**Program No.13 Implementation of distance vector (old ARPANET) routing algorithm using C**

**Aim: -** To implement distance vector routing algorithm, to compute shortest distance between each node and its neighboring nodes, and to update the distance (or cost) to reach its neighbors.

**Problem Definition:-**

A distance-vector routing protocol requires that a router informs its neighbors of topology changes / updates periodically. The term distance vector refers to the fact that protocol manipulates vectors (arrays) of distances to other nodes in the network. "Distance" is a measure of the cost to reach a certain node. This algorithm is used in Routing Information Protocol (RIP) to calculate the direction and least-cost distance (usually hop count) to any link in a network. Routers using distance-vector routing protocol do not have the knowledge of entire path to a destination. Instead they use two methods:

1. Direction in which router or exit interface a packet should be forwarded.
2. Distance from its destination

**Information kept by DV router -**

-Each router has an ID

-Associated with each link connected to a router, there is a link cost (static or dynamic).

-Intermediate hops

**Distance Vector Table Initialization -**

Distance to itself = 0

Distance to ALL other routers = infinity number or a finite cost (distance) value based on hop count.

**Distance Vector Algorithm –**

-A router transmits its distance vector to each of its neighbors in a routing packet.

-Each router receives and saves the most recently received distance vector from each of its neighbors.

-A router recalculates its distance vector when:

- It receives a distance vector from a neighbor containing different information than before.
- It discovers that a link to a neighbor has gone down.

The DV calculation is based on minimizing the cost to each destination

Dx(y) = Estimate of least cost from x to y

C(x,v) = Node x knows cost to each neighbor v

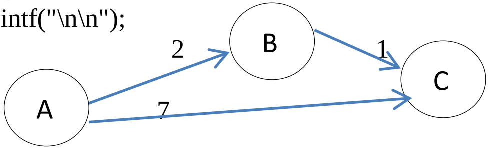
Dx = [Dx(y): y ∈ N ] = Node x maintains distance vector

Node x also maintains its neighbors' distance vectors

– For each neighbor v, x maintains Dv = [Dv(y): y ∈ N ]

**Program:-** /* Distance Vector Routing in this program is implemented using Bellman Ford Algorithm*/

```
#include<stdio.h>
struct node {
   unsigned dist[20];
   unsigned from[20];
}rt[10];
int main()
{   int costmat[20][20];
   int nodes,i,j,k,count=0;
   printf("\nEnter the number of nodes : ");
   scanf("%d",&nodes);
   printf("\nEnter the cost matrix :\n");
   for(i=0;i<nodes;i++)
   {
      for(j=0;j<nodes;j++)
      {
         scanf("%d",&costmat[i][j]);
         costmat[i][i]=0;
         rt[i].dist[j]=costmat[i][j];//initialize the distance equal to cost matrix
         rt[i].from[j]=j; // initialize the source node
      }
   }
do {
        count=0;
        for(i=0;i<nodes;i++) //We choose arbitrary vertex k and we calculate the direct distance from node i to k
using cost matrix.
        //and add the distance from k to node j
        for(j=0;j<nodes;j++)
        for(k=0;k<nodes;k++)
           if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])
           {//We calculate the minimum distance
              rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
              rt[i].from[j]=k;
              count++;
           }
      }while(count!=0);
       for(i=0;i<nodes;i++)
       {
        printf("\n\n For router %d\n",i+1);
        for(j=0;j<nodes;j++)
        {
           printf("\t\nnode %d via %d Distance %d ",j+1,rt[i].from[j]+1,rt[i].dist[j]);
        }
      }
   printf("\n\n");
}
```



Initial routing table

| Router Reachability | Router 1 | Router 2 | Router 3 |
|---|---|---|---|
| **Cost matrix** | **A** | **B** | **C** |
| **A** | 0 | 2 | 7 |
| **B** | 2 | 0 | 1 |
| **C** | 7 | 1 | 0 |

```
/*
A sample run of the program works as:-
Enter the number of nodes:
3
Enter the cost matrix:
0 2 7
2 0 1
7 1 0
For router 1
node 1 via 1 Distance 0
node 2 via 2 Distance 2
node 3 via 3 Distance 3 //It is estimated that there is a least cost route for A to reach C, via B at a cost of 3
For router 2
node 1 via 1 Distance 2
node 2 via 2 Distance 0
node 3 via 3 Distance 1
For router 3
node 1 via 1 Distance 3//It is estimated that there is a least cost route for C to reach A, via B at a cost of 3
node 2 via 2 Distance 1
node 3 via 3 Distance 0
*/
```

Updated routing table

| Router Reachability | Router 1 | Router 2 | Router 3 |
|---|---|---|---|
| **Cost matrix** | **A** | **B** | **C** |
| **A** | 0 | 2 | 3 |
| **B** | 2 | 0 | 1 |
| **C** | 3 | 1 | 0 |