

数据结构

****

**题 目 游乐场导航**

学生姓名 沈宇豪

学 号 202083290386

学 院 计算机与软件学院

专 业 计算机类

授课教师 陈金辉

**二Ｏ二一 年 12 月 6日**

目录

1. 需求分析-----------------------------------------------------------------------2
2. 概要设计-----------------------------------------------------------------------3
3. 详细设计-----------------------------------------------------------------------8
4. 调试分析----------------------------------------------------------------------12
5. 附录代码----------------------------------------------------------------------13

1.需求分析

1.1问题描述：

实现游乐场导航问题

基本要求：

设计要求：设计游乐场导游图，至少包括10个以上的场所，找出从游乐场大门口到达其他所有游乐场所的最佳路径（最短路径，迪杰斯特拉算法）。

1.2 设计构想

数据结构类型使用图，用图来描述游乐场网络图。

通过迪杰斯特拉算法来计算最短路径，来实现要求。

用A来代表门口（起点），从A开始计算

引进两个数组S1和S2.。 S1的作用是来记录已经求出最短路径的顶点和相应的最短路径，而S2是来记录还未求出最短路径的顶点和它到起点A的距离。

一开始数组S1中只有A，而数组S2中是除了A以外的顶点，并且数组S2中记录各个顶点到起点A的距离。如果顶点和起点A 不相邻，距离未无穷大。

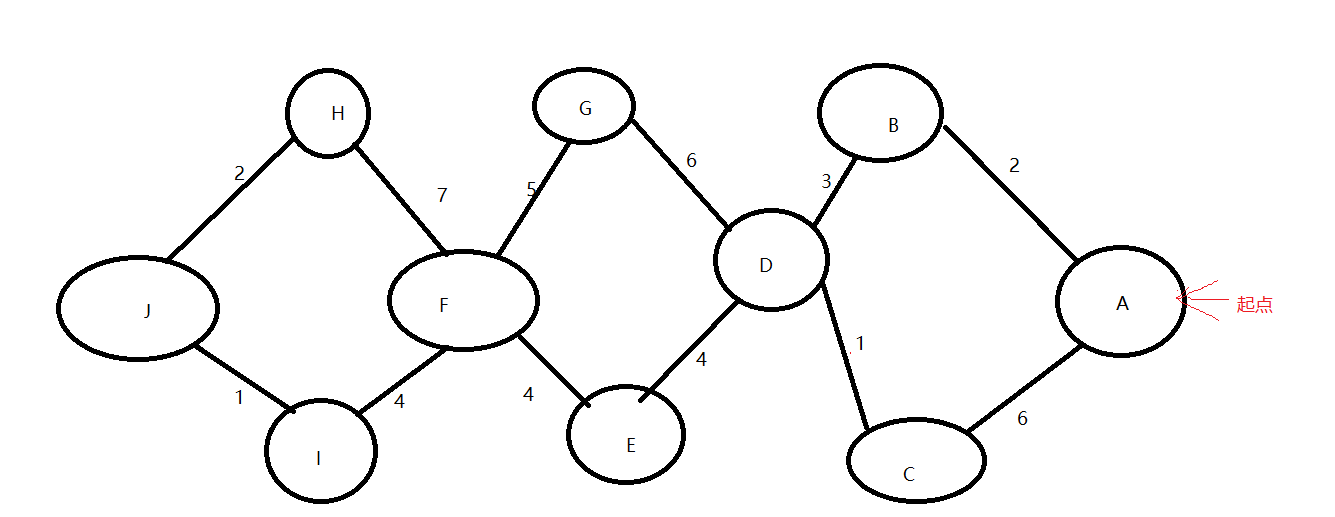
从S2中找到最短的顶点B，把他加入S1中，同时S2中移除顶点B，接着重新计算各个顶点到A的距离。

再从S2中找到最近的C，重复........

