|  |
| --- |
|  |
|  | #include <GL/glut.h> |
|  | #include <GL/gl.h> |
|  | #include<math.h> |
|  | #include<stdio.h> |
|  | #include<windows.h> |
|  | #define GL\_PI 3.14 |
|  | #define MAX 25 |
|  | #define INFINITY 999 |
|  | int n,i=1,a[25],b[25],cost[25][25],tree[25][25],src,l[2],dist[10]; |
|  | char s[20],\*s1; |
|  | void \*currentfont; |
|  |  |
|  |  |
|  | //BELLMAN VARIABLES |
|  | typedef struct { |
|  | int u, v, w; |
|  | } Edge; |
|  |  |
|  | const int NODES = 5 ; /\* the number of nodes \*/ |
|  | int EDGES=0; /\* the number of edges \*/ |
|  | Edge edges[32]; /\* large enough for n <= 2^NODES=32 \*/ |
|  | int d[32]; /\* d[i] is the minimum distance from source node s to node i \*/ |
|  |  |
|  |  |
|  |  |
|  | //FUNCTION TO SELECT BITMAP FONT |
|  | void setFont(void \*font) |
|  | { |
|  | currentfont=font; |
|  | } |
|  | //FUNCTION TO DRAW BITMAP string at (x,y) |
|  | void drawstring(GLfloat x,GLfloat y,char \*string) |
|  | { |
|  | char \*c; |
|  | glRasterPos2f(x,y); |
|  |  |
|  | for(c=string;\*c!='\0';\*c++) |
|  | { |
|  | glutBitmapCharacter(currentfont,\*c); |
|  | } |
|  | } |
|  |  |
|  | //FUNCTION TO DELAY |
|  | void delay() |
|  | { |
|  | for(int i=0;i<22000;i++) |
|  | for(int j=0;j<22000;j++); |
|  | } |
|  | //DISPLAY FUNCTION FOR TITLE PAGE |
|  | void title() |
|  | { |
|  | glLineWidth(3.0); |
|  | glColor3f(1.0,1.0,1.0); |
|  | glBegin(GL\_LINE\_LOOP); |
|  | glVertex2f(10,10); |
|  | glVertex2f(10,490); |
|  | glVertex2f(490,490); |
|  | glVertex2f(490,10); |
|  | glEnd(); |
|  |  |
|  | setFont(GLUT\_BITMAP\_HELVETICA\_18); |
|  | glColor3f(1.0,1.0,1.0); |
|  | drawstring(100,440,"Topic: Bellman Ford Algorithm"); |
|  | glColor3f(1.0,1.0,1.0); |
|  | drawstring(100,400,"Submitted by"); |
|  | glColor3f(1.0,0.0,0.0); |
|  | drawstring(100,360,"Abhishek Megotia"); |
|  | glColor3f(1.0,0.0,0.0); |
|  | drawstring(100,320,"VI CSE A"); |
|  | glColor3f(1.0,0.0,0.0); |
|  | drawstring(100,280,"1BG11CS003"); |
|  | glColor3f(1.0,1.0,1.0); |
|  | drawstring(100,100,"Right click in My Window for options"); |
|  | glFlush(); |
|  | } |
|  | //DISPLAY FUNCTION FOR INITIALIZING (DRAWING) THE INPUT AND OUTPUT AREAS |
|  | void initial() |
|  | { |
|  | glClear(GL\_COLOR\_BUFFER\_BIT); |
|  |  |
|  | setFont(GLUT\_BITMAP\_HELVETICA\_18); |
|  |  |
|  | glColor3f(0.0,0.0,0.0); |
|  | drawstring(20,230,"Input Area"); |
|  | drawstring(20,470,"Output Area"); |
|  |  |
|  | glColor3f(0.0,0.0,0.0); |
|  | glLineWidth(3.0); |
|  | glBegin(GL\_LINES); |
|  | glVertex2f(10,10); |
|  | glVertex2f(10,490); |
|  |  |
|  | glVertex2f(10,490); |
|  | glVertex2f(490,490); |
|  |  |
|  | glVertex2f(490,490); |
|  | glVertex2f(490,10); |
|  |  |
|  | glVertex2f(490,10); |
|  | glVertex2f(10,10); |
|  |  |
