**Assignment No : 8**

**Title :** ShortestpathusingDijkstra'salgorithm**.**

**Course Outcome :**

**CO1(**C214447.1**):** Analyze algorithms and to determine algorithm correctness and time efficiency class.

**CO2(**C214447.2**):** Implement abstract data type (ADT) and data structures for given application.

**CO3(**C214447.3**):**Design algorithms based on techniques like brute -force, divide and conquer, greedy, etc.).

**CO5(**C214447.5**):** Analyze of algorithms with respect to time and space complexity.

**Date of completion :** 17/12/2021

**Accessment Grade/Marks :**

**Accessor’s Sign with Date :**

**Assignment No: 8**

**Title:** Shortest path using Dijkstra's algorithm

**Aim:** Represent graph using adjacency matrix and find shortest path using Dijkstra's algorithm.

**Objective:** Represent graph using adjacency matrix and find shortest path using Dijkstra's algorithm.

**Problem Statement:**

Represent a graph of the city using adjacency matrix /adjacency list.

Nodes should represent the various landmarks and links should represent the distance between them. Find the shortest path using Dijkstra's algorithm from single source to all destinations.

**Course Outcome:** CO Number: Applicable CO : Blooms Taxonomy Category

**Requirements:** (Components / Digital Kits / Platform / Software / Hardware)

Platform :- Online GDB Compiler

**Theory / Procedure / Diagrams / Circuits:**

# Dijkstra's algorithm :-

Dijkstra’s algorithm is very similar to Prim’s algorithm for minimum spanning tree. Like Prim’s MST, we generate a SPT (shortest path tree) with a given source as a root. We maintain two sets, one set contains vertices included in the shortest-path tree, other set includes vertices not yet included in the shortest-path tree. At every step of the algorithm, we find a vertex that is in the other set (set of not yet included) and has a minimum distance from the source.

**Algorithm / Methods / Steps:** (if applicable):

# Dijkstra's algorithm (Algo) :-

**STEP 1:** START

**STEP 2:** First of all, we will mark all vertex as unvisited vertex.

**STEP 3:** Then, we will mark the source vertex as 0 and all other vertices as infinity.

**STEP 4:** Consider source vertex as current vertex.

**STEP 5:** Calculate the path length of all the neighboring vertex from the current vertex by adding the weight of the edge in the current vertex.

**STEP 6:** Now, if the new path length is smaller than the previous path length then replace it otherwise ignore it.

**STEP 7:** Mark the current vertex as visited after visiting the neighbor vertex of the current vertex. **STEP 8:**Select the vertex with the smallest path length as the new current vertex and go back to step 4. **STEP 9:** Repeat this process until all the vertex are marked as visited.

**STEP 10:** STOP

**Input**: (Test Cases / Data sets / Database Links):

Enter the number of Vertices : 5 Enter Vertices and Distance : 0 1 10

Enter Vertices and Distance : 0 2 20

Enter Vertices and Distance : 0 3 15

Enter Vertices and Distance : 1 2 5

Enter Vertices and Distance : 2 3 10

Enter Vertices and Distance : 1 4 8

Enter Vertices and Distance : 2 4 6

Enter Vertices and Distance : 3 4 9

Enter Vertices and Distance : -1 -1 -1

**Output:** (Results / Visualization):

===============================================================

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Dijkstra's Algorithm\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

===============================================================

Enter the Source : 0

===========================================================

|  |  |  |
| --- | --- | --- |
| Src->Dest Distance | Path |  |
| Railway Station->Bus Stand | 10 km | 0 1 |
| Railway Station-> Temple | 15 km | 0 1 2 |
| Railway Station->Hospital | 15 km | 0 3 |
| Railway Station->Water Tank | 18 km | 0 1 4 |

**Output**

-------------------------------------OUTPUT-------------------------------------------------

Enter Number of Cities :: 4

Enter Your Choice ::

1.Accept City Name

2.Accept Routes

3.Display

4.Shortest Distance

5.Exit

Input-->>1

Name of Cities.............

Enter Name of City : [1] :: a

Enter Name of City : [2] :: b

Enter Name of City : [3] :: c

Enter Name of City : [4] :: d

Enter Your Choice ::

1.Accept City Name

2.Accept Routes

3.Display

4.Shortest Distance

5.Exit

Input-->>2

Rent between cities [a][b] :  4

Rent between cities [a][c] :  2

Rent between cities [a][d] :  0

Rent between cities [b][c] :  0

Rent between cities [b][d] :  5

Rent between cities [c][d] :  2

Enter Your Choice ::

1.Accept City Name

2.Accept Routes

3.Display

4.Shortest Distance

5.Exit

Input-->>3

a b c d

a 0 4 2 0

b 4 0 0 5

c 2 0 0 2

d 0 5 2 0

Enter Your Choice ::

1.Accept City Name

2.Accept Routes

3.Display

4.Shortest Distance

5.Exit

Input-->>4

Enter Start Point :: a

Enter Destination :: d

Minimum Distance : 4

Shortest Path :  a c d

Shortest Distance

 a 0

a - - -

 b 4

a b - -

 c 2

a c - -

 d 4

a c d -

Enter Your Choice ::

1.Accept City Name

2.Accept Routes

3.Display

4.Shortest Distance

5.Exit

Input-->>5

**Inference:** Hence we have studied the represention of graph using adjacency matrix and find the shortest path using Dijkstra’s algorithm.