Algorithm: Assignment 2:

Name: Abhishek Sharma

Professor: Mohan Kumar

**Explanation For Algorithms:**

1. Dijkstra's algorithm: for calculating single source shortest path

Data set Used: Array- to store the Nodes and their adjacent node list.

For making the Breath First Search I maintained the array list corresponding to

each node. This will give the running time complexity of O(V+E).

Second array is for keeping the list of all node for the traversal.

Maintained one index respective to each node to keep track which nodes have been visited.

1. Kruskal’s Algorithm: Minimum Spanning Tree:

Data set used: array: to keep the record of all the edges

Maintained union-find data structure to have minimum time complexity to check for disjoint vertex. This will give the time complexity of find() function approximately O(1).

Maintained one Index for Root of set to each node to check in which set it belongs, and one index.

1. Floyd Warshall’s: All Pairs Shortest Paths:

Data set used: Maintained two dimensional array for keep the adjacency matrix.

1. Transitive Closure :

Data set used: Maintained two dimensional array for keep the adjacency matrix.

Note:

* For data set generator I used the reference provided.
* User need to provide the number of vertices and edges at command line in same order.

I maintained a two dimensional array (matrix[][]) to sore the data matrix get generated and while generating the data we need to keep the sequence of each data set generated and its calculation first before the second dataset get generated.

* Below are the table for providing the CPU time taken by each algorithm for each dataset
* Each time is in milliseconds.

**Table:**

Case 1 Case 2 Case 3

|  |  |  |  |
| --- | --- | --- | --- |
| CPU Running Time  For :  Dijkstra's algorithm | For connection <10  Avg CPU Time = 1 | For Connections>N/2  a) =40 : Avg Time: 0  b) = 60: Avg Time: 0  C) = 100: Avg Time: 1 | For Random Graph: for: 150 Nodes  a)weighted, directed: 42  b) weighted,Non-directed :50  c) nonweighted, directed  :26  d) nonweighted,Non-directed  :18 |

|  |  |  |  |
| --- | --- | --- | --- |
| CPU Running Time  For :  Kruskal's algorithm | For connection <10  Avg CPU Time = 1 | For Connections>N/2  a) =40 : Avg Time: 0  b) = 60: Avg Time: 0  C) = 100: Avg Time: 1 | For Random Graph: for: 150 Nodes  a)weighted, directed: 30  b) weighted, Non-directed: 25  c) nonweighted, directed :37  d) nonweighted,Non- directed  :18 |

|  |  |  |  |
| --- | --- | --- | --- |
| CPU Running Time  For :  Floyd Warshall | For connection <10  Avg CPU Time = 1 | For Connections>N/2  a) =40 : Avg Time: 9  b) = 60: Avg Time: 16  C) = 100:Avg Time:126 | For Random Graph: for: 150 Nodes  a)weighted, directed: 210  b) weighted,Non-directed:200  c) non weighted, directed :190    d) non-weighted,Non-directed  :200 |

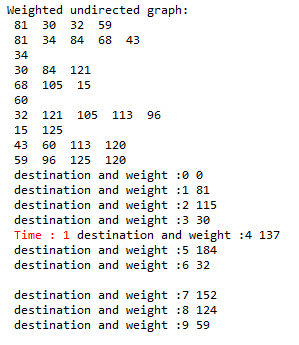
|  |  |  |  |
| --- | --- | --- | --- |
| CPU Running Time  For :  Transitive Clouser | For connection <10  Avg CPU Time = 1 | For Connections>N/2  a) =40 : Avg Time: 11  b) = 60: Avg Time: 15  C) = 100:Avg Time:140 | For Random Graph: for: 150 Nodes  a)weighted, directed: 200  b) weighted,Non-directed:205  c) non weighted, directed :190    d) non-weighted,Non-directed  :180 |

**Example execution:**

Weighted undirected graph: for nodes = 10 edge = 16

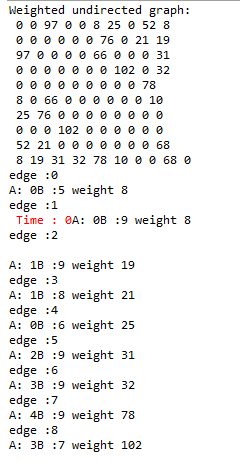
1) Dijkstra's Algorithm:

The out-put representation of minimum traversal weight from source (0 ) to each other vertices :



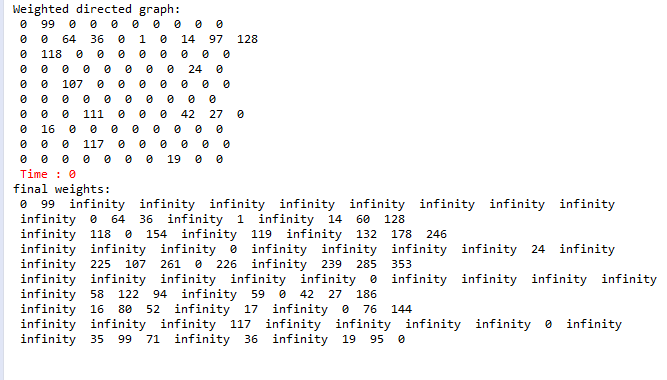
Kruskals algorithm:

The representation of out put is which all edges has been included, then its two vertices (A and B) and then the weight for that edge.



Floyd Warshall Algorithm:

The out put representation is simply in form of matrix which gives the minimum weight between the two nodes.



Transitive Clouser:

In this also the out-put is the matrix describing if the there is path between any two pair of vertices.

