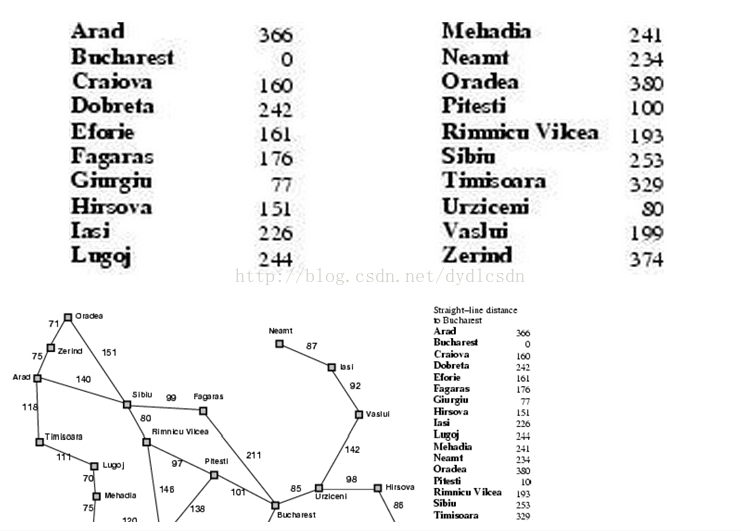
**Uniformed Search实现**

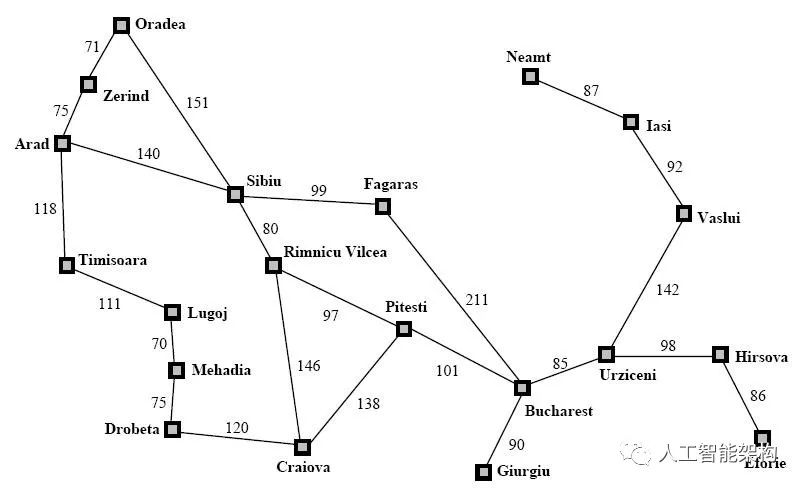
