八数码问题

171491106 安南 硬件一班

一、实验题目

八数码问题

二、算法简介

A-Star是一种静态路网中求解最短路最有效的直接搜索方法，估价值跟实例值非常接近。A-Star在理论上是时间最优的，但也有缺点:它的空间增长是指数级别的(格子越多计算的量也就越大)，A-Star的主要特点是通过从起始点开始,检查相邻方格的方式,向外扩展，直到找到目标。

三、实验目的

在3×3的方格棋盘上，摆放着1到8这八个数码，有1个方格是空的，其初始状态如图1所示，要求对空格执行空格左移、空格右移、空格上移和空格下移这四个操作使得棋盘从初始状态到目标状态。

四、实验代码

**package** 八数码;

**import** java.io.BufferedReader;

**import** java.io.FileNotFoundException;

**import** java.io.FileReader;

**import** java.io.IOException;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.Collections;

**import** java.util.Scanner;

@SuppressWarnings("rawtypes")

**public** **class** EightPuzzle **implements** Comparable{

**private** **int**[] num = **new** **int**[9];

**private** **int** depth; //当前的深度即走到当前状态的步骤

**private** **int** evaluation; //从起始状态到目标的最小估计值

**private** **int** misposition; //到目标的最小估计

**private** EightPuzzle parent; //当前状态的父状态

**public** **int**[] getNum() {

**return** num;

}

**public** **void** setNum(**int**[] num) {

**this**.num = num;

}

**public** **int** getDepth() {

**return** depth;

}

**public** **void** setDepth(**int** depth) {

**this**.depth = depth;

}

**public** **int** getEvaluation() {

**return** evaluation;

}

**public** **void** setEvaluation(**int** evaluation) {

**this**.evaluation = evaluation;

}

**public** **int** getMisposition() {

**return** misposition;

}

**public** **void** setMisposition(**int** misposition) {

**this**.misposition = misposition;

}

**public** EightPuzzle getParent() {

**return** parent;

}

**public** **void** setParent(EightPuzzle parent) {

**this**.parent = parent;

}

**public** **boolean** isTarget(EightPuzzle target){

**return** Arrays.*equals*(getNum(), target.getNum());

}

**public** **void** init(EightPuzzle target){

**int** temp = 0;

**for**(**int** i=0;i<9;i++){

**if**(num[i]!=target.getNum()[i])

temp++;

}

**this**.setMisposition(temp);

**if**(**this**.getParent()==**null**){

**this**.setDepth(0);

}**else**{

**this**.depth = **this**.parent.getDepth()+1;

}

**this**.setEvaluation(**this**.getDepth()+**this**.getMisposition());

}

**public** **boolean** isSolvable(EightPuzzle target){

**int** reverse = 0;

**for**(**int** i=0;i<9;i++){

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

**if**(num[j]>num[i])

reverse++;

**if**(target.getNum()[j]>target.getNum()[i])

reverse++;

}

}

**if**(reverse % 2 == 0)

**return** **true**;

**return** **false**;

}

@Override

**public** **int** compareTo(Object o) {

EightPuzzle c = (EightPuzzle) o;

**return** **this**.evaluation-c.getEvaluation();//默认排序为f(n)由小到大排序

}

**public** **int** getZeroPosition(){

**int** position = -1;

**for**(**int** i=0;i<9;i++){

**if**(**this**.num[i] == 0){

position = i;

}

}

**return** position;

}

**public** **int** isContains(ArrayList<EightPuzzle> open){

**for**(**int** i=0;i<open.size();i++){

**if**(Arrays.*equals*(open.get(i).getNum(), getNum())){

**return** i;

}

}

**return** -1;

}

**public** **boolean** isMoveUp() {

**int** position = getZeroPosition();

**if**(position<=2){

**return** **false**;

}

**return** **true**;

}

**public** **boolean** isMoveDown() {

**int** position = getZeroPosition();

**if**(position>=6){

**return** **false**;

}

**return** **true**;

}

**public** **boolean** isMoveLeft() {

**int** position = getZeroPosition();

**if**(position%3 == 0){

**return** **false**;

}

**return** **true**;

}

**public** **boolean** isMoveRight() {

**int** position = getZeroPosition();

**if**((position)%3 == 2){

**return** **false**;

}

**return** **true**;

}

**public** EightPuzzle moveUp(**int** move){

EightPuzzle temp = **new** EightPuzzle();

**int**[] tempnum = (**int**[])num.clone();

temp.setNum(tempnum);

**int** position = getZeroPosition(); //0的位置

**int** p=0; //与0换位置的位置

**switch**(move){

**case** 0:

p = position-3;

temp.getNum()[position] = num[p];