|  | glVertex2f(10,250); |
|  | glVertex2f(490,250); |
|  | glEnd(); |
|  |  |
|  | glFlush(); |
|  | } |
|  |  |
|  | //BLANK DISPLAY FUNCTION |
|  | void display (void) |
|  | { |
|  |  |
|  | glFlush(); |
|  |  |
|  | } |
|  |  |
|  | //DRAW A BITMAP NUMBER i at (x,y) |
|  | void raster(int x,int y,int i) |
|  | { |
|  | char z=i+'0'; |
|  | glRasterPos2f(x-5,y-5); |
|  | glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18,z); |
|  | } |
|  |  |
|  |  |
|  | //DRAW THE NODES (SQUARES) |
|  | void drawSquare(int x, int y) |
|  | { |
|  |  |
|  | if(i<=n) |
|  | { |
|  | y = 500-y; //Convert from screen coordinates |
|  | glPointSize(40); |
|  |  |
|  | if(i==src) |
|  | glColor3f(0.7f, 0.4f, 0.0f); |
|  | else |
|  | glColor3f(0.5f, 0.5f, 0.8f); |
|  |  |
|  | glBegin(GL\_POINTS); |
|  | glVertex2f(x , y); |
|  | glEnd(); |
|  |  |
|  | a[i]=x; |
|  | b[i]=y; |
|  |  |
|  | glColor3f(0.0f, 1.0f, 0.0f); |
|  | s1=itoa(i,s,10); |
|  | drawstring(x-5,y-5,s1); |
|  |  |
|  | glFlush(); |
|  | } |
|  | i=i+1; |
|  | } |
|  |  |
|  | //READ DATA: |V|,COST MATRIX, SOURCE VERTEX |
|  | void read() |
|  | { |
|  | printf("Enter the number of vertices\n"); |
|  | scanf("%d",&n); |
|  | printf("Enter the cost matrix\n"); |
|  | for(int j=1;j<=n;j++) |
|  | for(int k=1;k<=n;k++) |
|  | { |
|  | scanf("%d",&cost[j][k]); |
|  | if(cost[j][k]==0 || cost[j][k]==999) |
|  | cost[j][k]=999; |
|  | else |
|  | { |
|  | edges[EDGES].u=j; |
|  | edges[EDGES].v=k; |
|  | edges[EDGES].w=cost[j][k]; |
|  | EDGES++; |
|  | } |
|  | } |
|  | printf("\nGO TO MY WINDOW, PLACE THE NODES IN INPUT AREA AND THEN CLICK RIGHT BUTTON FOR NEXT OPTION\n"); |
|  | initial(); //Draw the initial screen |
|  | } |
|  |  |
|  | //DRAW THE EDGES |
|  | void drawline() |
|  | { |
|  | int j,k,x1,x2,y1,y2; |
|  | for(j=1;j<=n;j++) |
|  | { |
|  | for(k=1;k<=n;k++) |
|  | { |
|  | if(cost[j][k]!=999 && j<k) |
|  | { |
|  | x1=a[j]; |
|  | y1=b[j]; |
|  | x2=a[k]; |
|  | y2=b[k]; |
|  |  |
|  |  |
|  | glColor3f(0.0,0.5,0.0); |
|  |  |
|  | glLineWidth(3); |
|  | glBegin(GL\_LINES); |
|  | glVertex2i(x1-7,y1+10); |
|  | glVertex2i(x2-7,y2+10); |
|  | glEnd(); |
|  |  |
|  | s1=itoa(cost[j][k],s,10); |
|  | drawstring((x1+x2-16)/2,(y1+y2+22)/2,s1); |
|  | glFlush(); |
|  | } |
|  |  |
|  | if(cost[j][k]!=cost[k][j] && cost[j][k]!=999 && j>k) |
|  | { |
|  | x1=a[j]; |
|  | y1=b[j]; |
|  | x2=a[k]; |
|  | y2=b[k]; |
|  |  |
|  | glColor3f(1.0,0.5,0.0); |
|  | glBegin(GL\_LINES); |
|  | glVertex2i(x1+10,y1+18); |
|  | glVertex2i(x2+10,y2+18); |
|  | glEnd(); |
|  |  |
|  | s1=itoa(cost[j][k],s,10); |
|  | drawstring((x1+x2+20)/2,(y1+y2+36)/2,s1); |
|  | glFlush(); |
|  | } |
|  | } |
|  | } |
|  | } |
|  |  |
|  |  |
|  | void shortestpath(int src) |
|  | { |
|  |  |
