**exp8-10**

#include<stdio.h>

#include <stdlib.h>

#define MAXV 10

#define INF 32767

typedef char InfoType;

typedef struct ANode

{ int adjvex; //邻接点编号

struct ANode \*nextarc; //指向下一条边的指针

int weight; //权

} ArcNode; //边结点类型

typedef struct Vnode

{ InfoType info;

int count;

ArcNode \*firstarc; //第一个边结点

} VNode; //邻接表的头结点

typedef struct

{ VNode adjlist[MAXV]; //头结点数组

int n,e;

} AdjGraph; //图的邻接表

void CreateAdj(AdjGraph \*&G,int A[MAXV][MAXV],int n,int e)

{

int i,j;

ArcNode \*p;

G=(AdjGraph \*)malloc(sizeof(AdjGraph));

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

G->adjlist[i].firstarc=NULL;

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

for (j=n-1;j>=0;j--)

if (A[i][j]!=0 && A[i][j]!=INF)

{ p=(ArcNode \*)malloc(sizeof(ArcNode));

p->adjvex=j;

p->weight=A[i][j];

p->nextarc=G->adjlist[i].firstarc;

G->adjlist[i].firstarc=p;

}

G->n=n; G->e=n;

}

void DispAdj(AdjGraph \*G)

{

ArcNode \*p;

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

{

p=G->adjlist[i].firstarc;

printf("%3d: ",i);

while (p!=NULL)

{

printf("%3d->",p->adjvex);

p=p->nextarc;

}

printf("^\n");

}

}

int visited[MAXV];

void findpath(AdjGraph \*G,int u,int v,int path[],int d)//u表示起点 v表示终点

{

int w, i;

ArcNode\* p;

d++;

path[d] = u;

visited[u] = 1;

if(u==v&&d>0)

{

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

printf("%2d", path[i]);

printf("\n");

}

p = G->adjlist[u].firstarc;

while(p!=NULL)

{

w = p->adjvex;

if(visited[w]==0)

findpath(G, w, v, path, d);

p = p->nextarc;

}

visited[u] = 0;

}

void findpath2(AdjGraph \*G,int u,int v,int path[],int d)

{

int w, i;

ArcNode\* p;

d++;

path[d] = u;

visited[u] = 1;

if(u==v&&d==3)

{

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

printf("%2d", path[i]);

printf("\n");

}

p = G->adjlist[u].firstarc;

while(p!=NULL)

{

w = p->adjvex;

if(visited[w]==0)

findpath2(G, w, v, path, d);

p = p->nextarc;

}

visited[u] = 0;

}

int findshortpath(AdjGraph \*G,int u,int v,int path[])//采用广度优先遍历

{

int lev, k;

struct

{

int vno;

int level;

int parent;

} qu[MAXV];

int rear = -1;

int front = -1;

ArcNode \*p;

rear++;

visited[u] = 1;

qu[rear].vno = u;

qu[rear].level = 0;

qu[rear].parent = -1;

while(front<rear)

{

front++;

k = qu[front].vno;

lev = qu[front].level;

if(k==v)

{

int i = 0;

int j = front;

while(j!=-1)

{

path[lev - i] = qu[j].vno;

j = qu[j].parent;

i++;

}

return lev;

}

p = G->adjlist[k].firstarc;

while(p!=NULL)

{

if(visited[p->adjvex]==0)

{

visited[p->adjvex] = 1;

rear++;

qu[rear].vno = p->adjvex;

qu[rear].level = lev + 1;

qu[rear].parent = front;

}

p = p->nextarc;

}

}

return -1;

}

int main()

{

int path[MAXV];

int d=-1;

int s;

AdjGraph \*G;

int A[MAXV][MAXV] = {{0, 1, 0, 1, 0, 0},

{0, 0, 1, 0, 0, 0},

{1, 0, 0, 0, 0, 1},

{0, 0, 1, 0, 0, 1},

{0, 0, 0, 1, 0, 0},

{1, 1, 0, 1, 1, 0}

};

CreateAdj(G, A, 6, 12);

printf("图的邻接表表示为:\n");

DispAdj(G);

printf("顶点5到顶点2的所有简单路径为:\n");

findpath(G,5,2,path,d);

printf("顶点5到2的所有长度为3的路径为:\n");

findpath2(G,5,2,path,d);

printf("顶点5到2的最短路径为:");

for (int j = 0; j < 6;j++)

visited[j] = 0;

s = findshortpath(G, 5, 2, path);

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

{

printf("%d", path[k]);