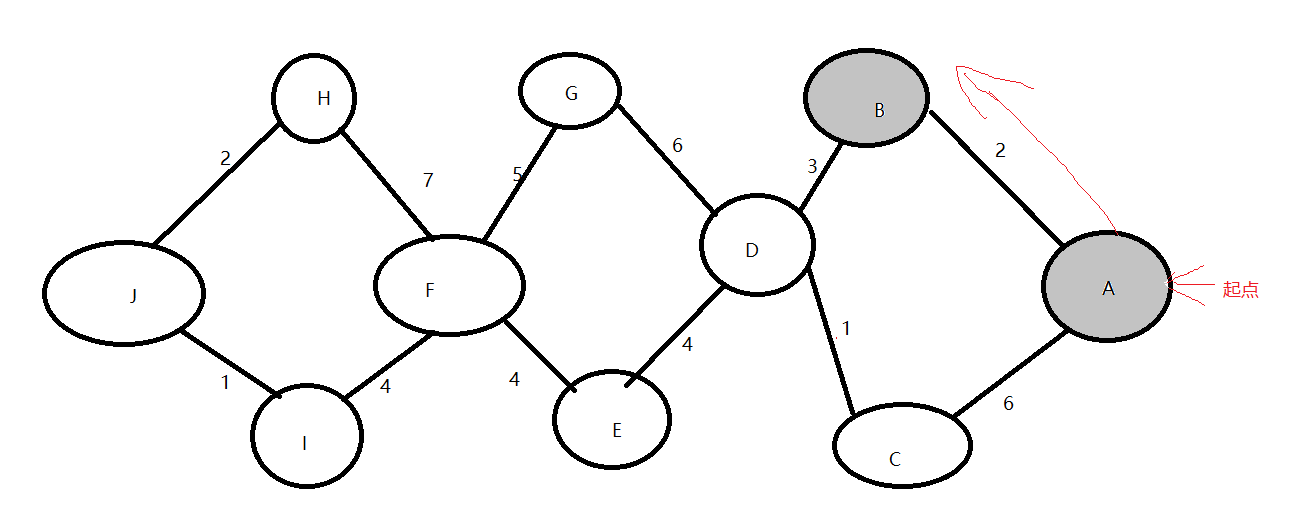
1. 概要设计：

使用迪杰斯特拉算法来完成设计

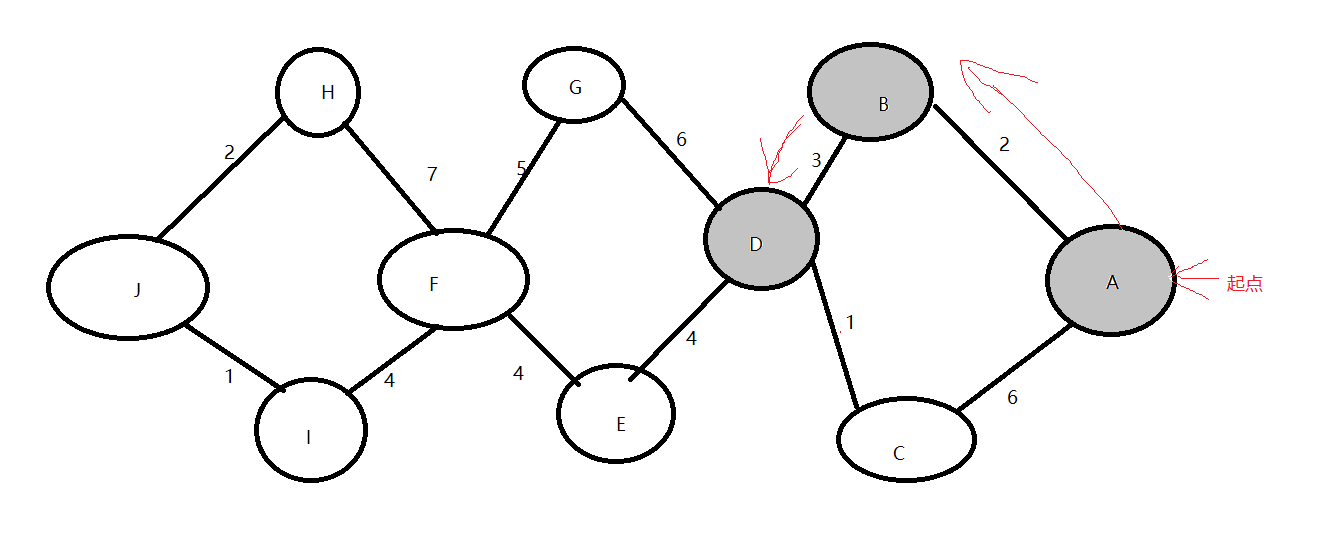
算法图解如下：



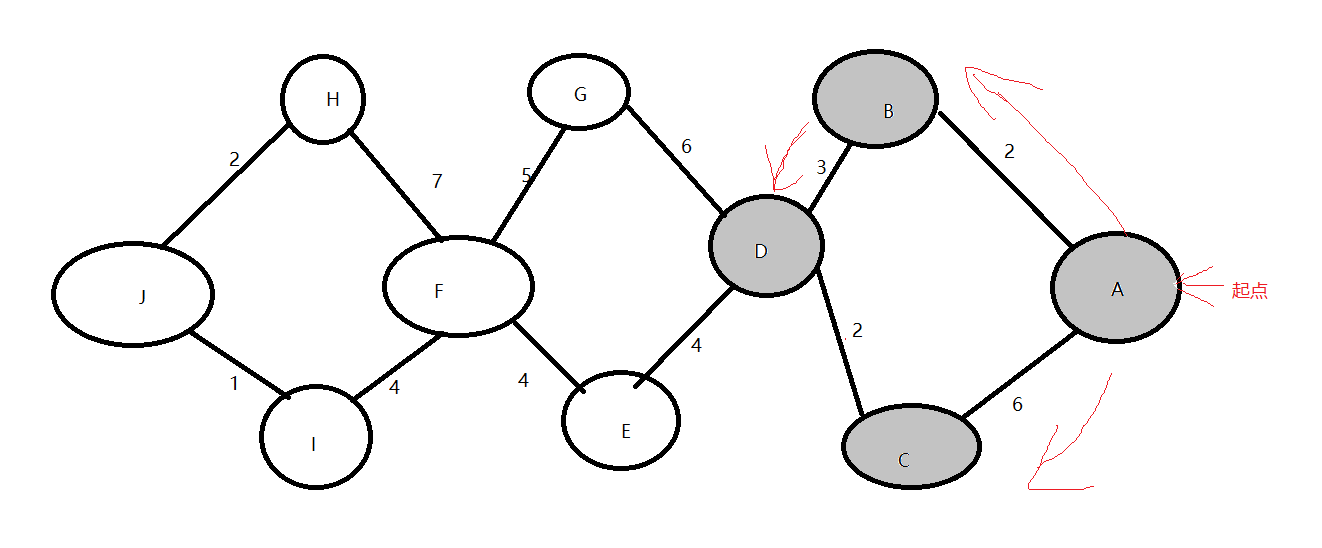
A是这张图的起点



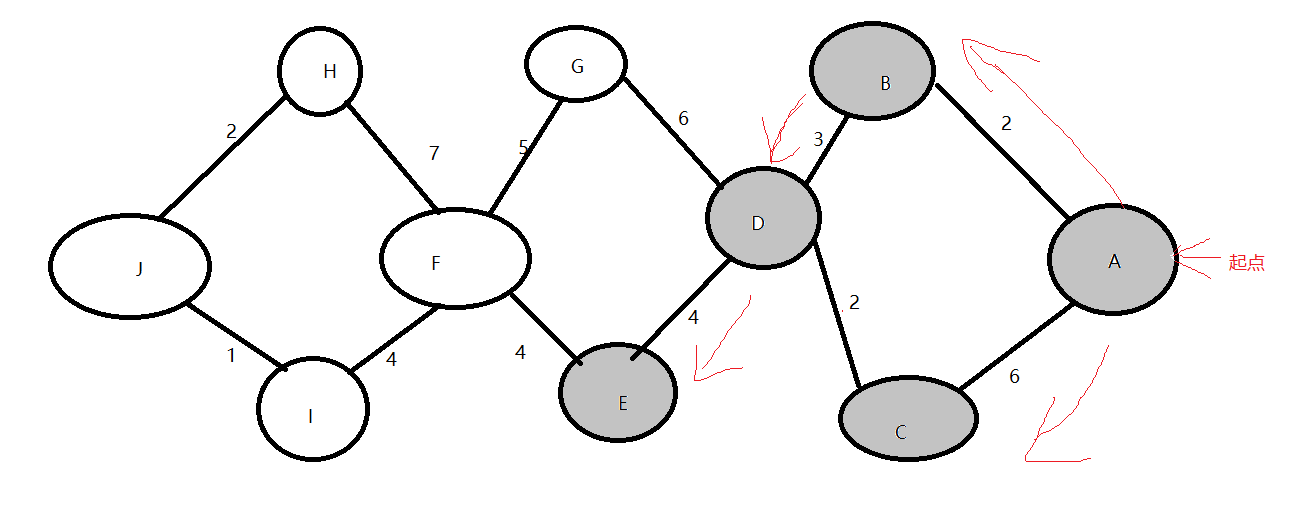
A可以通到B和C，B权值为2，短，先联通B