**171491125 吴俊**

1. **问题的提出：**

**通过使用Uniformed Search 得到从起始点Arad到目标点Bucharest的一条路径，即为罗马尼亚问题的一个解，在求解的过程中记录每种算法得到的解，即输出每种解得到的条路径。**

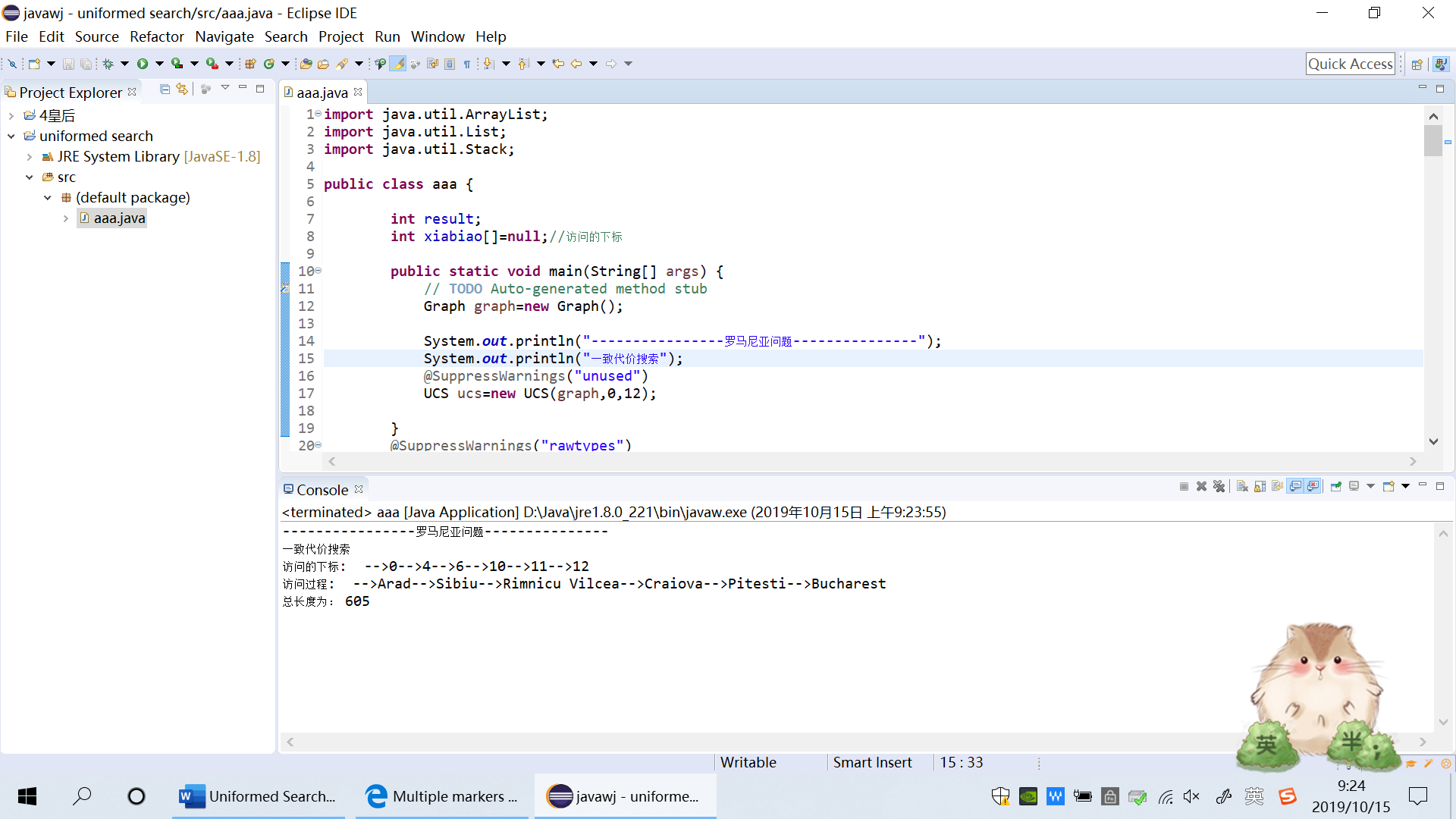
1. **罗马尼亚地图：**

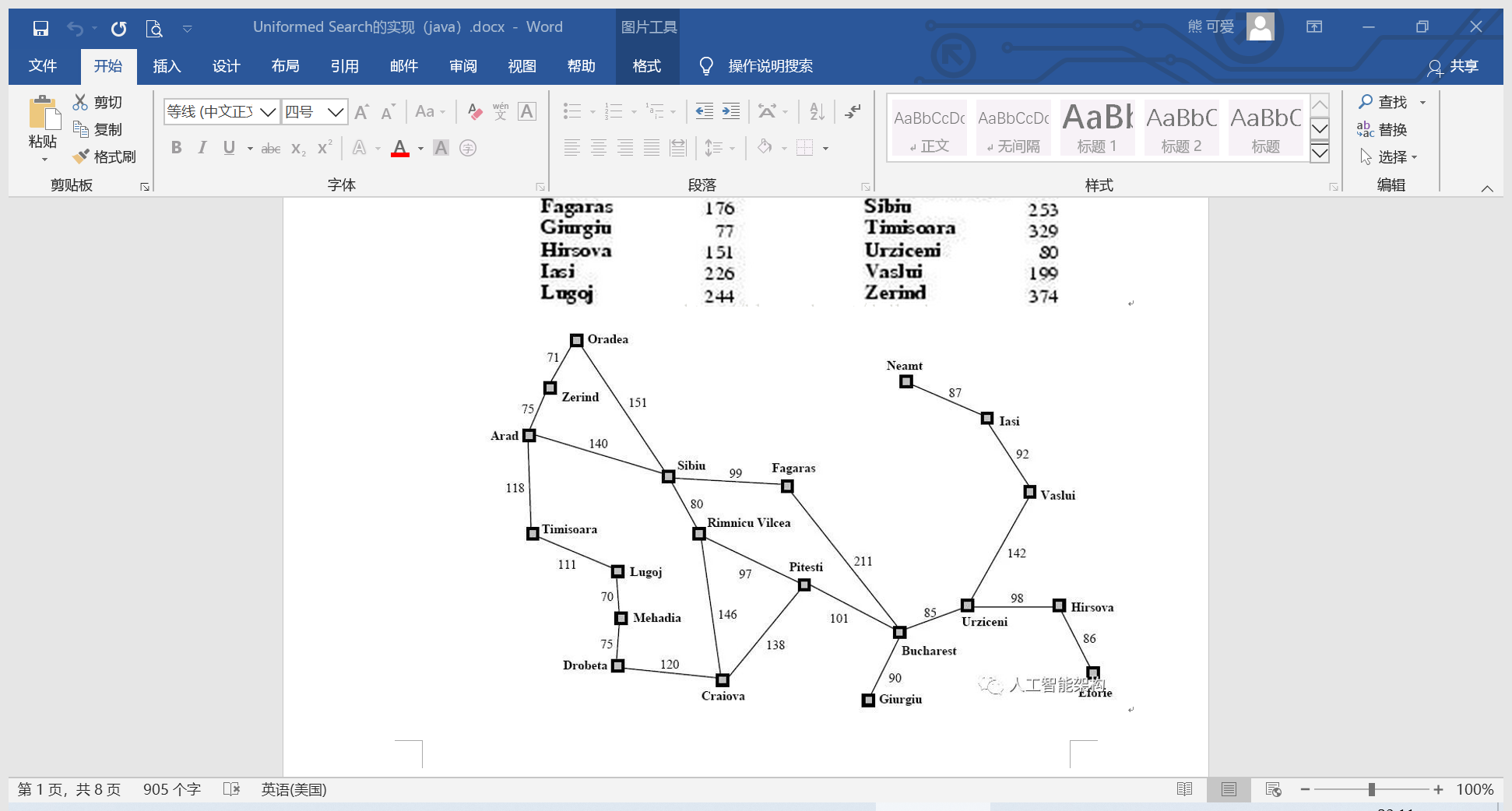
****

****

1. **实验内容**