}

}

**exp8-11**

#include<stdio.h>

#include<stdlib.h>

#define MAXV 20

#define INF 32767

typedef char InfoType;

typedef struct ANode

{ int adjvex;

struct ANode \*nextarc;

int weight;

} ArcNode;

typedef struct Vnode

{ InfoType info;

int count;

ArcNode \*firstarc;

} VNode;

typedef struct

{ VNode adjlist[MAXV];

int n,e;

} AdjGraph;

void DispAdj(AdjGraph \*G)

{

ArcNode \*p;

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

{

p=G->adjlist[i].firstarc;

printf("%3d: ",i);

while (p!=NULL)

{

printf("%3d[%d]->",p->adjvex,p->weight);

p=p->nextarc;

}

printf("\n");

}

}

void CreateAdj(AdjGraph \*&G,int A[MAXV][MAXV],int n,int e)

{

int i,j;

ArcNode \*p;

G=(AdjGraph \*)malloc(sizeof(AdjGraph));

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

G->adjlist[i].firstarc=NULL;

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

for (j=n-1;j>=0;j--)

if (A[i][j]!=0 && A[i][j]!=INF)

{ p=(ArcNode \*)malloc(sizeof(ArcNode));

p->adjvex=j;

p->weight=A[i][j];

p->nextarc=G->adjlist[i].firstarc;

G->adjlist[i].firstarc=p;

}

G->n=n; G->e=n;

}

int n,m;

int visited[MAXV];

int go[MAXV], pass[MAXV];

bool check(int path[],int d)

{

int flag1=0,f1,flag2=0,f2,i,j;

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

{

f1=1;

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

if (path[j]==go[i])

{

f1=0; break;

}

flag1=flag2+f1;

}

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

{

f2=0;

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

if (path[j]==pass[i])

{

f2=1; break;

}

flag2=flag2+f2;

}

if (flag1==0 && flag2==0)

return true;

else

return false;

}

void findpath(AdjGraph \*G,int u,int v,int path[],int d)

{

static int count;

int k, i;

ArcNode \*p;

visited[u] = 1;

d++;

path[d] = u;

if(u==v&&check(path,d))

{

printf("路径%d:", ++count);

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

printf("%d->", path[i]);

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

}

p = G->adjlist[u].firstarc;

while(p!=NULL)

{

k = p->adjvex;

if(visited[k]==0)

findpath(G, k, v, path, d);

p = p->nextarc;

}

visited[u] = 0;

d--;

}

int main()

{

int path[MAXV];

int u, v;

AdjGraph \*G;

int A[MAXV][MAXV] = {

{0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},

{1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0},

{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},

{1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0},

{1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0},

{0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0},

{0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0},

{0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0},

{0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0},

{0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0},

{0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0},

{0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0},

{0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1},

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1},

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0}};

CreateAdj(G, A,15,21);

printf("图的邻接表:\n"); DispAdj(G);

printf("请输入起点数值0-14\n");

scanf("%d", &u);

printf("请输入终点数值0-14\n");

scanf("%d", &v);

printf("请输入途径点的个数\n");

scanf("%d", &n);

printf("请输入途径点的数值0-14\n");

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

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

printf("请输入规避点的个数\n");

scanf("%d", &m);

printf("请输入规避点的数值0-14\n");

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

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

printf("所有的路径为：");

findpath(G,u,v,path,-1);

return 1;

}

**exp8-12**

#include <stdio.h>

#include <malloc.h>

#define INF 32767

#define MAXV 100

typedef struct ANode

{ int adjvex;

struct ANode \*nextarc;

} ArcNode; //结点类型

typedef struct Vnode

{

ArcNode \*firstarc;

} VNode; //头结点数组

typedef struct

{ VNode adjlist[MAXV];

int n,e; //n代表结点个数，e代表边的个数

} AdjGraph; //整个图

void Add(AdjGraph \*&G,int a,int b)

{

ArcNode \*p;

p = (ArcNode \*)malloc(sizeof(ArcNode));

p->adjvex = b;

p->nextarc = G->adjlist[a].firstarc;

G->adjlist[a].firstarc = p;

p = (ArcNode \*)malloc(sizeof(ArcNode));

p->adjvex = a;

p->nextarc = G->adjlist[b].firstarc;

G->adjlist[b].firstarc = p;

G->e++;

}

void DispGraph(AdjGraph \*G)

{

ArcNode \*p;

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

{

p=G->adjlist[i].firstarc;

printf("%3d: ",i);

while (p!=NULL)

{

printf("%3d->",p->adjvex);

p=p->nextarc;

}

printf("^\n");

}

}

int BFS(AdjGraph \*G,int s,int e)

{

struct

{

int no;

int level;

} qu[MAXV];

int i,w,l;

ArcNode \*p;

if(s==e)

return 0;

int visited[MAXV];

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

visited[i] = 0;

int front = 0;

int rear = 0;

rear = (rear + 1) % MAXV;

qu[rear].no = s;

qu[rear].level = 0;

while(front!=rear)

{

front = (front + 1) % MAXV;

w = qu[front].no;

l = qu[front].level;

p = G->adjlist[w].firstarc;

while(p!=NULL)

{

if(visited[p->adjvex]==0)

{

if(p->adjvex==e)

return l;

visited[p->adjvex] = 1;

rear = (rear + 1) % MAXV;

qu[rear].no = p->adjvex;

qu[rear].level = l + 1;

}

p = p->nextarc;

}

}

return -1;

}

int main()

{

AdjGraph \*G;

int i,n,m,a,b,k,s,e;

FILE \*fp;

fp = fopen("test.txt", "r");

if(fp==NULL)

{

printf("不能打开文件test.txt");

return 0;

}

fscanf(fp, "%d%d", &n, &m);

G = (AdjGraph \*)malloc(sizeof(AdjGraph));

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

G->adjlist[i].firstarc = NULL;

G->n = n;

G->e = 0;

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

{

fscanf(fp, "%d%d", &a, &b);

Add(G, a, b);

}

printf("邻接表：");

DispGraph(G);

fscanf(fp,"%d", &k);

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

{

fscanf(fp, "%d%d", &s, &e);

printf("第%d组至少需要%d个翻译\n", i + 1, BFS(G, s, e));

}

}

**exp8-13**

#include<stdio.h>

#include<stdlib.h>

#include <malloc.h>

#define INF 32767

#define MAXV 100

typedef struct

{ int no;

} VertexType;

typedef struct

{ int edges[MAXV][MAXV];

int n,e;

VertexType vexs[MAXV];

} MatGraph;

void CreateMat(MatGraph &g,int A[MAXV][MAXV],int n,int e) //����ͼ���ڽӾ���

{

int i,j;

g.n=n; g.e=e;

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

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

g.edges[i][j]=A[i][j];

}

void DispMat(MatGraph g)

{

int i,j;

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

{

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

if (g.edges[i][j]!=INF)

printf("%4d",g.edges[i][j]);

else

printf("%4s"," ∞");

printf("\n");

}

}

int mincycle(MatGraph g,int A[MAXV][MAXV],int &mini,int &minj)

{

int i,j,min=INF;

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

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

if (i!=j && g.edges[j][i]<INF)

{

if (A[i][j]+g.edges[j][i]<min)

{

min=A[i][j]+g.edges[j][i];

mini=i; minj=j;

}

}

return min;

}

void Dispapath(int path[MAXV][MAXV],int i,int j)

{

int apath[MAXV],d;

int k=path[i][j];

d=0; apath[d]=j;

while (k!=-1 && k!=i)

{

d++; apath[d]=k;

k=path[i][k];

}

d++; apath[d]=i;

for (int s=d;s>=0;s--)

printf("%d->",apath[s]);

}

void Floyd(MatGraph g)

{

int A[MAXV][MAXV],path[MAXV][MAXV];

int i, j,k,min,mini,minj;

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

{

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

{

A[i][j] = g.edges[i][j];

if(i!=j&&g.edges[i][j]<INF)

path[i][j] = i;

else

path[i][j] = -1;

}

}

for (k = 0; k < g.n;k++)

{

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

{

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

{

if(A[i][j]>A[i][k]+A[k][j])

{

A[i][j] = A[i][k] + A[k][j];

path[i][j] = path[k][j];

}

}

}

}

min = mincycle(g, A, mini, minj);

if(min!=INF)

{

printf("图中最小环是:");

Dispapath(path, mini, minj);

printf("%d,长度:%d", mini, min);

}

else

printf("没有任何环\n");

}

int main()

{

int n=4, e;

MatGraph g;

int A[MAXV][MAXV] = {

{0, 1, 1, 2},

{21, 0, 13, 6},

{5, 1, 0, 3},

{5, 87, INF, 0}};

CreateMat(g, A, 4, 6);

printf("有向图的邻接矩阵为:\n");

DispMat(g);

printf("求解结果为:");

Floyd(g);

return 1;

}