**break**;

**case** 1:

p = position+3;

temp.getNum()[position] = num[p];

**break**;

**case** 2:

p = position-1;

temp.getNum()[position] = num[p];

**break**;

**case** 3:

p = position+1;

temp.getNum()[position] = num[p];

**break**;

}

temp.getNum()[p] = 0;

**return** temp;

}

**public** **void** print(){

**for**(**int** i=0;i<9;i++){

**if**(i%3 == 2){

System.***out***.println(**this**.num[i]);

}**else**{

System.***out***.print(**this**.num[i]+" ");

}

}

}

**public** **void** printRoute(){

EightPuzzle temp = **null**;

**int** count = 0;

temp = **this**;

**while**(temp!=**null**){

temp.print();

System.***out***.println("----------分割线----------");

temp = temp.getParent();

count++;

}

System.***out***.println("步骤数："+(count-1));

}

**public** **void** operation(ArrayList<EightPuzzle> open,ArrayList<EightPuzzle> close,EightPuzzle parent,EightPuzzle target){

**if**(**this**.isContains(close) == -1){

**int** position = **this**.isContains(open);

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

**this**.parent = parent;

**this**.init(target);

open.add(**this**);

}**else**{

**if**(**this**.getDepth() < open.get(position).getDepth()){

open.remove(position);

**this**.parent = parent;

**this**.init(target);

open.add(**this**);

}

}

}

}

@SuppressWarnings("unchecked")

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

//定义open表

ArrayList<EightPuzzle> open = **new** ArrayList<EightPuzzle>();

ArrayList<EightPuzzle> close = **new** ArrayList<EightPuzzle>();

EightPuzzle start = **new** EightPuzzle();

EightPuzzle target = **new** EightPuzzle();

//BufferedReader br = new BufferedReader(new FileReader("./input.txt") );

String lineContent = **null**;

**int** stnum[] = {2,1,6,4,0,8,7,5,3};

**int** tanum[] = {1,2,3,8,0,4,7,6,5};

**int** order = 0;

**try** {

BufferedReader br;

br = **new** BufferedReader(**new** FileReader("input.txt") );

**while**((lineContent=br.readLine())!=**null**){

String[] str = lineContent.split(",");

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

**if**(order==0)

stnum[i] = Integer.*parseInt*(str[i]);

**else**

tanum[i] = Integer.*parseInt*(str[i]);

}

order++;

}

} **catch** (NumberFormatException e) {

System.***out***.println("请检查输入文件的格式，例如：2,1,6,4,0,8,7,5,3 换行 1,2,3,8,0,4,7,6,5");

e.printStackTrace();

} **catch** (IOException e) {

System.***out***.println("当前目录下无input.txt文件。");

e.printStackTrace();

}

start.setNum(stnum);

target.setNum(tanum);

**long** startTime=System.*currentTimeMillis*(); //获取开始时间

**if**(start.isSolvable(target)){

//初始化初始状态

start.init(target);

open.add(start);

**while**(open.isEmpty() == **false**){

Collections.*sort*(open); //按照evaluation的值排序

EightPuzzle best = open.get(0); //从open表中取出最小估值的状态并移除open表

open.remove(0);

close.add(best);

**if**(best.isTarget(target)){

//输出

best.printRoute();

**long** end=System.*currentTimeMillis*(); //获取结束时间

System.***out***.println("程序运行时间： "+(end-startTime)+"ms");

System.*exit*(0);

}

**int** move;

//由best状态进行扩展并加入到open表中

//0的位置上移之后状态不在close和open中设定best为其父状态，并初始化f(n)估值函数

**if**(best.isMoveUp()){

move = 0;

EightPuzzle up = best.moveUp(move);

up.operation(open, close, best, target);

}

//0的位置下移之后状态不在close和open中设定best为其父状态，并初始化f(n)估值函数

**if**(best.isMoveDown()){

move = 1;

EightPuzzle up = best.moveUp(move);

up.operation(open, close, best, target);

}

//0的位置左移之后状态不在close和open中设定best为其父状态，并初始化f(n)估值函数

**if**(best.isMoveLeft()){

move = 2;

EightPuzzle up = best.moveUp(move);

up.operation(open, close, best, target);

}

//0的位置右移之后状态不在close和open中设定best为其父状态，并初始化f(n)估值函数

**if**(best.isMoveRight()){

move = 3;

EightPuzzle up = best.moveUp(move);

up.operation(open, close, best, target);

}

}

}**else**

System.***out***.println("没有解，请重新输入。");

}

}

五、实验结果