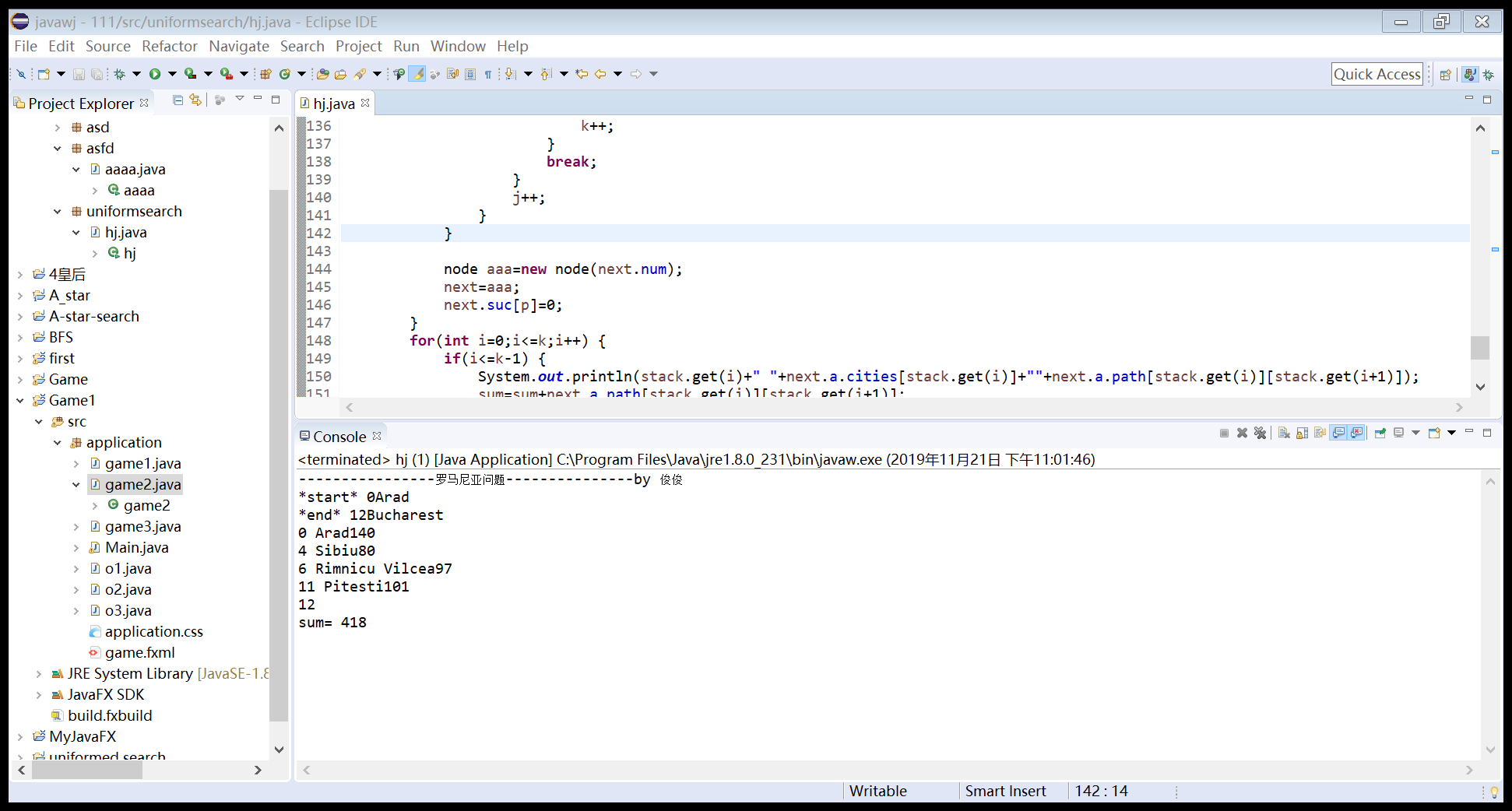
在网络上找到的一致代价搜索，运行结果是错误的。如图:在Rimnicu Vilcea和Pitesti中多出了一个Craiova，这显然是错误的。