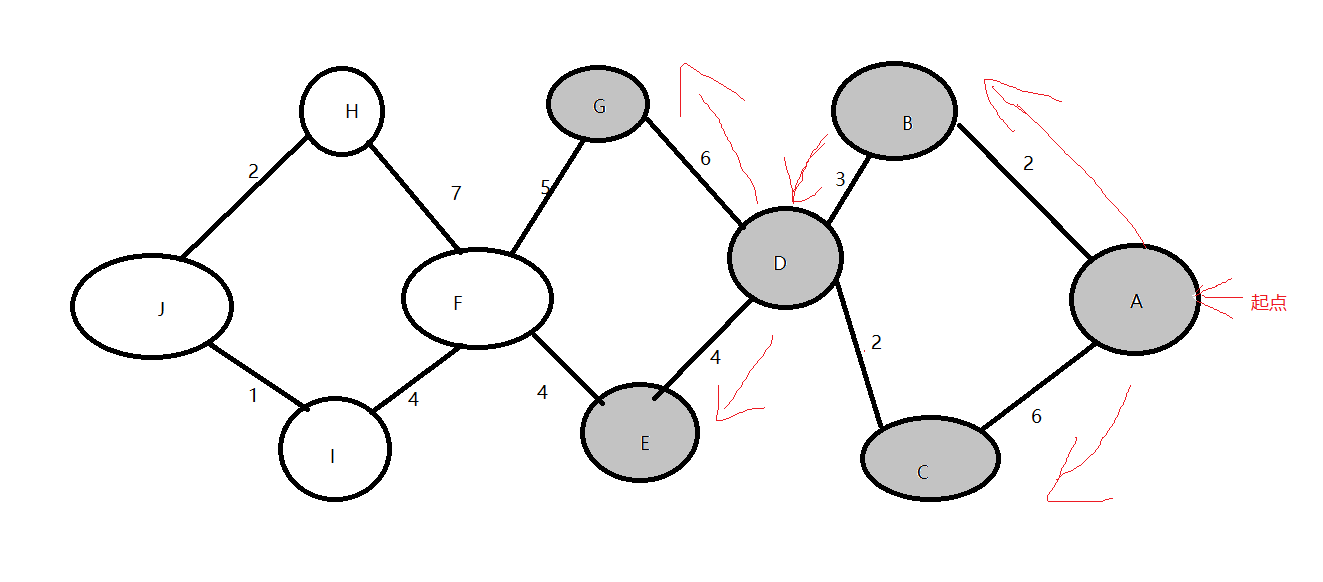
A联通B后，此时对A来说，D和C都是联通的，通过计算到达D更短，先链接D



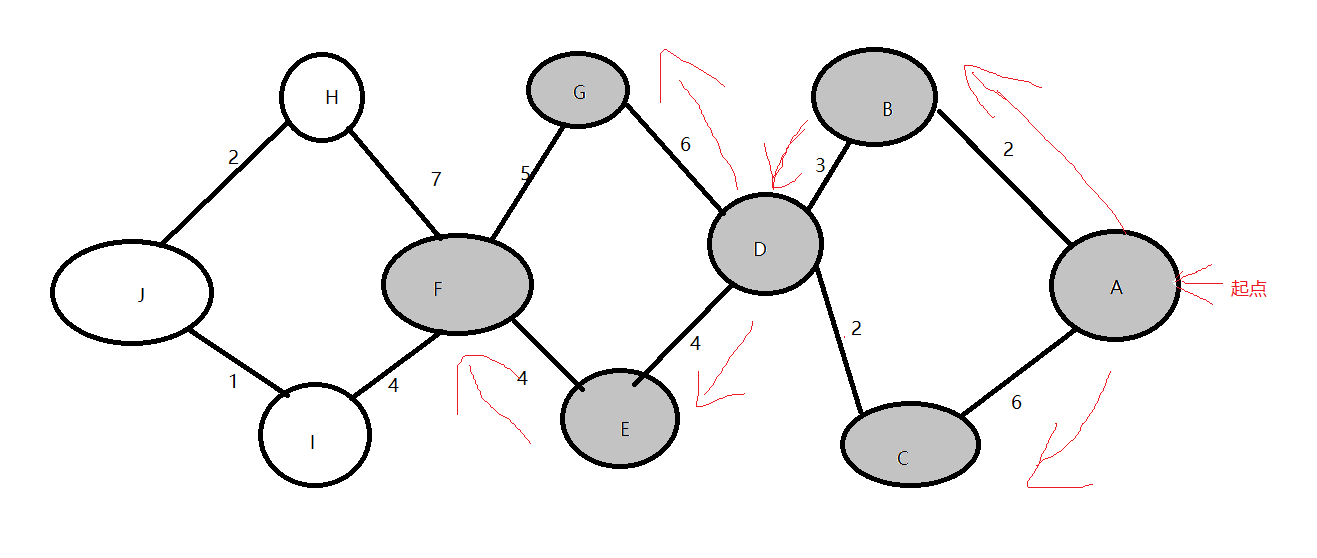
A联通D后，此时对A来说，G,E和C都是联通的，通过计算到达C更短，先链接C



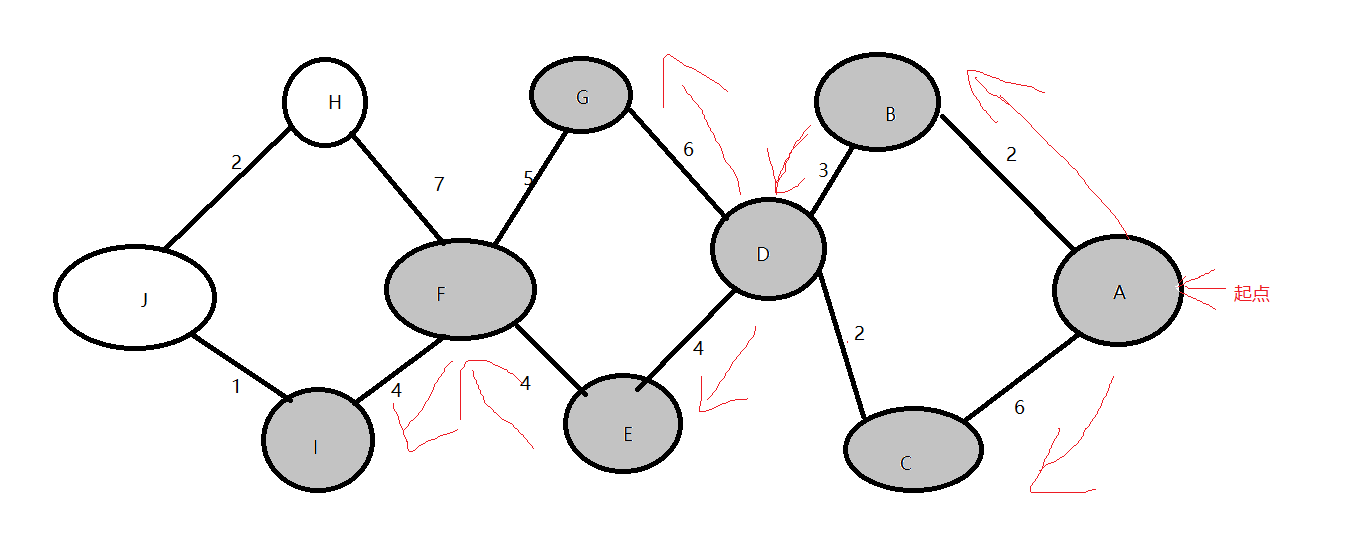
A联通C后，此时对A来说，G和E都是联通的，通过计算到达E更短，先链接E