|  | //START OF BELLMAN FORD |
|  | int j,p,q,x1,y1,x2,y2,x,y; |
|  | int d[MAX],parent[MAX]; |
|  |  |
|  | int it,flag=0,child[MAX]; |
|  |  |
|  | //INITIALIZE DATA OBJECTS |
|  | for (it = 1; it <= n; ++it) |
|  | { |
|  | d[it] = INFINITY; |
|  | parent[it]=src; |
|  |  |
|  | } |
|  | d[src] = 0; |
|  |  |
|  | //RELAXATION METHOD |
|  | for(int m=0;m<n;m++)//REPEAT N TIMES |
|  | { |
|  | //RELAX ALL EDGES |
|  | for (it = 1; it <=n; ++it) { |
|  | for (j = 1; j <=n; ++j) { |
|  |  |
|  | if(d[it]+cost[it][j]<d[j]) |
|  | { |
|  |  |
|  | d[j]=d[it]+cost[it][j]; |
|  | parent[j]=it; |
|  | } |
|  | } |
|  | } |
|  | } |
|  |  |
|  | //CHECK FOR NEGATIVE LOOPS |
|  | for (it = 1; it <=n; ++it) { |
|  | for (j = 1; j <=n; ++j) { |
|  | if(cost[it][j]==INFINITY) continue; |
|  | if(d[it]+cost[it][j]<d[j]) |
|  | { |
|  | printf("\n\nGraph contains a negative-weight cycle\n"); |
|  |  |
|  |  |
|  | return; |
|  | } |
|  | } |
|  | } |
|  | printf("From source %d\n",src); |
|  | for(i=1;i<=n;i++) |
|  | if(i!=src) |
|  | printf("The shortest distance to %d is %d\n",i,d[i]); |
|  | printf("\n"); |
|  | //INITIALIZE SPANNING TREE EDGES |
|  | int l=0; |
|  | for (int it = 1; it <= n; ++it) { |
|  | if(parent[it]==it) continue; |
|  | tree[l][1]=parent[it]; |
|  | tree[l++][2]=it; |
|  | } |
|  |  |
|  | //DRAW THE SPANNING TREE |
|  | for(int r=1;r<=n;r++) |
|  | { |
|  |  |
|  | x=a[r]; |
|  | y=b[r]; |
|  |  |
|  | glPointSize(25); |
|  | if(r==src) |
|  | glColor3f(0.7f, 0.4f, 0.0f); |
|  | else |
|  | glColor3f(0.5f, 0.5f, 0.8f); |
|  |  |
|  | glBegin(GL\_POINTS); |
|  | glVertex2f(x,y+250); |
|  | glEnd(); |
|  |  |
|  | glColor3f(0.0,1.0,0.0); |
|  |  |
|  | s1=itoa(r,s,10); |
|  | drawstring(x,y+250,s1); |
|  |  |
|  | glFlush(); |
|  |  |
|  | } |
|  |  |
|  | for(int x=0;x<l;x++) |
|  | { |
|  | p=tree[x][1]; |
|  | q=tree[x][2]; |
|  | if(p==0||q==0) continue; |
|  |  |
|  | x1=a[p]; |
|  | y1=b[p]; |
|  | x2=a[q]; |
|  | y2=b[q]; |
|  |  |
|  | if(p<q) |
|  | { |
|  | glColor3f(0.0,0.5,0.0); |
|  | glBegin(GL\_LINES); |
|  | glVertex2i(x1,y1+250); |
|  | glVertex2i(x2,y2+250); |
|  | glEnd(); |
|  |  |
|  | s1=itoa(cost[p][q],s,10); |
|  | drawstring((x1+x2)/2,(y1+y2+500)/2,s1); |
|  | } |
|  |  |
|  | else |
|  | { |
|  | glColor3f(1.0,0.5,0.0); |
|  | glBegin(GL\_LINES); |
|  | glVertex2i(x1,y1+250); |
|  | glVertex2i(x2,y2+250); |
|  | glEnd(); |
|  |  |
|  | s1=itoa(cost[p][q],s,10); |
|  | drawstring((x1+x2)/2,(y1+y2+500)/2,s1); |
|  | } |
|  | } |
|  | glFlush(); |
|  |  |
|  | } |
|  |  |
|  |  |
|  | void mouse(int bin, int state , int x , int y) |
|  | { |
|  | if(bin==GLUT\_LEFT\_BUTTON&&state==GLUT\_DOWN) |
|  | drawSquare(x,y); |
|  |  |
|  | } |
|  |  |
|  | void top\_menu(int option) |
|  | { |
