**Name: Shubhangi Shinde**

**TE-IT**

**Roll: 64**

**PROJECT ABSTRACT**

**Objective:** The objective of the project is to simulate dijkstra's algorithm using C and OpenGL. The idea is to create a graph and show a circle traversing shortest path from source to destination.

**Algorithm:** The algorithms used in this project are dijkstra’s algorithm and DDA line algorithm.

**Steps:**

1. Take the number of nodes in graph from user.
2. Enter coordinates of each of the vertices.
3. Enter cost matrix for dijkstra’s algorithm
4. Compute shortest path using dijkstra’s algorithm.
5. Create graph using OpenGL and DDA.
6. Enter source and destination.

**Code:**

#include<stdio.h>

#include<graphics.h>

#include<conio.h>

#include<math.h>

#define infinite 999

struct node{

int no;

int x;

int y;

};

void drawLine(float x1,float y1,float x2,float y2){

float x,y,dx,dy,pixel;

int i;

dx=x2-x1;

dy=y2-y1;

if(dx>=dy)

pixel=dx;

else

pixel=dy;

dx=dx/pixel;

dy=dy/pixel;

x=x1;

y=y1;

i=1;

while(i<=pixel)

{

putpixel(x,y,GREEN);

x=x+dx;

y=y+dy;

i=i+1;

}

}

void drawgraph(struct node nod[],int cost[10][10],int n){

int i,j;

for(i=1;i<=n;i++){

setcolor(RED);

circle(nod[i].x,nod[i].y,25);

}

for(i=1;i<=n;i++){

for(j=i;j<=n;j++){

if(cost[i][j]!=999){

if(nod[i].x<nod[j].x || nod[i].y<nod[j].y)

drawLine(nod[i].x,nod[i].y,nod[j].x,nod[j].y);

else

drawLine(nod[j].x,nod[j].y,nod[i].x,nod[i].y);

}

}

}

}

void startAnimation(struct node nod[],int cost[10][10],int path[],int n,int v,int d){

int gd=DETECT,gm,i,k,count,temp; float x,y,dx,dy,pixel,x1,x2,y1,y2;

int order[10],j;

temp=d;

order[0]=d;

j=1;

do{

temp=path[temp];

if(temp==0){

break;

}

order[j]=temp;

j++;

}while(temp!=0);

initgraph(&gd,&gm,"C:\\turboc3\\bgi");

for(k=j-1;k>=0;k--){

x1=nod[v].x;

y1=nod[v].y;

x2=nod[order[k]].x;

y2=nod[order[k]].y;

dx=x2-x1;

dy=y2-y1;

if(dx>=dy)

pixel=dx;

else

pixel=dy;

if(dx==0 && dy<0){

pixel=abs(dy);

}

if(dx<0 && dy==0){

pixel=abs(dx);

}

dx=dx/pixel;

dy=dy/pixel;

x=x1;

y=y1;

i=1;

while(i<=pixel)

{

drawgraph(nod,cost,n);

setcolor(YELLOW);

circle(x,y,25);

x=x+dx;

y=y+dy;

i=i+1;

delay(50);

cleardevice();

}

v=order[k];

}

drawgraph(nod,cost,n);

setcolor(YELLOW);

circle(x,y,25);

getch();

closegraph();

}

void dijiktras(int n,int v,int cost[10][10],int dist[],int path[]){

int i,u,count,w,flag[10],min;

for(i=1;i<=n;i++){

flag[i]=0;

dist[i]=cost[v][i];

//printf("\n%d",dist[i]);

}

count=2;

while(count<=n){

min=99;

for(w=1;w<=n;w++){

if(dist[w]<min && !flag[w]){

min=dist[w];

u=w;

}

}

flag[u]=1;

count++;

for(w=1;w<=n;w++){

if((dist[u]+cost[u][w]<dist[w]) && !flag[w]){

dist[w]=dist[u]+cost[u][w];

path[w]=u;

}

}

}

}

void main(){

int i,n,j,cost[10][10],dist[10],path[10],v,d;

struct node nod[10];

clrscr();

printf("Enter the number of nodes:");

scanf("%d",&n);

printf("\nEnter corrdinates for each node\n");

for(i=1;i<=n;i++){

printf("Coodinate of node %d:",i);

scanf("%d %d",&nod[i].x,&nod[i].y);

nod[i].no=i;

}

printf("\n\nEnter the cost matrix\n");

for(i=1;i<=n;i++){

for(j=1;j<=n;j++){

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

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

cost[i][j]=infinite;

}

}

printf("\nEnter Source Matrix\n");

scanf("%d",&v);

printf("\nEnter Destination Matrix\n");

scanf("%d",&d);

dijiktras(n,v,cost,dist,path);

for(i=1;i<=n;i++){

if(i!=v)

printf("%d->%d,cost=%d,path=%d\n",v,i,dist[i],path[i]);

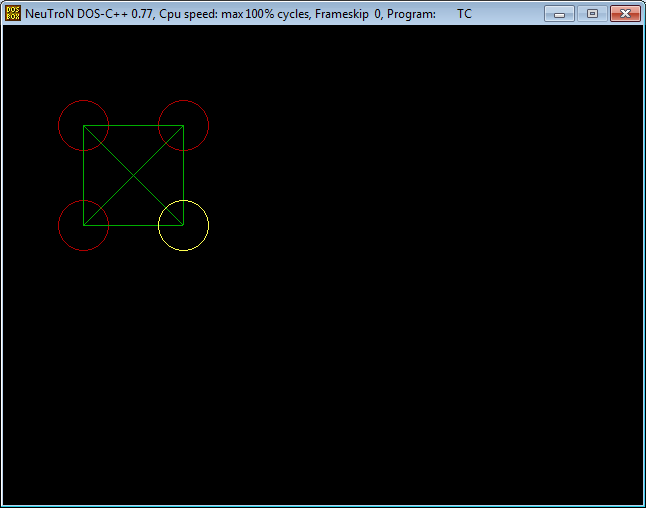
}

startAnimation(nod,cost,path,n,v,d);

getch();

}

**Output:**

****