EXPERIMENT-11

Write a Program to implement Distance vector routing algorithm by obtaining routing table.

Distance vector routing algorithm

Distance Vector Routing (DVR) is a dynamic routing algorithm used in computer networks to find the shortest path from one router to all other routers.

Each router:

- Maintains a routing table (a "distance vector") that stores the minimum distance (cost) to reach every other node in the network.
- Periodically shares this table with its neighboring routers.
- Updates its own table based on the information received from neighbors.

Program

```
#include <stdio.h>
struct node
{
   unsigned dist[20];
   unsigned from[20];
} rt[10];
void main()
{
   int costmat[20][20];
   int nodes, i, j, k, count = 0;
   clrscr();
   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]);
            if (costmat[i][j] == 999)  // 999 = infinity (no direct link)
                  costmat[i][j] = 999;
         }
         costmat[i][i] = 0;
   }
   // Initialize routing tables
   for (i = 0; i < nodes; i++)
   {
         for (j = 0; j < nodes; j++)
         {
            rt[i].dist[j] = costmat[i][j];
            rt[i].from[j] = j;
```

```
        }
    }
     // Distance Vector Algorithm
    do
    {
        count = 0;
        for (i = 0; i < nodes; i++)
        {
           for (j = 0; j < nodes; j++)
           {
                for (k = 0; k < nodes; k++)
                {
                   if (rt[i].dist[j] > costmat[i][k] + rt[k].dist[j])
                   {
                        rt[i].dist[j] = costmat[i][k] + rt[k].dist[j];
                        rt[i].from[j] = k;
                        count++;
                   }
                }
           }
        }
    } while (count != 0);
    // Display the final routing tables
    for (i = 0; i < nodes; i++)
    {
        printf("\n\nRouting table for router %d\n", i + 1);
        printf("Destination\tNext Hop\tDistance\n");
        for (j = 0; j < nodes; j++)
        {
           printf("%d\t\t%d\t\t%d\n", j + 1, rt[i].from[j] + 1, rt[i].dist[j]);
        }
    }
     printf("\n");
    getch();
}
```

Output: