Fundamentals of Algorithm Project

Escape The Demon

B-10

Submitted By :

Sakshi Dubey 13103716

Subham Gupta 13103456

Anand Tiwari 13103445

Sanket Chaturvedi 13103458

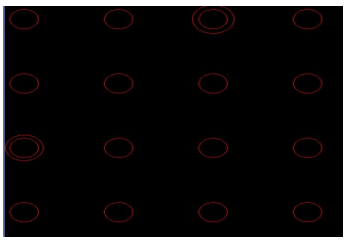
Problem Statement : The game revolves around the two key players ,one of them is controlled by the user while other is a devil who is in hunt of eating the player. The user controls the player. However a monster always keeps following him wherever he goes. When one level is complete, player is taken to the next stage. Enemy roams around the maze, trying to catch player, if an enemy touches player or technically if their co-ordinates coincides , a life is lost and player itself withers and dies. The game ends if player successfully reaches the desired location without being eaten my monster.

Gobble up all the places to get to the next level in this fun and addicting game. You lose a life every time they catch you. Avoid getting trapped in a corner because your dangerous little opponents are clever and will attack you there!

How to Play: Use the keys as follows-
A- Left
D- Right
W-Up
S- Down

Description:

The game roams about a matrix sort of maze



0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Level 1:
The player starts at 12 and is supposed to reach point 4 for completion of the level.
Level 2:
The player starts at 12 and is supposed to reach point 7 for completion of the level.
Level 3:
The player starts at 12 and is supposed to reach point 11 for completion of the level.

Algorithm Used:

Demon Movement: The demon has kept an cryptic eye on the movement of the player. The movement of the devil is governed by the "Floyd Warshall" algorithm. The **Floyd–Warshall algorithm** is an algorithm for finding shotest paths in a weighted graph with edge weights. A single execution of the algorithm will find the lengths of the shortest paths between all pairs of vertices.
Here with every movement of the player demon with the help of Floyd Warshall Algorithm finds out all the shortest possible path between him and the user and follows it to end the game.

Data Structure Used:

Data Structures which are used in the project are adjacency matrix where we have made a matrix and we are calculating the shortest path. Next we have used graphics which is the main soul of the program.