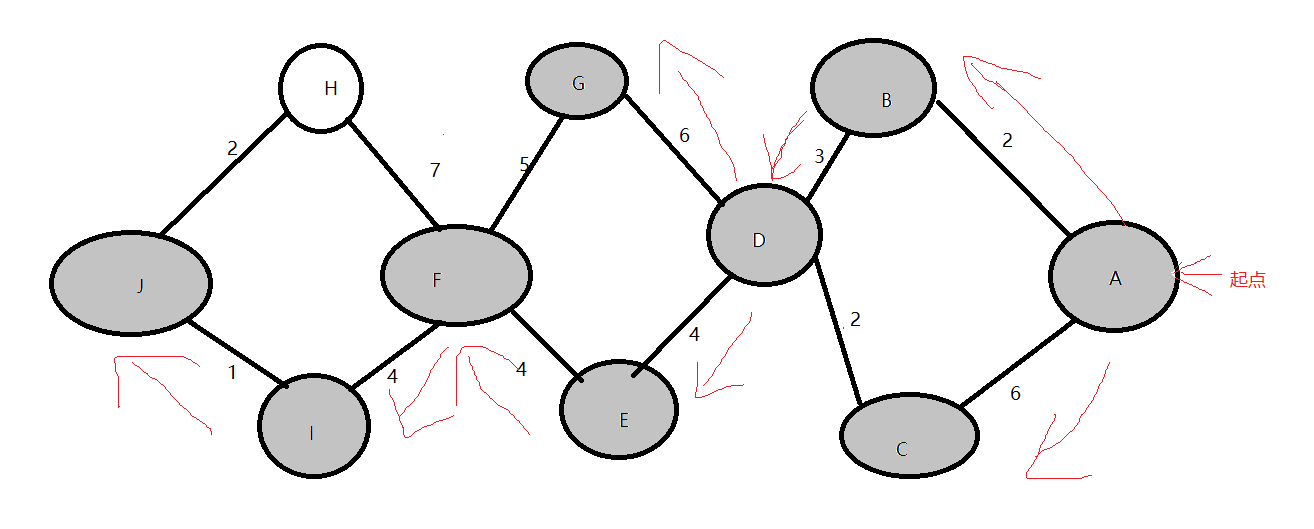
A联通E后，此时对A来说，F和G都是联通的，通过计算到达G更短，先链接G



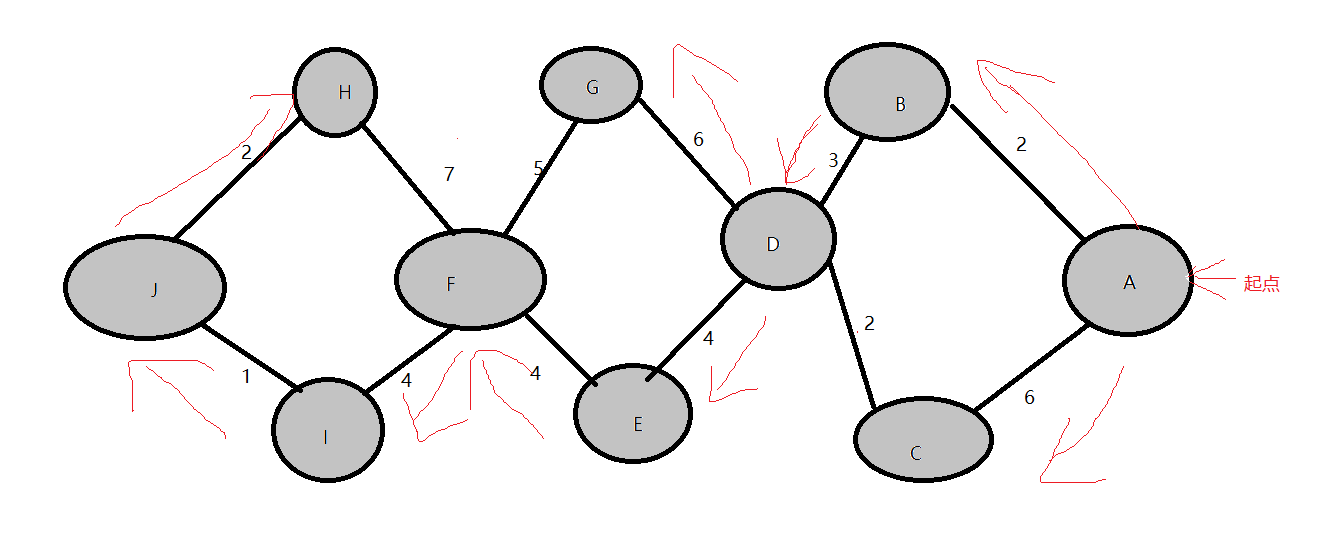
A联通E后，此时对A来说，F是联通的，通过计算到达F更短，先链接F



A联通F后，此时对A来说，H和I都是联通的，通过计算到达I更短，先链接I



A联通I后，此时对A来说，J和H都是联通的，通过计算到达J更短，先链接J



。

A联通J后，此时对A来说，只有H未联通，链接H，则图全部走完。

每次连接完一个节点都要将该节点到A的最短路径记入数组S2。

3.详细设计

1.定义图：

typedef struct Graph

{

char vexs[max]; //顶点集合

int vexnum; //顶点数

int edgnum; //边数

int matrix[max][max];//邻接矩阵

}Graph,\*PGraph;

