

AIM : Write a program to implement distance vector routing protocol.

THEORY: A **distance-vector routing (DVR)** protocol requires that a router inform its neighbors of topology changes periodically. Historically known as the old ARPANET routing algorithm (or known as Bellman-Ford algorithm).

In Bellman Ford , each router maintains a Distance Vector table containing the distance between itself and ALL possible destination nodes. Distances, based on a chosen metric, are computed using information from the neighbors' distance vectors.

Distance Vector Algorithm –

1. A router transmits its distance vector to each of its neighbors in a routing packet.
2. Each router receives and saves the most recently received distance vector from each of its neighbors.
3. 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.

Advantages of Distance Vector routing –

- It is simpler to configure and maintain than link state routing.

Disadvantages of Distance Vector routing –

- It is slower to converge than link state.
- It is at risk from the count-to-infinity problem.
- It creates more traffic than link state since a hop count change must be propagated to all routers and processed on each router. Hop count updates take place on a periodic basis, even if there are no changes in the network topology, so bandwidth-wasting broadcasts still occur.

IMPLEMENTATION:

```
#include <bits/stdc++.h>
using namespace std;

int Bellman_Ford(int G[100][100] , int V, int E, int
edge[100][2]) {
    int i,u,v,k,distance[100],S,flag=1;

    for(i=0;i<V;i++)
        distance[i] = 1000 ;

    cout<<"\nEnter source : ";
    cin>>S;
    distance[S-1]=0;

    for(i=0;i<V-1;i++){
        for(k=0;k<E;k++){
            u = edge[k][0];
            v = edge[k][1];
            if(distance[u]+G[u][v] < distance[v])
                distance[v] = distance[u] + G[u][v];
        }
    }

    for(k=0;k<E;k++){
        u = edge[k][0];
        v = edge[k][1] ;
        if(distance[u]+G[u][v] < distance[v])
            flag = 0 ;
    }

    if(flag)
        for(i=0;i<V;i++)
            cout<<"\nDistance from source "<<S<<" to vertex
"<<i+1<<" is "<<distance[i];

    return flag;
```

```

    }

int main()
{
    int V,edge[100][2],G[100][100],i,j,k=0;

    cout<<"Enter no. of vertices: ";
    cin>>V;
    cout<<"Enter graph in matrix form:\n";
    for(i=0;i<V;i++)
        for(j=0;j<V;j++)
        {
            cin>>G[i][j];
            if(G[i][j]!=0)
                edge[k][0]=i,edge[k++][1]=j;
        }

    if(Bellman_Ford(G,V,k,edge))
        cout<<"\nNo negative weight cycle exists\n";

    return 0;
}

```

OUTPUT:

```

Enter no. of vertices: 4
Enter graph in matrix form:
0 3 5 7
9 0 3 4
2 4 0 8
6 4 3 0

Enter source : 2

Distance from source 2 to vertex 1 is 5
Distance from source 2 to vertex 2 is 0
Distance from source 2 to vertex 3 is 3
Distance from source 2 to vertex 4 is 4
No negative weight cycle exists

```

LEARNING : In this program, we learnt about the distance vector routing protocol. This protocol makes use of Bellman ford algorithm.