|  | int x,y; |
|  | switch(option) |
|  | { |
|  | case 1: |
|  | read(); |
|  | glutPostRedisplay(); |
|  | break; |
|  | case 2: |
|  | drawline(); |
|  | glutPostRedisplay(); |
|  | break; |
|  | case 3: |
|  | for(int i=1; i<=n; i++) |
|  | { |
|  | glClear(GL\_COLOR\_BUFFER\_BIT); |
|  | initial(); |
|  | for(int r=1;r<=n;r++) |
|  | { |
|  | x=a[r]; |
|  | y=b[r]; |
|  | glPointSize(40); |
|  | if(r==src) |
|  | glColor3f(0.7f, 0.4f, 0.0f); |
|  | else |
|  | glColor3f(0.5f, 0.5f, 0.8f); |
|  | glBegin(GL\_POINTS); |
|  | glVertex2f(x,y); |
|  | glEnd(); |
|  | glColor3f(0.0,1.0,0.0); |
|  | s1=itoa(r,s,10); |
|  | drawstring(x-5,y-5,s1); |
|  | setFont(GLUT\_BITMAP\_HELVETICA\_18); |
|  | glColor3f(0.0,0.0,0.0); |
|  | drawstring(130,470,"For source"); |
|  | glFlush(); |
|  | } |
|  | drawline(); |
|  | s1=itoa(i,s,10); |
|  | setFont(GLUT\_BITMAP\_HELVETICA\_18); |
|  | glColor3f(0.0,0.0,0.0); |
|  | drawstring(225,470,s1); |
|  | glutPostRedisplay(); |
|  | shortestpath(i); |
|  | delay(); |
|  | } |
|  | break; |
|  | case 4: |
|  | exit(0); |
|  | } |
|  | } |
|  |  |
|  |  |
|  | void init (void) |
|  | { |
|  | glClearColor (1.0, 1.0, 1.0, 1.0); |
|  | glClear(GL\_COLOR\_BUFFER\_BIT); |
|  | glViewport( 0,0, 500, 500 ); |
|  | glMatrixMode( GL\_PROJECTION ); |
|  | glOrtho( 0.0, 500.0, 0.0, 500.0, 1.0, -1.0 ); |
|  | glMatrixMode(GL\_MODELVIEW); |
|  | glLoadIdentity(); |
|  | glFlush(); |
|  | } |
|  | void myInit1() |
|  | { |
|  | glClearColor(0.0,0.0,0.0,0.0); |
|  | glColor3f(0.0f,0.0f,0.0f); |
|  | glPointSize(5.0); |
|  | gluOrtho2D(0.0,500.0,0.0,500.0); |
|  | glMatrixMode(GL\_PROJECTION); |
|  | glLoadIdentity(); |
|  | setFont(GLUT\_BITMAP\_HELVETICA\_18); |
|  | } |
|  |  |
|  |  |
|  | void display1(void) |
|  | { |
|  | glClear(GL\_COLOR\_BUFFER\_BIT); |
|  | title(); |
|  |  |
|  | } |
|  | int main (int argc,char\*\* argv) |
|  | { |
|  | glutInit(&argc,argv); |
|  |  |
|  | glutInitDisplayMode(GLUT\_SINGLE|GLUT\_RGB); |
|  | glutInitWindowPosition(520,0); |
|  | glutInitWindowSize(500,500); |
|  | glutCreateWindow("Front Sheet"); |
|  | glutDisplayFunc(display1); |
|  | myInit1(); |
|  |  |
|  | glutInitDisplayMode( GLUT\_SINGLE|GLUT\_RGB ); |
|  | glutInitWindowSize(500,500); |
|  | glutInitWindowPosition(0,0); |
|  | glutCreateWindow("My Window"); |
|  | glutDisplayFunc(display); |
|  | glutMouseFunc(mouse); |
|  | glutCreateMenu(top\_menu); |
|  | glutAddMenuEntry("Read Cost Matrix",1); |
|  | glutAddMenuEntry("Display Weighted Graph",2); |
|  | glutAddMenuEntry("Display Shortest Path",3); |
|  | glutAddMenuEntry("Exit",4); |
|  | glutAttachMenu(GLUT\_RIGHT\_BUTTON); |
|  | printf("\nGO TO MY WINDOW AND CLICK RIGHT BUTTON FOR NEXT OPTION\n"); |
|  | init(); |
|  |  |
|  | glutMainLoop(); |
|  | } |