

```

/*
Computer Networks Laboratory (Lab) 15CSL77
7. Write a program for distance vector algorithm to find suitable path for
transmission
*/

/* Network layer
Router, Routing
Routing Table
Single source shortest path
All pair shortest path
Distance vector
*/

/* The Bellman-Ford algorithm solves the single-source shortest-paths problem in
the general case in which edge weights may be negative.
Where as Dijkstra Algorithm fails in this case.

Negative edge weights are found in various applications of graphs

Bellman-Ford algorithm can also detect negative cycles
*/

/* Dijkstra's algorithm uses a priority queue to greedily select the closest
vertex that has not yet been processed, and performs this relaxation
process on all of its outgoing edges;

Relaxation: select the best(minimum in this case)

By contrast, the Bellman-Ford algorithm simply relaxes all the edges, and does
this  $|V| - 1$  times, where  $|V|$  is the number of vertices in the graph
In each of these repetitions, the number of vertices with correctly calculated
distances grows, from which it follows that eventually all vertices will
have their correct distances.
*/

/* Run time - is it based on data structure used in implementation

Bellman-Ford runs in  $O(|V| \cdot |E|)$  time, where  $|V|$  and  $|E|$  are the
number of vertices and edges respectively.
 $O(|V| \cdot |E|)$  is for List based graph representation implementation
*/

/* Bellman-Ford: algorithm calculates single source shortest path

Repeat the algorithm for all n vertices to get all source shortest path:

Step 1: initialize graph
for each vertex v in vertices:
    distance[v] := inf // Initially, all vertices have a weight of infinity
    predecessor[v] := null // And a null predecessor

distance[source] := 0 // Except for the Source, where the Weight is zero

Step 2: relax edges repeatedly
for i from 1 to size(vertices)-1:
    for each edge (u, v) with weight w in edges:
        if distance[u] + w < distance[v]:
            distance[v] := distance[u] + w
            predecessor[v] := u

If the relax step is run once again and the cost from source to destination
decreases, then the graph has negative-weight cycle (Proof)
*/

/* Application
For graph with Negative edge weights
Detect negative cycles and report their existence
Applications in routing

```

```

    A distributed variant of the Bellman–Ford algorithm is used in distance-
    vector routing protocols, Routing Information Protocol (RIP)
*/

#include<stdio.h>

// Number of nodes
// Matrix representation of graph, path known or unknown
// First intermediate vertex in path from vertex u to v
// or hops or latency, depends on the network parameter
// index, to iterate through array

void readRoutingTable()
{
    // 999 to represent infinity: no direct edge, no known path between
    // vertex u and v , and 0 if its same node
    printf("\n If no direct edge between vertex u and v, or ");
    printf("if cost is unknown, then enter 999, enter 0 if its same node");
    printf("\n\n Enter the routing table : \n  |");

    // From node

    // read cost/distance

    // save if edge/path exists
}

int main()
{
    readRoutingTable();    // read network graph in terms of adjacency matrix

    // assume via, through, or intermediate node

    // If edge exists between vertex i and j, or
    // path is known

    // Relax edges repeatedly, Check if j as
    // intermediate vertex is better, ij -> jk is better than ik

    // update if i through j to k is better than existing i to k

    // update j as intermediate vertex
    // to go from i to k

    // Print router tables

    return 0;
}

/* The algorithm was first proposed by Shimbel in 1955, ( but )
   but is instead named after Richard Bellman and Lester Ford, Jr., who
   published it in 1958 and 1956, respectively.

   Edward F. Moore also published the same algorithm in 1957, and for this reason
   it is also sometimes called the Bellman–Ford–Moore algorithm.
*/

/* Like:
   GUI: Microsoft or Apple vs Xerox PARC
   X-Ray: Edison or Röntgen
       There are lot more on Edison or . .
   Boson: Bose and Einstein or just Bose
*/

/* Textbook:
   Behrouz Forouzon - Data Communications and Networking, McGraw Hill Edition
   Introduction to the design & analysis of algorithms / Anany Levitin
   Introduction to Algorithms , Thomas H. Cormen , Charles E. Leiserson ,

```

```

    Ronald L. Rivest, Clifford Stein (for Proof)
Data Structures and Algorithm Analysis in Java , Mark Allen Weiss
*/

/* Output:
    Enter number of nodes : 4

    If no direct edge between vertex u and v, or if cost not known, then enter
999, enter 0 if its same node

    Enter the routing table :
      | a b c d
-----
a | 0 5 1 4
b | 5 0 6 2
c | 1 6 0 3
d | 4 2 3 0

    Table for router a
a:: 0 via a
b:: 5 via a
c:: 1 via a
d:: 4 via a

    Table for router b
a:: 5 via b
b:: 0 via b
c:: 5 via d
d:: 2 via b

    Table for router c
a:: 1 via c
b:: 5 via d
c:: 0 via c
d:: 3 via c

    Table for router d
a:: 4 via d
b:: 2 via d
c:: 3 via d
d:: 0 via d
*/

```