罗马尼亚问题

171491106 安南 硬件一班

一、实验题目

解决罗马尼亚问题

二、算法简介

迪杰斯特拉(**Dijkstra**)算法是典型最短路径算法，用于计算一个节点到其他节点的最短路径。  
它的主要特点是以起始点为中心向外层层扩展(广度优先搜索思想)，直到扩展到终点为止。

三、实验目的

运用迪杰斯特拉算法（Dijkstra算法），找到Arad到Bucharest的最短路径

四、实验代码

**package** 罗马尼亚;

**import** java.util.Scanner;

**public** **class** path {

**static** Scanner *s*=**new** Scanner(System.***in***);

**public** **static** **void** main(String[] args) {

String[] names={"A","B","M","C","D","E","F","G","H","I","J",

"K","L"};

arc[] arc={**new** arc("D","E",75),**new** arc("D","I",140),**new** arc("D","C",118),**new** arc("C","M",111),

**new** arc("M","B",70),**new** arc("B","A",75),**new** arc("A","G",120),**new** arc("G","J",138),**new** arc("G","H",146),

**new** arc("J","K",101),**new** arc("K","L",211),

**new** arc("H","J",97),**new** arc("H","I",80),**new** arc("I","L",99),**new** arc("I","F",151),**new** arc("F","E",71)};

Graph gra = *createGraph*(names,arc);

System.***out***.println(gra);

**int** path[]=*Dijkstra*(gra,"D");

*Dispath*(gra,path,"D","K");

}

**static** Graph createGraph(String[] names,arc[] arc){//String储存节点名称，arc储存每一个节点的边所到到的边和权值

**int** n=names.length;

Graph graph =**new** Graph(n);

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

graph.graph[i]=**new** vexnode();

graph.graph[i].name=names[i];

}

**for**(**int** i=0;i<arc.length;i++){

*add*(arc[i].name1,arc[i].name2,arc[i].weight,graph);

*add*(arc[i].name2,arc[i].name1,arc[i].weight,graph);

}

**return** graph;

}

**static** **void** add(String name1,String name2,**int** weight,Graph graph){

**int** i=*find*(name1,graph);

**int** j=*find*(name2,graph);

**if**(i==-1||j==-1)

{

System.***out***.println(name1+"->"+name2+"失败");

**return**;

}

arcnode ar=**new** arcnode();

ar.ver=j;

ar.weight=weight;

ar.nextarc=**null**;

arcnode pre = **null**;

arcnode temp=graph.graph[i].firstarc;

**if**(temp==**null**){//顶点之前无边

graph.graph[i].firstarc=ar;

}

**else**

{**while**(temp!=**null**)//顶点之前无边

{

pre=temp;

temp=temp.nextarc;

}

pre.nextarc=ar;

}

}

**static** **int** find(String name,Graph graph){

**int** n=graph.graph.length;

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

**if**(graph.graph[i].name.equals(name))

**return** i;

**return** -1;

}

**static** **int**[] Dijkstra(Graph graph,String start){

**int** v=*find*(start,graph);

**int**[] path=**new** **int**[graph.graph.length];

**for**(**int** i=0;i<graph.graph.length;i++)

path[i]=-1;

**int**[] dist=**new** **int**[graph.graph.length];

**for**(**int** i=0;i<graph.graph.length;i++)

dist[i]=1000000;

**int**[] s=**new** **int**[graph.graph.length];//s保存那哪些结点已经被访问过

**if**(v==-1){

System.***out***.println("无"+start+"这个节点");

**return** **null**;

}

arcnode p =graph.graph[v].firstarc;

**while**(p!=**null**)

{

path[p.ver]=v;

dist[p.ver]=p.weight;

p=p.nextarc;

}

s[v]=1;//顶点V加入s中

**int** u=0,mindis=1000000;

**for**(**int** i=0;i<graph.graph.length;i++){

mindis=1000000;

**for**(**int** j=0;j<graph.graph.length;j++)

**if**((s[j]==0)&&dist[j]<mindis){

u=j;

mindis=dist[j];

}

s[u]=1;//顶点u加入s中

p =graph.graph[u].firstarc;

**while**(p!=**null**)

{

**if**(s[p.ver]==0)

**if**(dist[p.ver]>dist[u]+p.weight){

dist[p.ver]=dist[u]+p.weight;

path[p.ver]=u;

}

p=p.nextarc;

}

}

**return** path;

}

**static** **void** Dispath(Graph graph,**int**[] path,String start,String end){

**int** sta=*find*(start,graph);

**int** en=*find*(end,graph);

**if**(sta==-1||en==-1){

System.***out***.println("没有找到相关结点");

**return**;

}

**if**(path[en]==-1){

System.***out***.println("路径有误");

**return**;

}

**else**{

**int** temp=0,i=1;

**int**[] path1=**new** **int**[path.length];

path1[0]=en;

temp=path1[0];

**while**(path[temp]!=sta){

path1[i]=path[temp];

temp=path1[i];

i++;

}

path1[i]=sta;

System.***out***.println(start+"到"+end+"最短路径为：");

**for**(**int** j=i;j>0;j--){

System.***out***.print(graph.graph[path1[j]].name+"-->");

}

System.***out***.println(graph.graph[path1[0]].name);

}

}

}

**class** arcnode{//arc，网弧节点

**int** ver=-1;//指向边结点

**int** weight=0;//权值

arcnode nextarc=**null**;

}

**class** vexnode//vertex顶点节点

{

String name;

arcnode firstarc=**null**;//指向第一条弧节点

} //顶点结点

**class** Graph{//储存图

vexnode[] graph;//一个叫graph的vexnode

Graph(**int** num){

graph=**new** vexnode[num];

}

**public** String toString(){

String[] names=**new** String[graph.length];

**for**(**int** i=0;i<graph.length;i++)

{

names[i]=graph[i].name;

}

**for**(**int** i=0;i<graph.length;i++)

{

arcnode pre = **null**;

arcnode p =graph[i].firstarc;

**while**(p!=**null**)

{

names[i]=names[i]+" "+graph[p.ver].name+" "+p.weight;

p=p.nextarc;

}

}

String s="";

**for**(**int** i=0;i<graph.length;i++)

s+=names[i]+"\n";

**return** s;

}

}

**class** arc{

String name1=**null**;

String name2=**null**;

**int** weight;

arc(String name1,String name2,**int** weight){

**this**.name1=name1;

**this**.name2=name2;

**this**.weight=weight;

}

}

五、实验结果

