| | |
|---|---|
| **Name** | **Harsh Chandra** |
| **UID no.** | **2021700013** |
| **Experiment No.** | **6** |

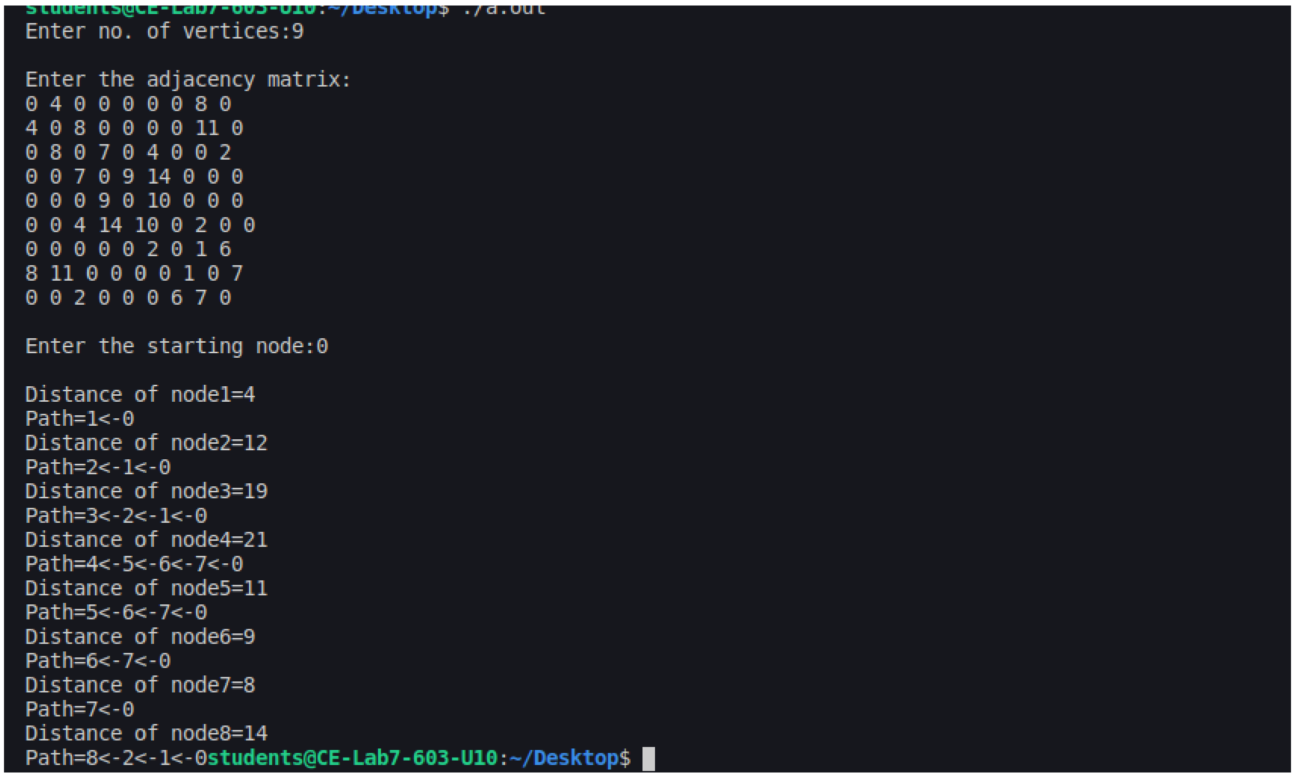
| | |
|---|---|
| **AIM:** | **Implement dijkstra's algorithm in c.** |
| **Program 1** | |
| **ALGORITHM/ THEORY:** | Create a set **sptSet** (shortest path tree set) that keeps track of vertices included in the shortest path tree, i.e., whose minimum distance from the source is calculated and finalized. Initially, this set is empty.<br>• Assign a distance value to all vertices in the input graph. Initialize all distance values as **INFINITE**. Assign the distance value as 0 for the source vertex so that it is picked first.<br>• While **sptSet** doesn't include all vertices<br>  • Pick a vertex **u** that is not there in **sptSet** and has a minimum distance value.<br>  • Include u to **sptSet**.<br>  • Then update the distance value of all adjacent vertices of u.<br>    • To update the distance values, iterate through all adjacent vertices.<br>    • For every adjacent vertex v, if the sum of the distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v. |

**PROGRAM:**

```
#include <stdio.h>

#define INFINITY 9999
#define MAX 10

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;
            }

        visited[nextnode] = 1;
        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("\nDistance of node%d=%d", i, distance[i]);
            printf("\nPath=%d", i);
            j = i;
            do
            {
                j = pred[j];
                printf("<-%d", j);
            } while (j != startnode);
        }
}

int main()
{
    int G[MAX][MAX], i, j, n, u;
    printf("Enter no. of vertices:");
    scanf("%d", &n);
    printf("\nEnter the adjacency matrix:\n");
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            scanf("%d", &G[i][j]);
    printf("\nEnter the starting node:");
    scanf("%d", &u);
    dijkstra(G, n, u);
    return 0;
}
```

**RESULT:**



```
students@CE-Lab7-603-U10:~/Desktop$ ./a.out
Enter no. of vertices:9

Enter the adjacency matrix:
0 4 0 0 0 0 0 8 0
4 0 8 0 0 0 0 11 0
0 8 0 7 0 4 0 0 2
0 0 7 0 9 14 0 0 0
0 0 0 9 0 10 0 0 0
0 0 4 14 10 0 2 0 0
0 0 0 0 0 2 0 1 6
8 11 0 0 0 0 1 0 7
0 0 2 0 0 0 6 7 0

Enter the starting node:0

Distance of node1=4
Path=1<-0
Distance of node2=12
Path=2<-1<-0
Distance of node3=19
Path=3<-2<-1<-0
Distance of node4=21
Path=4<-5<-6<-7<-0
Distance of node5=11
Path=5<-6<-7<-0
Distance of node6=9
Path=6<-7<-0
Distance of node7=8
Path=7<-0
Distance of node8=14
Path=8<-2<-1<-0students@CE-Lab7-603-U10:~/Desktop$
```

0 4 0 0 0 0 0 8 0
4 0 8 0 0 0 0 11 0
0 8 0 7 0 4 0 0 2
0 0 7 0 9 14 0 0 0

```
0 0 0 9 0 10 0 0 0
0 0 4 14 10 0 2 0 0
0 0 0 0 0 2 0 1 6
8 11 0 0 0 0 1 0 7
0 0 2 0 0 0 6 7 0
```

| | |
|---|---|
| **CONCLUSION:** | From this experiment I understood dijkstra’s algorithm to find shortest path. |