****



所以我对这个算法进行了改进。

改进结果为：



1. **实验代码**

**package uniformsearch;**

**import java.util.Stack;**

**public class hj {**

**int result;**

**int xiabiao[]=null;//访问的下标**

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

**// TODO Auto-generated method stub**

**node start=new node(0);**

**node end=new node(12);**

**System.out.println("----------------罗马尼亚问题---------------by 俊俊");**

**ucs(start,end);**

**}**

**public static class Graph{**

**public int path[][]=new int[][]{**

**{0,75,10000,118,140,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000},**

**{75,0,71,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000},**

**{10000,71,0,10000,151,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000},**

**{118,10000,10000,0,10000,111,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000},**

**{140,10000,151,10000,0,10000,80,99,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000},**

**{10000,10000,10000,111,10000,0,10000,10000,70,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000},**

**{10000,10000,10000,10000,80,10000,0,10000,10000,10000,146,97,10000,10000,10000,10000,10000,10000,10000,10000},**

**{10000,10000,10000,10000,99,10000,10000,0,10000,10000,10000,10000,211,10000,10000,10000,10000,10000,10000,10000},**

**{10000,10000,10000,10000,10000,70,10000,10000,0,75,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000},**

**{10000,10000,10000,10000,10000,10000,10000,10000,75,0,120,10000,10000,10000,10000,10000,10000,10000,10000,10000},**

**{10000,10000,10000,10000,10000,10000,146,10000,10000,120,0,138,10000,10000,10000,10000,10000,10000,10000,10000},**

**{10000,10000,10000,10000,10000,10000,97,10000,10000,10000,138,0,101,10000,10000,10000,10000,10000,10000,10000},**

**{10000,10000,10000,10000,10000,10000,10000,211,10000,10000,10000,101,0,90,85,10000,10000,10000,10000,10000},**

**{10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,90,0,10000,10000,10000,10000,10000,10000},**

**{10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,85,10000,0,98,10000,142,10000,10000},**

**{10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,98,0,86,10000,10000,10000},**

**{10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,86,0,10000,10000,10000},**

**{10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,142,10000,10000,0,92,10000},**

**{10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,92,0,87},**

**{10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,10000,87,0}**

**};**

**public int[]h=new int[]{516,524,530,479,403,394,343,326,391,392,310,160,150,155,100,0};**

**public String[] cities=new String[]{**

**"Arad","Zerind","Oradea","Timisoara",**

**"Sibiu","Lugoj","Rimnicu Vilcea",**

**"Fagaras","Mehadia","Drobeta","Craiova",**

**"Pitesti","Bucharest","Giurgiu","Urziceni",**

**"Hirsova","Eforie","Vaslui","Isi","Neamt"**

**};//城市名**

**}**

**public static class node{**

**int num;**

**int[] suc=new int[20];**

**Graph a=new Graph();**

**node(int num)**

**{**

**this.num=num;**

**for(int i =0; i < 20; i++) {**

**if(a.path[num][i]>0&&a.path[num][i]<10000)**

**suc[i]=1;**

**else**

**suc[i]=0;**

**}**

**}**

**}**

**public static void ucs(node start, node end) {**

**// TODO Auto-generated method stub**

**int length[]=new int [20];**

**int min;**

**int p=0;**

**int sum=0;**

**int sum1=0;**

**int k=0;**

**int l;**

**System.out.println("\*start\*"+" "+start.num+start.a.cities[start.num]);**

**System.out.println("\*end\*"+" "+end.num+end.a.cities[end.num]);**

**Stack<Integer> stack=new Stack<Integer>();**

**stack.push(start.num);**

**node next=start;**

**while(next.num!=end.num) {**

**p=next.num;**

**for(int i=0;i<20;i++) {**

**if(next.suc[i]!=0)**

**length[i]=next.a.path[next.num][i];**

**else**

**length[i]=10000;**

**}**

**min=length[0];**

**for(int i=0;i<20;i++)**

**{**

**if(length[i]<min) {**

**min=length[i];**

**next.num=i;**

**}**

**}**

**stack.push(next.num);**

**k++;**

**if(k>1) {**

**int j=0;**

**while(j<k-1){**

**node emp=new node(0);**

**emp.num=stack.get(j);**

**node recorder=new node(stack.get(k));**

**if(emp.suc[stack.get(k)]==1) {**

**l=emp.a.path[j][stack.get(k)];**

**for(int i=j;i<k;i++) {**

**sum1=sum1+emp.a.path[stack.get(i)][stack.get(i+1)];**

**}**

**if(l<sum1){**

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

**stack.pop();**

**}**

**stack.push(recorder.num);**

**k++;**

**}**

**break;**

**}**

**j++;**

**}**

**}**

**node aaa=new node(next.num);**

**next=aaa;**

**next.suc[p]=0;**

**}**

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

**if(i<=k-1) {**

**System.out.println(stack.get(i)+" "+next.a.cities[stack.get(i)]+""+next.a.path[stack.get(i)][stack.get(i+1)]);**

**sum=sum+next.a.path[stack.get(i)][stack.get(i+1)];**

**}**

**else**

**{**

**System.out.println(stack.get(i));**

**System.out.println("sum= "+sum);**

**}**

**}**

**}**

**}**