定义边：

typedef struct Edgedata

{

char start; //边的起点

char end; //边的终点

int weight; //边的权重

}EData;

1. 图的基本操作:

创建图：

Graph\* create\_graph()

{

char vexs[]={'A','B','C','D','E','F','G','H','I','J'};

int matrix[][10]=

{

{0,2,6,100,100,100,100,100,100,100},//A

{2,0,100,3,100,100,100,100,100,100},//B

{6,100,0,1,100,100,100,100,100,100},//C

{100,3,1,0,4,100,6,100,100,100},//D

{100,100,100,4,0,4,100,100,100,100},//E

{100,100,100,100,4,0,5,7,4,100},//F

{100,100,100,6,100,5,0,100,100,100},//G

{100,100,100,100,100,7,100,0,100,2},//H

{100,100,100,100,100,4,100,100,0,1},//I

{100,100,100,100,100,100,100,2,1,0},//J

};

int vlen=sizeof(vexs);

int i,j;

Graph \*pG;

if((pG=(Graph\*)malloc(sizeof(Graph)))==NULL)

return NULL;

memset(pG,0,sizeof(pG));

pG->vexnum=vlen;

for(i=0;i<pG->vexnum;i++)

pG->vexs[i]=vexs[i];

for(i=0;i<pG->vexnum;i++)

{

for(j=0;j<pG->vexnum;j++)

{

pG->matrix[i][j]=matrix[i][j];

}

}

for(i=0;i<pG->vexnum;i++)

{

for(j=0;j<pG->vexnum;j++)

{

if(i!=j&&pG->matrix[i][j]!=-1)

pG->edgnum++;

}

}

pG->edgnum/=2;

return pG;

}

输出图：

void print\_graph(Graph G)

{

int i,j;

printf("图为： \n");

for(i=0;i<G.vexnum;i++)

{

for(j=0;j<G.vexnum;j++)

{

printf("%10d",G.matrix[i][j]);

}

printf("\n");

}

}

对边权重的操作和赋值

EData\* get\_edges(Graph G)

{

EData \*edges;

edges=(EData\*)malloc(G.edgnum\*sizeof(EData));

int i,j;

int index=0;

for(i=0;i<G.vexnum;i++)

{

for(j=0;j<G.edgnum;j++)

{

if(G.matrix[i][j]!=100)

{

edges[index].start=G.vexs[i];

edges[index].end=G.vexs[j];

edges[index].weight=G.matrix[i][j];

index++;

}

}

}

return edges;

}

迪杰斯特拉算法：

void dijkstra(Graph G,int vs,int prev[],int dist[])

{

int i,j,k;

int min;

int tmp;

int flag[max];

for(i=0;i<G.vexnum;i++)

{

flag[i]=0;

prev[i]=vs;

dist[i]=G.matrix[vs][i];

}

flag[vs]=1;

dist[vs]=0;

for(i=1;i<G.vexnum;i++)

{

min=100;

for(j=0;j<G.vexnum;j++)

{

if(flag[j]==0&&dist[j]<min)

{

min=dist[i];

k=j;

}

}

flag[k]=1;

for(j=0;j<G.vexnum;j++)

{

tmp=((G.matrix[k][j]==-1)?-1:(min+G.matrix[k][j]));

if(flag[j]==0&&tmp<dist[j])

{

dist[j]=tmp;

prev[j]=k;

}

}

}

printf("迪杰斯特拉算法后：\n",G.vexs[vs]);

for(i=0;i<G.vexnum;i++)

printf("从A到%c最短路径长度为：%d\n",G.vexs[i],dist[i]);

}

主函数：

int main()

{

printf("游乐园导航\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("(不妨假设游乐园入口为A，各个景点为B~J。)\n");

printf("\n");

Graph \*pG;

pG=create\_graph();

print\_graph(\*pG);

int prev[max];

int dist [max];

dijkstra(\*pG,0,prev,dist);

int i,j;

for(i=0;i<pG->vexnum;i++)

