**Exercise 13: Implementation of Shortest Path Algorithm**

**adt.h:**

#include<stdio.h>

struct graph

{

int dist[10][10],pred[10][10],adj[10][10];

int vtx;

int edge;

};

void create(struct graph \*G,int v,int e,int d);

void disp(struct graph \*G);

void floyd(struct graph \*G);

void path(struct graph \*G,int i,int j);

**impl.h:**

#include "adt.h"

void create(struct graph \*G,int v,int e,int d)

{

G->vtx=v;

G->edge=e;

//init(&G->Q,20);

for(int i=0;i<v;i++)

{

for(int j=0;j<v;j++)

{

G->dist[i][j]=9999;

G->pred[i][j]=-1;

}

G->dist[i][i]=0;

}

//vQ(G);

char edge[2];

//if(d)

for(int i=0;i<e;i++)

{

printf("Enter the edge: ");

scanf("%s",edge);

int dist;

printf("Enter the Dist: ");

scanf("%d",&dist);

G->dist[edge[0]-'1'][edge[1]-'1']=dist;

G->adj[edge[0]-'1'][edge[1]-'1']=1;

if(!d)

G->dist[edge[1]-'1'][edge[0]-'1']=dist;

}

/\*G->dist[0][1]=4;

G->dist[1][2]=2;

G->dist[0][2]=7;

G->dist[2][0]=6;

G->dist[1][0]=1;\*/

//G->dist[0][1]=4;

//G->dist[0][1]=4;

/\*G->dist[0][1]=4;

G->dist[1][2]=5;

G->dist[2][1]=1;

G->dist[1][0]=1;

G->dist[1][3]=2;

G->dist[3][1]=8;

G->dist[3][2]=1;

G->dist[0][3]=3;\*/

}

void disp(struct graph \*G)

{

printf("\nDist:\n");

for(int i=0;i<G->vtx;i++)

{

for(int j=0;j<G->vtx;j++)

{

printf("%d ",G->dist[i][j]);

//printf("%d ",G->pred[i][j]);

}

printf("\n");

}

printf("\n");

printf("\nPred:\n");

for(int i=0;i<G->vtx;i++)

{

for(int j=0;j<G->vtx;j++)

{

//printf("%d ",G->dist[i][j]);

printf("%d ",G->pred[i][j]+1);

}

printf("\n");

}

printf("\n\n");

}

void floyd(struct graph \*G)

{

int n=G->vtx;

for(int k=0;k<n;k++)

{

printf("\nWhen k=%d\n",k+1);

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

{

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

{

if(G->dist[i][j]>G->dist[i][k]+G->dist[k][j])

{

G->dist[i][j]=G->dist[i][k]+G->dist[k][j];

G->pred[i][j]=k;

}

}

}

disp(G);

}

}

void path(struct graph \*G,int i,int j)

{

//int t=0;

if(G->pred[i][j]==-1)

{

//G->dist[i][j];

printf("->%d",j+1);

}

else

{

path(G,i,G->pred[i][j]);

path(G,G->pred[i][j],j);

}

//return t;

}

**appl.c:**

#include "impl.h"

#include<stdlib.h>

int main()

{

struct graph \*G=(struct graph \*)malloc(sizeof(struct graph));

int ch;

printf("\nMenu:\n1.Create a graph\n2.Display the graph matrices\n3.Apply Floyd Warshall\n4.Display Path\n5.Exit\n");

do

{

printf("\nChoice: ");

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("Enter the number of verties and edges: ");

int v,e;

scanf("%d%d",&v,&e);

create(G,v,e,1);

break;

case 2:

disp(G);

break;

case 3:

floyd(G);

break;

case 4:

printf("Enter source and destination: ");

int s,d;

scanf("%d%d",&s,&d);

printf("\nPath from %d to %d: %d",s,d,s);

path(G,s-1,d-1);

printf("\nShoretst dist: %d\n",G->dist[s-1][d-1]);

break;

case 5:printf("\nExiting...\n");

break;

default:printf("\nInavlid Input!\n");

}

}while(ch!=5);

//create(G,3,5,1);

//int s=4,d=1;

}

*Sample I/O:*





