graph to user which the algorithm produced, at the end we have to provide all this in a better interface for user.

Following are the steps to follow in order to acquire the result:

- Clean the given input files (file parsing).
- Take them as inputs in each algorithm.
- Show results.
- Show the graph produced using coordinates given.
- Provide all this in a better interface.

Conclusion:

By running all the algorithms, we get the complexities of the algorithms as follows:

- Dijkstra: O(ELogV)
- Bellman Ford: O(VE)
- Floyd Warshall: O(V^3)
- Kruskal: O(ElogE) or O(ElogV)
- Prims: O(ElogV)
- Local Clustering Coefficient: O(V^2)

Since from the above cost table we conclude that the Prims, and Kruskal gives us a minimum optimal cost. Therefore based on the given benchmark to run Prims, and Kruskal are the best algorithms among all.

References:

1. ***For GUI:***

https://likegeeks.com/python-gui-examples-tkinter-tutorial/#Change-button_foreground-and-background-colors

https://www.geeksforgeeks.org/python-add-style-to-tkinter-button/

https://stackoverflow.com/questions/46253274/tkinter-button-pack-in-the-same-window

https://stackoverflow.com/questions/26852465/calling-a-python-script-on-button-click-using-python-and-tkinter

2. ***For Algorithms:***

https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/

https://www.geeksforgeeks.org/bellman-ford-algorithm-dp-23/

https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5/

https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/

https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/

http://pythonfiddle.com/clustering-coefficient-algorithm/