Project Code:

```
#include<stdio.h>

#define V 16

#define INF 9999

#include <conio.h>

#include <graphics.h>

#include <dos.h>
```

```
#include<stdlib.h>

#define MAX 20



int path[100][100],pathList[100];

void printSolution(int dist[][V]);

int l=0,m=0;

int a=20,b=120,c=220,d=320,ch,newx=20,newy=320,k=0,dnewx=320,dnewy=20;

int
x1=15,x2=15,x3=15,x4=15,x5=15,x6=15,x7=15,x8=15,x9=15,x10=15,x11=15,x12=15,x13=15,x14=15,x
15=15,x16=15;

int source=12,dest=3;

int adj[MAX][MAX];

int n=16;

void setsource();

void graphics()

{

int i;

setbkcolor(GREEN);

setfillstyle(SOLID_FILL,WHITE);

bar(120,80,500,380);

line(120,80,70,50);

line(70,50,70,420);

line(70,420,120,380);

line(500,80,550,50);

line(550,50,550,420);

line(550,420,500,380);

circle(100,20,10);

circle(110,50,10);
```

```
circle(50,90,10);
circle(30,300,20);
circle(40,200,15);
circle(200,25,25);
circle(290,35,15);
circle(350,25,20);
circle(410,30,10);
circle(470,35,15);
circle(605,25,25);
circle(610,80,20);
circle(580,140,10);
circle(590,180,20);
circle(600,300,25);
circle(610,380,15);
circle(400,420,25);
circle(300,410,20);
circle(180,405,10);
circle(120,425,26);
circle(530,445,25);
circle(490,420,15);
settextstyle(1,0,3);
setcolor(BLACK);
outtextxy(130,90,"SAKSHI DUBEY (13103716)");
settextstyle(1,0,3);
setcolor(BLACK);
outtextxy(130,140,"SUBHAM GUPTA (13103456)");
settextstyle(1,0,3);
setcolor(BLACK);
```

```
outtextxy(130,190,"ANAND TIWARI (13103445)");
settextstyle(1,0,3);
setcolor(BLACK);
outtextxy(130,240,"SANKET CHATURVEDI(13103458)");
settextstyle(1,0,4);
setcolor(BLACK);
outtextxy(260,300,"B-10");
settextstyle(1,0,4);
getch();
cleardevice();




setbkcolor(BLACK);
setcolor(YELLOW);
setfillstyle(SOLID_FILL,YELLOW);
for(i=0;i<250;i++)
{
setfillstyle(SOLID_FILL,YELLOW);
setcolor(YELLOW);
circle(70,50+i,30);
floodfill(70,50+i,YELLOW);

setcolor(RED);
setfillstyle(SOLID_FILL,RED);
circle(70,100+i,5);
floodfill(70,100+i,RED);
```

```
setcolor(GREEN);
setfillstyle(SOLID_FILL,GREEN);
circle(70,130+i,5);
floodfill(70,130+i,GREEN);

setcolor(BLUE);
setfillstyle(SOLID_FILL,BLUE);
circle(70,160+i,5);
floodfill(70,160+i,BLUE);

delay(2);
clrscr();
}

for(i=0;i<400;i++)
{
setfillstyle(SOLID_FILL,YELLOW);
setcolor(YELLOW);
circle(70+i,300,30);
floodfill(70+i,300,YELLOW);

setcolor(RED);
setfillstyle(SOLID_FILL,RED);
circle(120+i,300,5);
floodfill(120+i,300,RED);

setcolor(GREEN);
setfillstyle(SOLID_FILL,GREEN);
```

```
circle(150+i,300,5);
floodfill(150+i,300,GREEN);

setcolor(BLUE);
setfillstyle(SOLID_FILL,BLUE);
circle(180+i,300,5);
floodfill(180+i,300,BLUE);
delay(2);
clrscr();
}

setfillstyle(SOLID_FILL,YELLOW);
setcolor(YELLOW);
circle(470,300,30);
floodfill(470,300,YELLOW);

setcolor(RED);
setfillstyle(SOLID_FILL,RED);
circle(520,300,5);
floodfill(520+i,300,RED);

setcolor(GREEN);
setfillstyle(SOLID_FILL,GREEN);
circle(550,300,5);
floodfill(550,300,GREEN);

setcolor(BLUE);
setfillstyle(SOLID_FILL,BLUE);
```

```
circle(580,300,5);
floodfill(580,300,BLUE);

circle(470,300,30);
floodfill(470,300,YELLOW);
settextstyle(4,0,9);
setcolor(RED);
outtextxy(120,50,"ESCAPE");
settextstyle(1,0,4);
getch();
setbkcolor(BLACK);
cleardevice();
}



  void direction()
 {


        circle(newx,newy,20);
        if(ch==97)
        {

        cleardevice();
        if(a+l-100<350&&a+l-100>10)
        {
        l=l-100;
```

```
        newx=a+l;



        circle(newx,newy,20);
        circle(a,a,x1);
        circle(a,b,x2);
        circle(a,c,x3);
        circle(a,d,x4);
        circle(b,a,x5);
        circle(b,b,x6);
        circle(b,c,x7);
        circle(b,d,x8);
        circle(c,a,x9);
        circle(c,b,x10);
        circle(c,c,x11);
        circle(c,d,x12);
        circle(d,a,x13);
        circle(d,b,x14);
        circle(d,c,x15);
        circle(d,d,x16);
        setsource();


         }



}
```

```
        else if(ch==100)
        {

                cleardevice();
                if(a+l+100<350&&a+l+100>10)
                {
                l=l+100;
