|  |  |
| --- | --- |
| **Experiment No.** | **6** |
| **Aim** | **Experiment based on graph Algorithm** |
| **Name** | **Rucha Sudhir Kulkarni** |
| **UID No.** | **2021300067** |
| **Class & Division** | **SE Computer Engineering (Div:A)(Batch:D)** |
| **Date of Performance:** | **05.04.2023** |
| **Date of Submission:** | **11.04.2023** |

**Aim: – To implement single source shortest path-Dijkstra’s algorithm.**

**Algorithm:**

1. Start
2. Input number of vertices n and the adjacency matrix adj[][]
3. Input starting vertex ,start.
4. Call dj\_algo(start);
5. Stop
6. Function dj\_algo(int start):
   1. Create new arrays distance[] , cost[][],pred[], visited[]
   2. If(adj[i][j]==0)
   3. Cost[i][j]=infinity
   4. Else cost[i][j]=ad[i][j]
   5. Initialize pred[], distance[],visited[]
   6. Distance[start]=0
   7. Visited[start]=1
   8. Count=1
   9. while(count<n-1)
   10. mindistance=INFINITY;
   11. for(i=0;i<n;i++)
   12. if(distance[i]<mindistance && !visited[i])
   13. mindistance=distance[i];
   14. nextnode=i;
   15. visited[nextnode]=1;
   16. for(i=0;i<n;i++)
   17. if(!visited[i])
   18. if(mindistance+cost[nextnode][i]<distance[i])
   19. distance[i]=mindistance+cost[nextnode][i];
   20. pred[i]=nextnode;
   21. count++;
   22. print the path and distance of each node

**Program:**

#include<stdio.h>

#define INFINITY 9999

int n;

int adj[100][100];

void dj\_algo(int start){

int cost[100][100],distance[100],pred[100];

int visited[100],count,mindistance,nextnode,i,j;

//pred[] stores the predecessor of each node

//count gives the number of nodes seen so far

//create the cost matrix

for(i=0;i<n;i++)

for(j=0;j<n;j++)

if(adj[i][j]==0)

cost[i][j]=INFINITY;

else

cost[i][j]=adj[i][j];

//initialize pred[],distance[] and visited[]

for(i=0;i<n;i++)

{

distance[i]=cost[start][i];

pred[i]=start;

visited[i]=0;

}

distance[start]=0;

visited[start]=1;

count=1;

printf("Order of visiting vertices is: %d ",start);

while(count<n-1)

{

mindistance=INFINITY;

//nextnode gives the node at minimum distance

for(i=0;i<n;i++)

if(distance[i]<mindistance && !visited[i])

{

mindistance=distance[i];

nextnode=i;

}

//check if a better path exists through nextnode

printf("%d ",nextnode);//visiting vertex

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(int i=0;i<n;i++){

if(visited[i]==0)

{

 printf("%d ",i);// showing visiting vertex

 break;

}

}

printf("\n");

//print the path and distance of each node

for(i=0;i<n;i++)

if(i!=start)

{

printf("\nDistance of node%d=%d",i,distance[i]);

printf("\nPath=%d",i);

j=i;

do

{

j=pred[j];

printf("<-%d",j);

}while(j!=start);

}

}

int main(){

printf("Enter no. of vertices ");

scanf("%d",&n);

printf("\nEnter adjacency matrix\n ");

for(int i=0;i<n;i++){

printf("Vertice %d: ",i);

for(int j=0;j<n;j++){

scanf("%d",&adj[i][j]);

}

printf("\n");

}

printf("\nThe adjacency matrix is:\n");

for(int i=0;i<n;i++){

for(int j=0;j<n;j++){

printf(" %d",adj[i][j]);

}

printf("\n");

}

printf("\nEnter starting vertex");

int start;

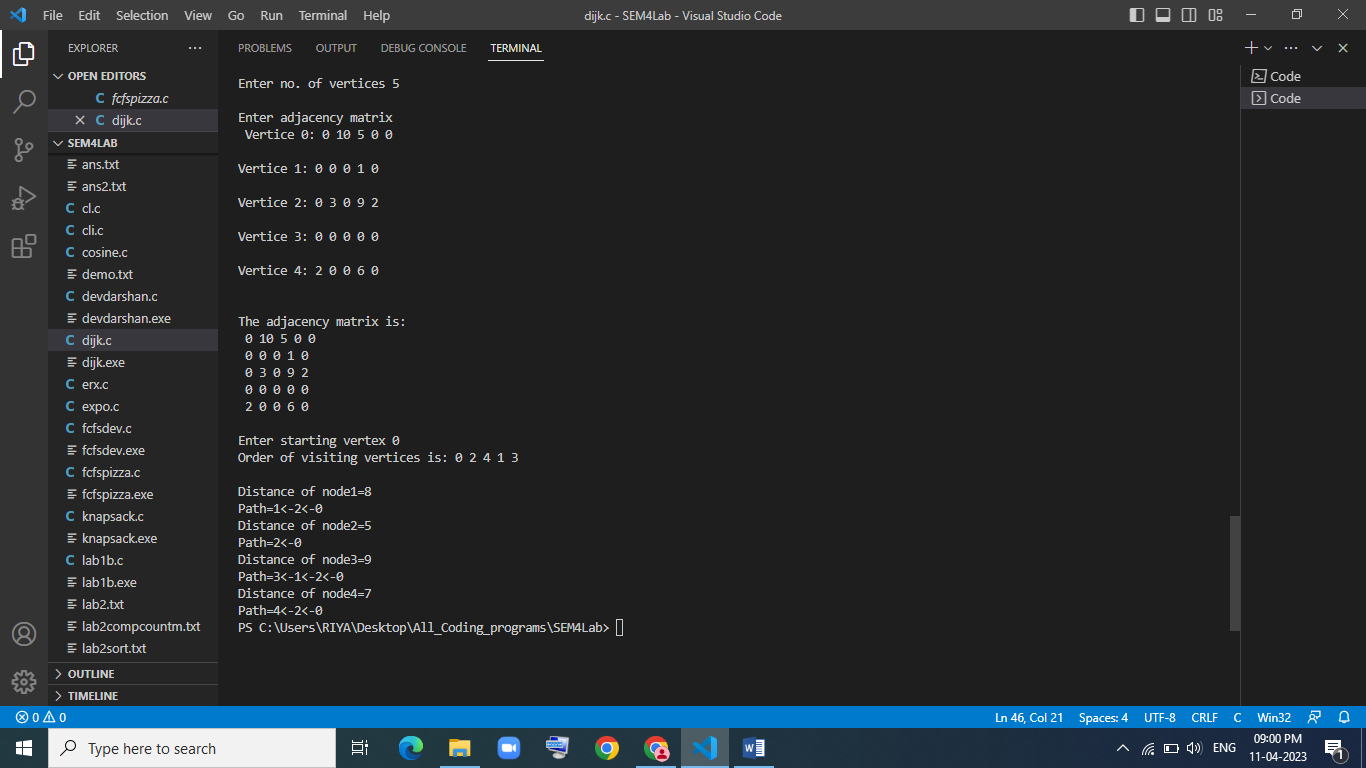
scanf("%d",&start);

dj\_algo(start);

return 0;

}

**Output and Observation:**



**Conclusion:**

After performing the above experiment, I got to know how to find shortest path of a point from a single source in a graph.