{

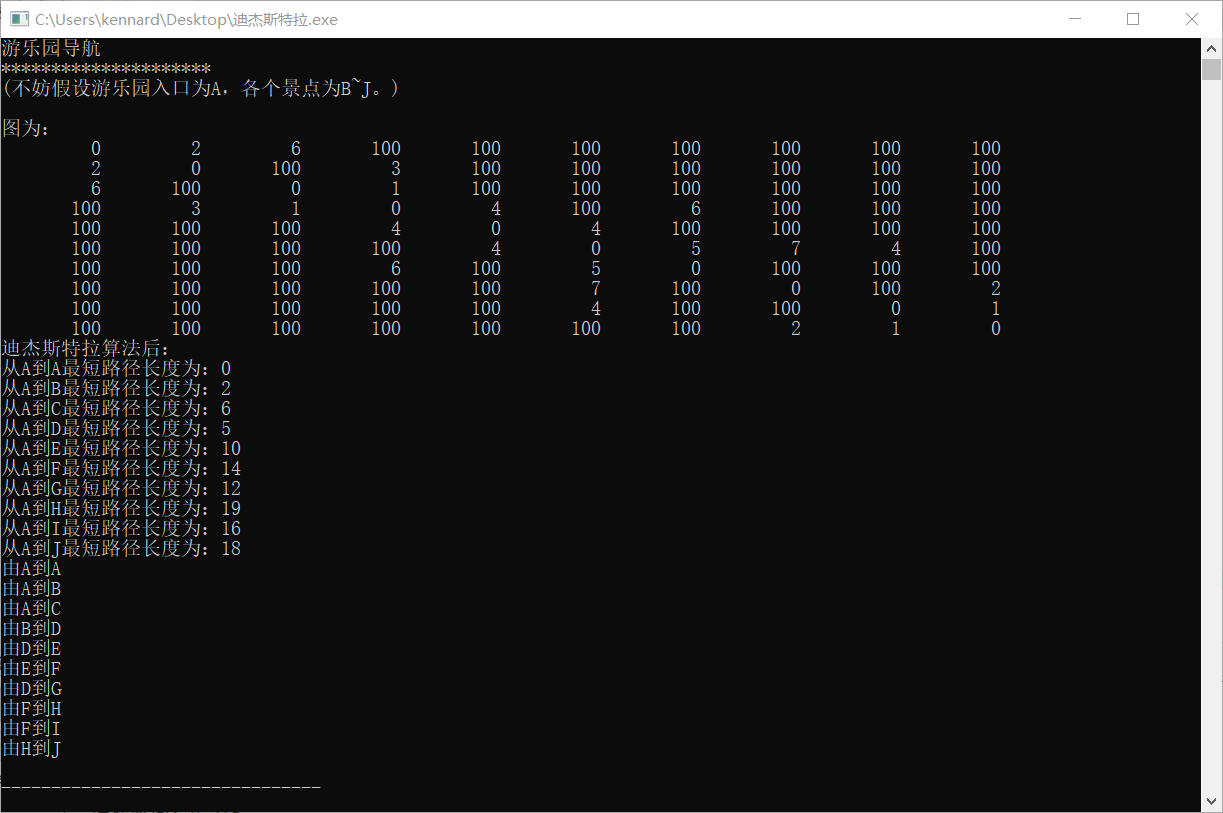
printf("由%c到%c \n",pG->vexs[prev[i]],pG->vexs[i]);