                newx=a+l;

                circle(newx,newy,20);
                circle(a,a,15);
circle(a,a,x1);
circle(a,b,x2);
circle(a,c,x3);
circle(a,d,x4);
circle(b,a,x5);
circle(b,b,x6);
circle(b,c,x7);
circle(b,d,x8);
circle(c,a,x9);
circle(c,b,x10);
circle(c,c,x11);
circle(c,d,x12);
circle(d,a,x13);
circle(d,b,x14);
circle(d,c,x15);
circle(d,d,x16);
```

```
setsource();

        }

    }
        else if(ch==119)
        {

            cleardevice();
            if(d+k-100<350&&d+k-100>10)
            {
                    k=k-100;
                    newy=d+k;

                    circle(newx,newy,20);
                    circle(a,a,15);
                    circle(a,a,x1);
circle(a,b,x2);
circle(a,c,x3);
circle(a,d,x4);
circle(b,a,x5);
circle(b,b,x6);
circle(b,c,x7);
circle(b,d,x8);
circle(c,a,x9);
circle(c,b,x10);
circle(c,c,x11);
circle(c,d,x12);
```

```
circle(d,a,x13);
circle(d,b,x14);
circle(d,c,x15);
circle(d,d,x16);

    setsource();
          }

     }
          else if(ch==115)
          {

                    cleardevice();
                    if(d+k+100<350&&d+k+100>10)
                    {
                    k=k+100;
                    newy=d+k;

                         circle(newx,newy,20);
                         circle(a,a,x1);
circle(a,b,x2);
circle(a,c,x3);
circle(a,d,x4);
circle(b,a,x5);
circle(b,b,x6);
circle(b,c,x7);
circle(b,d,x8);
circle(c,a,x9);
```

```
circle(c,b,x10);

circle(c,c,x11);

circle(c,d,x12);

circle(d,a,x13);

circle(d,b,x14);

circle(d,c,x15);

circle(d,d,x16);



setsource();


  }


            }


circle(newx,newy,20);

circle(a,a,x1);

circle(a,b,x2);

circle(a,c,x3);

circle(a,d,x4);

circle(b,a,x5);

circle(b,b,x6);

circle(b,c,x7);

circle(b,d,x8);

circle(c,a,x9);

circle(c,b,x10);

circle(c,c,x11);

circle(c,d,x12);
```

```
        circle(d,a,x13);
        circle(d,b,x14);
        circle(d,c,x15);
        circle(d,d,x16);
        setsource();


}


void check_kbhit()
{

        ch=getch();
        if(ch==27)
        {
                exit(0);
        }
        else
        {
                direction();

        }

}

 void setsource()
 {
```

```
        if(newx==20 && newy==20)
        source=0;
        if(newx==20 && newy==120)
        source=4;
    if(newx==20 && newy==220)
        source=8;
    if(newx==20 && newy==320)
        source=12;
    if(newx==120 && newy==20)
        source=1;
    if(newx==120 && newy==120)
        source=5;
    if(newx==120 && newy==220)
        source=9;
    if(newx==120 && newy==320)
        source=13;
    if(newx==220 && newy==20)
        source=2;
    if(newx==220 && newy==120)
        source=6;
    if(newx==220 && newy==220)
        source=10;
     if(newx==220 && newy==320)
        source=14;


    if(newx==320 && newy==20)
        source=3;
    if(newx==320 && newy==120)
```

```
    source=7;
if(newx==320 && newy==220)
    source=11;
if(newx==320 && newy==320)
   source=15;
 }



void getdest(int source)
 {
    if(source==0)
    {
    dnewx=20;
    dnewy=20;
    }
        if(source==4)
    {
    dnewx=20;
    dnewy=120;
    }
        if(source==8)
    {
    dnewx=20;
    dnewy=220;
    }

        if(source==12)
```

```
{
dnewx=20;
dnewy=320;
}

    if(source==1)
{
dnewx=120;
dnewy=20;
}

    if(source==5)
{
dnewx=120;
dnewy=120;
}

    if(source==9)
{
dnewx=120;
dnewy=220;
}

    if(source==13)
{
dnewx=120;
dnewy=320;
}
```

```
                if(source==2)
{
dnewx=220;
dnewy=20;
}

    if(source==6)
{
dnewx=220;
dnewy=120;
}

    if(source==10)
{
dnewx=220;
dnewy=220;
}

    if(source==14)
{
dnewx=220;
dnewy=320;
}

                if(source==3)
{
```

```
dnewx=320;
dnewy=20;
}

        if(source==7)
{
dnewx=320;
dnewy=120;
}

    if(source==11)
{
dnewx=320;
dnewy=320;
}

    if(source==15)
{
dnewx=320;
dnewy=320;
}

}

void drawdemon(int val)
{
```

```
   getdest(val);
   circle(dnewx,dnewy,22);




}




void floydWarshell (int graph[16][16])
{
   int dist[16][16];
   int i, j, k,l;



    for (i = 0; i < V; i++) {
