A\*算法的代码实现

实验语言：java

实验过程：通过学习课本上的A\*算法，结合课本上 的实例实现了a\*算法。

实验代码：

package artificialIntelligence;

import java.util.Scanner;

public class AStar {

static int g=0;//R,L,T,B

static int[] h=new int[4];

static int[] f=new int[4];

static int[][] objective=new int[3][3];

static int[][] realbegin=new int[3][3];

public static void main(String[] args)

{

AStar aStar=new AStar();

//int findzero[][]=aStar.creatF();

//int diffrence[][]=aStar.creatD();

realbegin=aStar.evalution(realbegin,new String("初始图"));

objective=aStar.evalution(objective,new String("目标图"));

aStar.exchnageRight();

aStar.exchnageLeft();

aStar.select();

}

public int abso(int a,int b) {//求绝对值

int c;

c=a-b;

if(c<0)

return -c;

else

return c;

}

public void evalotionH() {

for(int i=0;i<4;i++) {

h[i]=1000;

}

}

public int[][] creatF( )//建立空白点储存数组

{

int[][] a=new int[1][2];

return a;

}

public int[][] creatD( ) //建立不同点的行列数储存数组

{

int[][] a=new int[4][9];

return a;

}

public int[][] creatA( ) //建立图的存储数组

{

int[][] a=new int[3][3];

return a;

}

public int[][] evalution(int[][] what,String name)//对图赋值

{

Scanner read=new Scanner(System.in);

System.out.println(name);

for(int i=0;i<3;i++)

{

for(int j=0;j<3;j++)

{

System.out.println("请输入第"+i+"行第"+j+"列：");

what[i][j]=read.nextInt();

}

}

return what;

}

public int[][] findZero(int[][] begin,int[][] findzero)//找出空白处并记录行列数

{

for(int i=0;i<3;i++)

{

for(int j=0;j<3;j++)

{

if(begin[i][j]==0)

{

findzero[0][0]=i;

findzero[0][1]=j;

i=2;

}

}

}

return findzero;

}

public int Compare(int[][] begin,int[][] objective,int[][] diffrence)

{//找出初始图与目标图的不同点并记录行列数

int cloumn=0;

for(int i=0;i<3;i++)

{

for(int j=0;j<3;j++) {

for(int k=0;k<3;k++)

{

for(int l=0;l<3;l++)

{

if(begin[i][j]==objective[k][l]&&(i!=k||j!=l)&&begin[i][j]!=0)

{

diffrence[0][cloumn]=i;//br

diffrence[1][cloumn]=j;//bc

diffrence[2][cloumn]=k;//or

diffrence[3][cloumn]=l;//oc

cloumn++;

}

}

}

}

}

return cloumn;

}

public int calculateH(int[][] deference,int clomun)

{

int h=0;

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

{

h=(h+abso(deference[2][i],deference[0][i])+abso(deference[3][i],deference[1][i]));

//System.out.println(h);

}

return h;

}

public void exchange(int[][] begin,int x,int y,int m,int n)//节点移动

{

int a;

int b;

a=begin[x][y];

b=begin[m][n];

begin[m][n]=a;

begin[x][y]=b;

}

public int[][] exchnageRight() //检查是否可向右交换并交换

{

if(findzero[0][1]+1<3)

{

//System.out.println(find[0][0]+" "+find[0][1]);

int[][] beginRight;

beginRight=new int[3][3];

for(int i=0;i<3;i++) {

for(int j=0;j<3;j++)

beginRight[i][j]=realbegin[i][j];

}

exchange(beginRight, findzero[0][0], findzero[0][1], findzero[0][0],( findzero[0][1]+1));

int[][] deference=creatD();

int clomun=Compare(beginRight, objective, deference);

System.out.println();

h[0]=calculateH(deference, clomun);

//System.out.println(h[0]);

return beginRight;

}

return realbegin;

}

public int[][] exchnageLeft() //检查是否可向左交换并交换

{

if(findzero[0][1]-1>=0)

{

int[][] beginLeft;

beginLeft=new int[3][3];

for(int i=0;i<3;i++) {

for(int j=0;j<3;j++)

beginLeft[i][j]=realbegin[i][j];

}

exchange(beginLeft, findzero[0][0], findzero[0][1], findzero[0][0], (findzero[0][1]-1));

int[][] deference=creatD();

int clomun=Compare(beginLeft, objective, deference);

h[1]=calculateH(deference, clomun);

return beginLeft;

}

return realbegin;

}

public int[][] exchnageTop(int[][] realbegin,int[][] objective) //检查是否可向上交换并交换

{

if(findzero[0][0]-1>=0)

{

int beginTop[][];

beginTop=new int[3][3];

for(int i=0;i<3;i++) {

for(int j=0;j<3;j++)

beginTop[i][j]=realbegin[i][j];

}

exchange(beginTop, findzero[0][0], findzero[0][1], (findzero[0][0]-1), findzero[0][1]);

int[][] deference=creatD();

int clomun=Compare(beginTop, objective, deference);

h[2]=calculateH(deference, clomun);

return beginTop;

}

return realbegin;

}

public int[][] exchnageBottom(int[][] realbegin,int[][] objective)//检查是否可向下交换并交换

{

if(findzero[0][0]+1<3)

{

int beginBottom[][];

beginBottom=new int[3][3];

for(int i=0;i<3;i++) {

for(int j=0;j<3;j++)

beginBottom[i][j]=realbegin[i][j];

}

exchange(beginBottom, findzero[0][0], findzero[0][1],(findzero[0][0]+1), findzero[0][1]);

int[][] deference=creatD();

int clomun=Compare(beginBottom, objective, deference);

h[3]=calculateH(deference, clomun);

return beginBottom;

}

return realbegin;

}

public void calculateF() {

for(int i=0;i<4;i++) {

f[i]=g+h[i];

}

}

public int min() {

int mark;

if(f[0]<f[1]&&f[0]<f[2]&&f[0]<f[3])

mark=0;

else if(f[1]<f[0]&&f[1]<f[2]&&f[1]<f[3])

mark=1;

else if(f[2]<f[0]&&f[2]<f[1]&&f[2]<f[3])

mark=2;

else mark=3;

return mark;

}

static int[][] findzero=new int[1][2];

public void select()

{

evalotionH();

findZero(realbegin, findzero);

int[][] beginBottom=exchnageBottom(realbegin,objective);

int[][] beginRight=exchnageRight();

int[][] beginLeft=exchnageLeft();

int[][] beginTop=exchnageTop(realbegin,objective);

calculateF();

int mark=min();

if(mark==0)

{

realbegin=beginRight;

System.out.println("右");

}

else if(mark==1)

{

realbegin=beginLeft;

System.out.println("左");

}

else if(mark==2)

{

realbegin=beginTop;

System.out.println("顶");

}

else

{

realbegin=beginBottom;

System.out.println("底");

}

g++;

if(h[0]==0||h[1]==0||h[2]==0||h[3]==0)

{

System.out.println(g);

}

else {

select();

}

}

}

实验结果：