}

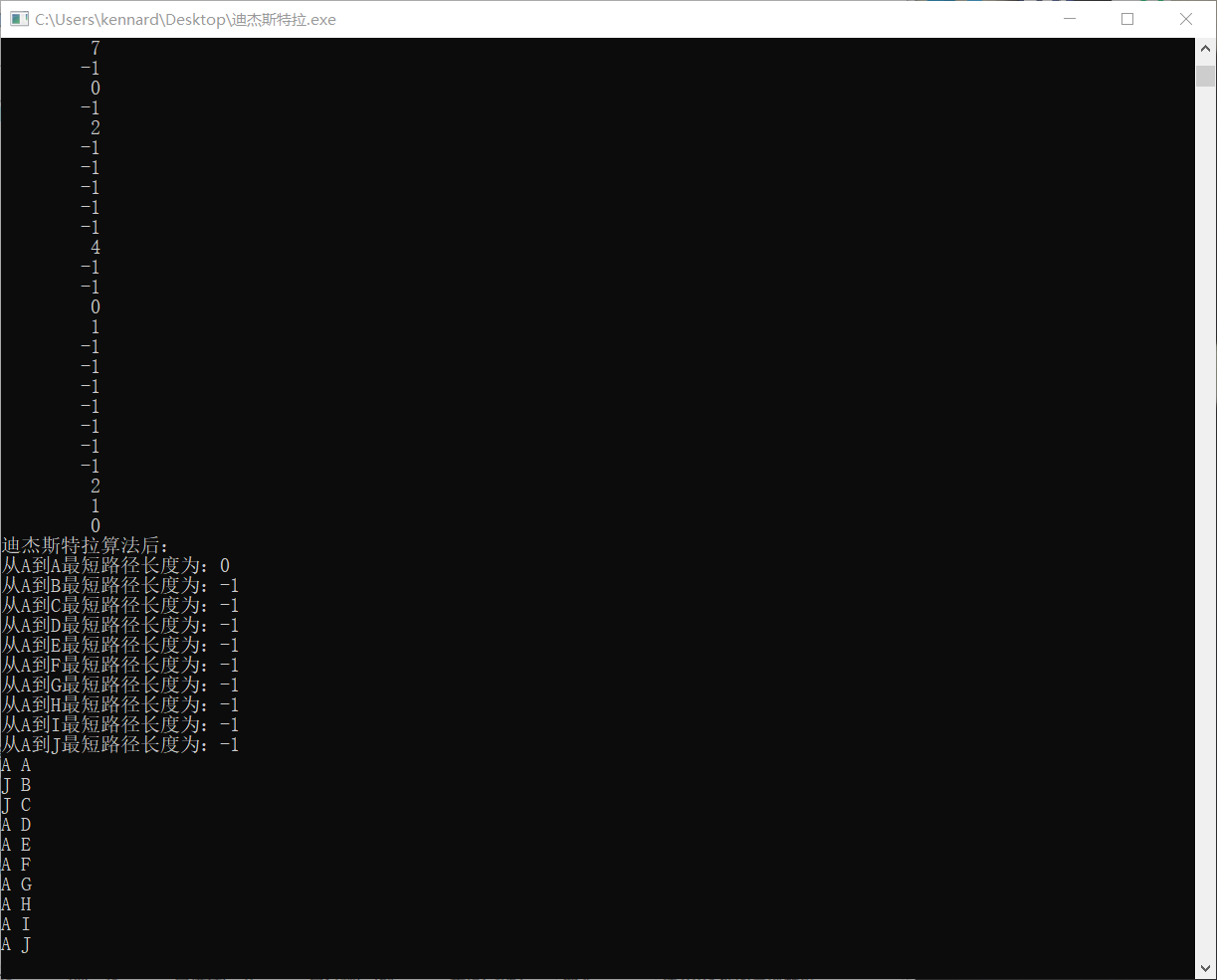
}

1. 调试分析

成功调试：



1. 失败调试



失败分析：

1. 在输出图时，因为使用的时邻接矩阵，所以应该输出完一行再换行，而不是输出一个换行一个。
2. 在对未链接的两个结点定义时，用-1来代表两个结点不连通在求最短路径时是行不通的，应该将它设为最大值。

附录代码：

#include<stdio.h>

#include<stdlib.h>

#include<malloc.h>

#include<string.h>

#define max 100

typedef struct Graph

{

char vexs[max]; //顶点集合

int vexnum; //顶点数

int edgnum; //边数

int matrix[max][max];//邻接矩阵

}Graph,\*PGraph;

typedef struct Edgedata

{

char start; //边的起点

char end; //边的终点

int weight; //边的权重

}EData;

static int get\_position(Graph g,char ch)

{

int i;

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

{

if(g.vexs[i]==ch)

return i;

}

return -1;

}

Graph\* create\_graph()

{

char vexs[]={'A','B','C','D','E','F','G','H','I','J'};

int matrix[][10]=

{

{0,2,6,100,100,100,100,100,100,100},//A

{2,0,100,3,100,100,100,100,100,100},//B

{6,100,0,1,100,100,100,100,100,100},//C

{100,3,1,0,4,100,6,100,100,100},//D

{100,100,100,4,0,4,100,100,100,100},//E

{100,100,100,100,4,0,5,7,4,100},//F

{100,100,100,6,100,5,0,100,100,100},//G

{100,100,100,100,100,7,100,0,100,2},//H

{100,100,100,100,100,4,100,100,0,1},//I

{100,100,100,100,100,100,100,2,1,0},//J

};

int vlen=sizeof(vexs);

int i,j;

Graph \*pG;

if((pG=(Graph\*)malloc(sizeof(Graph)))==NULL)

return NULL;

memset(pG,0,sizeof(pG));

pG->vexnum=vlen;

for(i=0;i<pG->vexnum;i++)

pG->vexs[i]=vexs[i];

for(i=0;i<pG->vexnum;i++)

{

for(j=0;j<pG->vexnum;j++)

{

pG->matrix[i][j]=matrix[i][j];

}

}

for(i=0;i<pG->vexnum;i++)

{

for(j=0;j<pG->vexnum;j++)

{

if(i!=j&&pG->matrix[i][j]!=-1)

pG->edgnum++;

}

}

pG->edgnum/=2;

return pG;

}

void print\_graph(Graph G)

{

int i,j;

printf("图为： \n");

for(i=0;i<G.vexnum;i++)

{

for(j=0;j<G.vexnum;j++)

{

printf("%10d",G.matrix[i][j]);

}

printf("\n");

}

}

EData\* get\_edges(Graph G)

{

EData \*edges;

edges=(EData\*)malloc(G.edgnum\*sizeof(EData));

int i,j;

int index=0;

for(i=0;i<G.vexnum;i++)

{

for(j=0;j<G.edgnum;j++)

{

if(G.matrix[i][j]!=100)

{

edges[index].start=G.vexs[i];

edges[index].end=G.vexs[j];

edges[index].weight=G.matrix[i][j];

index++;

}

}

}

return edges;

}

void dijkstra(Graph G,int vs,int prev[],int dist[])

{

int i,j,k;

int min;

int tmp;

int flag[max];

for(i=0;i<G.vexnum;i++)

{

flag[i]=0;

prev[i]=vs;

dist[i]=G.matrix[vs][i];

}

flag[vs]=1;

dist[vs]=0;

for(i=1;i<G.vexnum;i++)

{

min=100;

for(j=0;j<G.vexnum;j++)

{

if(flag[j]==0&&dist[j]<min)

{

min=dist[i];

k=j;

}

}

flag[k]=1;

for(j=0;j<G.vexnum;j++)

{

tmp=((G.matrix[k][j]==-1)?-1:(min+G.matrix[k][j]));

if(flag[j]==0&&tmp<dist[j])

{

dist[j]=tmp;

prev[j]=k;

}

}

}

printf("迪杰斯特拉算法后：\n",G.vexs[vs]);

for(i=0;i<G.vexnum;i++)

printf("从A到%c最短路径长度为：%d\n",G.vexs[i],dist[i]);

}

int main()

{

printf("游乐园导航\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("(不妨假设游乐园入口为A，各个景点为B~J。)\n");

printf("\n");

Graph \*pG;

pG=create\_graph();

print\_graph(\*pG);

int prev[max];

int dist [max];

dijkstra(\*pG,0,prev,dist);

int i,j;

for(i=0;i<pG->vexnum;i++)

{

printf("由%c到%c \n",pG->vexs[prev[i]],pG->vexs[i]);

}

}