Rajdhani
DATE / /

Roll No. 23355

Title : Assignment 8 : Shortest Path finding

Aim : To implement shortest path using Dijikstras algorithm

Problem statement : Represent a graph of 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 Dijikstra's algorithm from single source to all destination.

Theory :

What is shortest path?
In graph theory, the shortest path is the path between two vertices such that the sum of the weights of its edges is minimized.
The problem of finding the shortest path in a graph from one vertex to another. Shortest can be least number of edges, least total weight etc.

various algorithm to find shortest path
1) Dijikstra's algorithm
2) Bellman-Ford algorithm
3) Floyd-Warshall algorithm
4) Johnson's algorithm
5) Viterbi algorithm

Rajdhani
DATE / /

Greedy approach:
An algorithm is designed to acheive optimum solution for a given problem. In greedy algorithm approach, decisions are made from given solution domain. As being greedy, the closedst solution that seems to provide an optimum solution is chosen.
Greedy algorithm builds up solution peice by peice, always choosing the next peice that offers the most obvious & immediate benefits.

- Dijikstra's algorithm:
It is an algorithm for finding the shortest paths between nodes in a graph.
The algorithm creates a tree of shortest path from the starting vertex (source) to all other points in graph.
Dijkstra algorithm finds a shortest path tree from a single source node by building a set of nodes that have minimum distance from source.

Real time uses of Dijkstra's algorithm:
1) Social networking applications
2) Telephone network
3) Digital mapping services in Google map
4) IP routing to find open shortest path first
5) Flighting agenda

Rajdhani
DATE / /

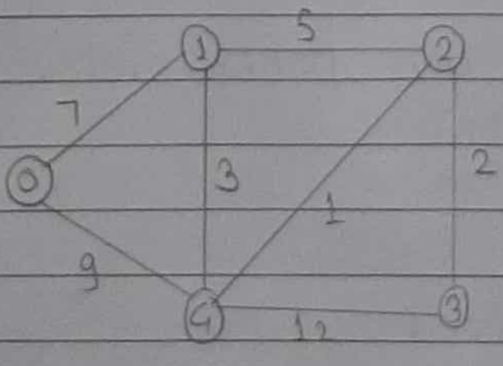
Algorithms for Dijkstra's single source to multiple destinations

```
Procedure Dijkstra
//src is the source vertex
   for i=0 to V
      // find initial distance
      If weight[src][i]!=0
           dist[i]=weight[src][i]
      else
           dist[i]=32767
      path[i]=src
      visited[i]=0
   End for
   // take source as current vertex & make it as visited
   current=src
   visited[src]=1

   // reapeate for all vertices
   for j=0 to V-2
     mindist = 32767
     // find minimum distance from current to all other
     for i=0 to V
          If visited[i]!=0 and dist[i]<mindist
               mindist=dist[i]
               current=i
     End for
     // make current as visited
     visited[current]=1
     //find shortest path from current
```

```
    for i=0 to v
      If visited [i] = 0  and (dist [current] + weight [current] [i])
                                         < dist [i])
          dist [i] = dist [current] + weight [current] [i]
          path [i] = current
    End for
End for
// display shortest path
for i=0 to v
   If i ≠ src
      print i , dist [i]
      int j=i
        do
           j=path[j]
           print path [j]
        while j≠src
End for
End
```

Example:



Rajdhani
DATE / /

Select 0 as source vertex,
From 0, 7 is minimum distance, select it

Initial:

vertex	Path	distance
0	0	0
1	0	~~∞~~ 7
2	0	∞
3	0	∞
4	0	9

Selected vertex : 1 cost : 7

vertex	Path	Distance
0	0	0
1	0	7
2	1	12
3	0	∞
4	0	9

selected vertex : 4 cost : 9

vertex	Path	distance
0	0	0
1	0	7
2	~~0~~ 4	10
3	4	21
4	0	9

selected vertex : 2
cost : 10

vertex	Path	distance
0	0	0
1	0	7
2	4	10
3	2	12
4	0	9

Shortest path :

Vertex 1 : 0 → 1 . Distance = 7
Vertex 2 : 0 → 4 → 2 Distance : 10
Vertex 3 : 0 → 4 → 2 → 3 Distance : 12
Vertex 4 : 0 → 4 Distance : 9

Rajdhani
DATE / /

Test cases:
1) Directed graph with no loops & parallel edges
2) Undirected graph with no loop & parallel edges

Validations:
Number of vertices & edges are positive

Conclusion:
Time complexity of Dijkstra algorithm is $O(V^2)$. It can be reduced to $O(E \log V)$ if graph is represented using adjacency list.