12/1/23

EXPERIMENT-10

PAGE NO :
DATE :

Implement Dijkstra's algorithm to compute the shortest path for a given topology.

```
#include <stdio.h>
#include <conio.h>
#define INFINITY 9999
#define MAX 10
    void dijkstra (int G[MAX][MAX], int n,
                   int startnode);
int main()
 {int G[MAX][MAX], i,j, n,u;
printf("Enter no of vertices");
scanf("%d", &n);
  printf("\n Enter the adjacency matrix:\n");
for (i=0; i<n; i++)
  for(j=0; j<n; j++)
    scanf("%d", &G[i][j]);
    dijkstra(G, n, u);
          return 0; }
void dijkstra (int G[MAX][MAX], int n,
               int startnode)
 {
  int cost[MAX][MAX], distance[MAX], pred[MAX];
  int visited[MAX], count, mindistance,
    nextnode, i, j;
    for (i=0; i<n; i++)
     for (j=0; j<n; j++)
       if (G[i][j]==0)
        cost[i][j] = INFINITY;
       else cost[i][j] = G[i][j];
         for (i=0; i<n; i++)
          {
```

```
distance [i] = cost [startnode][i];
pred [i] = startnode;
visited [i] = 0;
}
distance [startnode] = 0;
visited [startnode] = 1;
count = 1;
while (count < n-1)
{
mindistance = INFINITY;
for (i = 0; i < n; i++)
if (distance [i] < mindistance
&& !visited [i])
{
mindistance = distance [i];
nextnode = i;
}
for (i = 0; i < n; i++)
if (distance [i] < mindistance &&
visited [i])
{
mindistance = distance [i];
nextnode = i;
}
for (i=0; i<n; i++)
if (!visited [i])
if (mindistance + cost [nextnode][i] <
distance [i])
{
distance [i] = mindistance + cost [nextnode]
[i];
pred [i] = nextnode;
}
```

```
        count ++;
    }
    for (i=0; i<n; i++)
        if (i != startnode)
    {
        printf ("\n Distance of node %d = %d",
                i, distance[i]);
        printf ("\n Path = %d", i);
        j = i;
        do {
            j = pred[j];
            printf ("<-%d", j);
        }
        while (j != startnode);
    }
}
```

OUTPUT :

Enter no. of vertices : 4
Enter adjacency matrix :

12/1/2023.

```
0 1 1 1
1 0 1 0
1 1 0 1
1 0 1 0
```

Enter starting node : 1
Distance of 0 = 1
path = 0 <- 1
Distance of 2 = 1
path = 2 <- 1
Distance of 3 = 2
path = 3 <- 0 <- 1