           for (j = 0; j < V; j++) {
              path[i][j] = -1;
           }
   }

   for (i = 0; i < V; i++)
   {

           for (j = 0; j < V; j++)
           {
```

```
            dist[i][j] = graph[i][j];
        }

    }


for(l=0;l<2;l++)

{
 for (i= 0; i < V; i++)
   {
        for (k = 0; k < V; k++)
        {
            for (j = 0; j < V; j++)
            {
                if (dist[i][k] + dist[k][j] < dist[i][j])
                {
                    dist[i][j] = dist[i][k] + dist[k][j];
                   path[i][j]=k;

                }

            }
        }
   }

}
  // printSolution(dist);
```

```
}



void printSolution(int dist[][V])
{

   int i,j;
   printf("Following matrix shows the shortest distances\n");
   for ( i =0; i <V; i++)
   {
         for ( j =0; j <V; j++)
         {
               if(dist[i][j]==INF)
               printf("%s", "INF");
            else

               printf ("%5d",dist[i][j]);
         }
         printf("\n");
   }
}



int getPath(int i, int j) {
         int k;
         if(path[i][j] == -1) {
               pathList[m++] = i;
            //pathList[l++] = j;
```

```
                return 0;
        }
        k = path[i][j];
        getPath(i,k);
        getPath(k,j);
        return 0;
}

void imp(int source,int d)
 {
    int i;
    if(source==dest)
    {
    cleardevice();
    settextstyle(1,HORIZ_DIR,3);
    outtextxy(200,200,"sorry you lost!!!");
    delay(2000);
    exit(1);

    }
    else
    {
        getPath(source,d);

         d=pathList[m-1];

         drawdemon(d);
         dest=d;
```

```
    }


}



int main() {
    int i,j;

    int gdriver = DETECT, gmode, err,mn;
        int graph[16][16]={
                        {0,1,INF,INF,1,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF},
                        {1,0,1,INF,INF,1,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF},
                        {INF,1,0,1,INF,INF,1,INF,INF,INF,INF,INF,INF,INF,INF,INF},
                        {INF,INF,1,0,INF,INF,INF,1,INF,INF,INF,INF,INF,INF,INF,INF},
                        {1,INF,INF,INF,0,1,INF,INF,1,INF,INF,INF,INF,INF,INF,INF},
                        {INF,1,INF,INF,1,0,1,INF,INF,1,INF,INF,INF,INF,INF,INF},
                        {INF,INF,1,INF,INF,1,0,1,INF,INF,1,INF,INF,INF,INF,INF},
                        {INF,INF,INF,1,INF,INF,1,0,INF,INF,INF,1,INF,INF,INF,INF},
                        {INF,INF,INF,INF,1,INF,INF,INF,0,1,INF,INF,1,INF,INF,INF},
                        {INF,INF,INF,INF,INF,1,INF,INF,INF,0,1,INF,INF,1,INF,INF},
                        {INF,INF,INF,INF,INF,INF,1,INF,INF,1,0,1,INF,INF,1,INF},
                        {INF,INF,INF,INF,INF,INF,INF,1,INF,INF,1,0,INF,INF,INF,1},
                        {INF,INF,INF,INF,INF,INF,INF,INF,1,INF,INF,INF,0,1,INF,INF},
                        {INF,INF,INF,INF,INF,INF,INF,INF,INF,1,INF,INF,1,0,1,INF},
                        {INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,1,INF,INF,1,0,1},
```

```
                    {INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,INF,1,INF,INF,1,0}


                    };


                    clrscr();
    initgraph(&gdriver, &gmode, "C:/TURBOC3/BGI");
    settextstyle(0,0,0);
    cleardevice();
    setcolor(WHITE);
    setfillstyle(BKSLASH_FILL, WHITE);
            setcolor(YELLOW);
    setfillstyle(SOLID_FILL, YELLOW);
graphics();
    circle(a,a,15);
    circle(a,b,15);
    circle(a,c,15);
    circle(a,d,15);
    circle(b,a,15);
    circle(b,b,15);
    circle(b,c,15);
    circle(b,d,15);
    circle(c,a,15);
    circle(c,b,15);
    circle(c,c,15);
    circle(c,d,15);
    circle(d,a,15);
    circle(d,b,15);
    circle(d,c,15);
```

```
circle(d,d,15);
floydWarshell(graph);



do{
        do
        {

                if(kbhit())

                {

                        check_kbhit();
                        setsource();

                        imp(source,dest);

                }







        }while(1);


  }while(1);
```

```
return 0;

}
```