Atharva Bilonikar

Uid No: 2021700011

CSE DS D1

DAA EXP 6

Aim: Greedy Approach- Single Source Shortest path-Dijkstra's Algorithm

Algorithm:

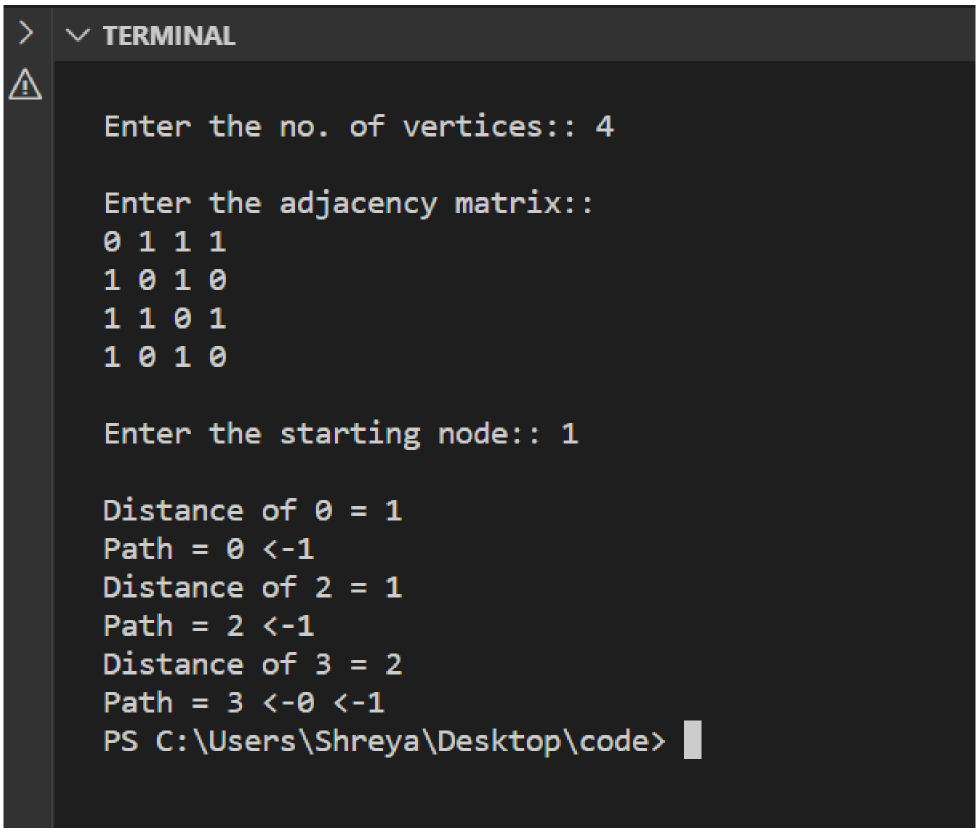
```
DIJKSTRA(G,s)
1 INITIALIZE-SINGLE-SOURCE(G, S)
2 S ← Ø
3 Q ← V[G]
4 while Q ≠ Ø
5       do u ← EXTRACT-MIN(Q)
6             S ← S U {u}
7                   for each vertex v ∈ Adj[u]
8                         do if dist[v] > dist[u] + w(u,v)
9                          then d[v] ←d[u] + w(u,v)
INITIALIZE-SINGLE-SOURCE( Graph g, Node s )
  dist[s] = 0;
  for each vertex v in Vertices V[G] - s
     dist[v] ← ∞
```

Code :

```
#include<stdio.h>
// #include<conio.h>
#define INFINITY 9999
#define MAX 10
```

```c
void dijikstra(int G[MAX][MAX], int n, int startnode);

void main(){
    int G[MAX][MAX], i, j, n, u;
    // clrscr();
    printf("\nEnter the 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);
    dijikstra(G,n,u);
    // getch();
}

void dijikstra(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 %d = %d", i, distance[i]);
            printf("\nPath = %d", i);
            j=i;
            do
            {
                j=pred[j];
                printf(" <-%d", j);
            }
            while(j!=startnode);
        }
}
```

Output:



```
Enter the no. of vertices:: 4

Enter the adjacency matrix::
0 1 1 1
1 0 1 0
1 1 0 1
1 0 1 0

Enter the 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
PS C:\Users\Shreya\Desktop\code>
```

Conclusion: In this experiment , I understood how to implement single source shortest path algorithm.