**A\*算法分析八数码问题**

Tip:请使用文档结构图查看

A\*(A-Star)算法是一种静态路网中求解最短路最有效的直接[搜索](https://baike.baidu.com/item/%E6%90%9C%E7%B4%A2/5395421" \t "_blank)方法，也是许多其他[问题](https://baike.baidu.com/item/%E9%97%AE%E9%A2%98/13018756" \t "_blank)的常用启发式算法。注意——是最有效的**直接**搜索算法，之后涌现了很多预处理算法（如ALT，CH，HL等等），在线查询效率是A\*算法的数千甚至上万倍。

公式表示为： f(n)=g(n)+h(n),其中:

f(n) 是从初始[状态](https://baike.baidu.com/item/%E7%8A%B6%E6%80%81/5809342" \t "_blank)经由状态n到目标状态的代价估计，

g(n) 是在[状态空间](https://baike.baidu.com/item/%E7%8A%B6%E6%80%81%E7%A9%BA%E9%97%B4" \t "_blank)中从初始状态到[状态](https://baike.baidu.com/item/%E7%8A%B6%E6%80%81/5809342" \t "_blank)n的实际代价，

h(n) 是从状态n到目标状态的最佳路径的估计代价。

（对于路径搜索问题，状态就是图中的节点，代价就是距离）

# A\*算法

保证找到[最短路径](https://baike.baidu.com/item/%E6%9C%80%E7%9F%AD%E8%B7%AF%E5%BE%84" \t "_blank)（最优解的）条件，关键在于估价函数f(n)的选取（或者说h(n)的选取）。

我们以d(n)表达状态n到目标状态的距离，那么h(n)的选取大致有如下三种情况：

1. 如果h(n)< d(n)到目标状态的实际距离，这种[情况](https://baike.baidu.com/item/%E6%83%85%E5%86%B5" \t "_blank)下，搜索的点数多，搜索范围大，效率低。但能得到最优解。
2. 如果h(n)=d(n)，即距离估计h(n)等于最短距离，那么搜索将严格沿着最短路径进行， 此时的搜索效率是最高的。
3. 如果 h(n)>d(n)，搜索的点数少，搜索范围小，效率高，但不能保证得到最优解。

1.在GRAPHSEARCH过程中，如果第8步的重排OPEN表是依据f(n)=g(n)+h(n)进行的，则称该过程为A算法。在A算法中，如果对所有的n存在h(n)≤h\*(n),则称h(n)为h\*(n)的下界，它表示某种偏于保守的估计。采用h\*(n)的下界h(n)为启发函数的A算法，称为A\*算法。当h=0时，A\*算法就变为有序搜索算法。  
  在A算法中，如果满足条件：

  (1) g(n)是对g\*(n)的估计，且g(n)>0；

  (2) h(n)是h\*(n)的下界，即对任意节点n均有0≤h(n)≤h\*(n)则A算法称为A\*算法  
  A\*算法的可纳性，对任一个图，存在从S到目标的路径，如果一个搜索算法总是结束在一条从S到目标的最佳路径上，则称此算法是可采纳的。算法A\*保证只要最短路径存在，就一定能找出这条路径，所以算法A\*是可纳的。  
  估价函数：f(n)=d(n)+w(n)

  其中：d(n)为n的深度 w(n)为不在位的棋子数

      取h(n)=w(n),则有w(n)≤h\*(n),h(n)满足A\*算法的限制条件。   
  在八数码难题中, 令估价函数

    f(n)=d(n)+p(n)

  启发函数h(n)=p(n),p(n)为不在位的棋子与其目标位置的距离之和，则有p(n)≤h\*(n),满足A\*算法的限制条件。    
  w(n)——不在位的棋子数，不够贴切，错误选用节点加以扩展。  
  更接近于h\*(n)的h(n)，其值是节点n与目标状态节点相比较，每个错位棋子在假设不受阻拦的情况下，移动到目标状态相应位置所需走步的总和。(n)比w(n)更接近于h\*(n)，因为p(n)不仅考虑了错位因素，还考虑了错位的距离（移动次数）。  
  说明h值越大,启发功能越强, 搜索效率越高.特别地

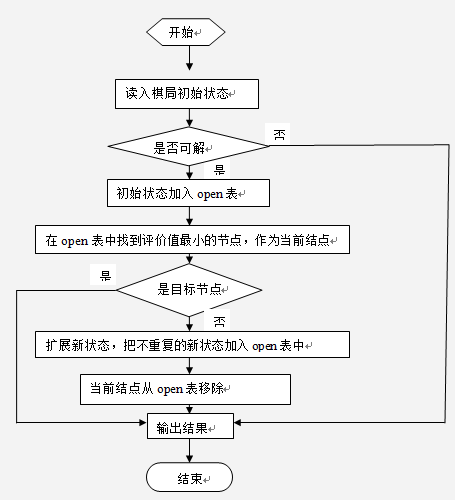
  (1)h(n)=h\*(n)

    搜索仅沿最佳路径进行, 效率最高.  
 (2)h(n)=0

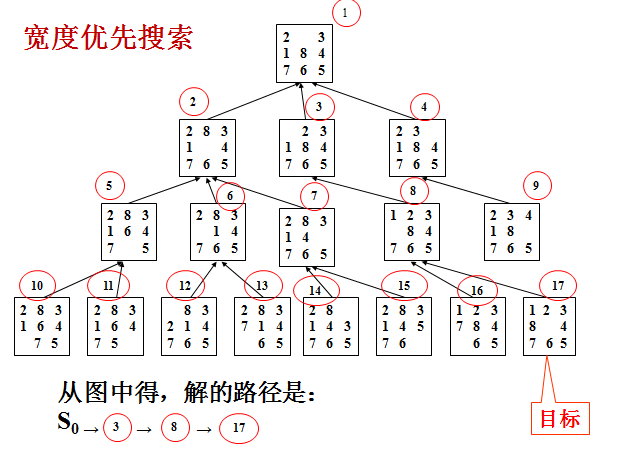
无启发信息, 盲目搜索, 效率低.  
 (3)h(n)>h\*(n)

常用的状态空间搜索有深度优先和广度优先。广度优先是从初始状态一层一层向下找，直到找到目标为止。深度优先是按照一定的顺序前查找完一个分支，再查找另一个分支，以至找到目标为止。广度和深度优先搜索有一个很大的缺陷就是他们都是在一个给定的状态空间中穷举。这在状态空间不大的情况下是很合适的算法，可是当状态空间十分大，且不预测的情况下就不可取了。他的效率实在太低，甚至不可完成。由于八数码问题状态空间共有9!个状态，对于八数码问题如果选定了初始状态和目标状态，有9!/2个状态要搜索，考虑到时间和空间的限制，在这里采用A\*算法作为搜索策略。在这里就要用到启发式搜索  
  启发式搜索就是在状态空间中的搜索对每一个搜索的位置进行评估，得到最好的位置，再从这个位置进行搜索直到目标。这样可以省略大量无畏的搜索路径，提到了效率。在启发式搜索中，对位置的估价是十分重要的。采用了不同的估价可以有不同的效果。  
  启发中的估价是用估价函数表示的，如：f(n) = g(n) +h(n)其中f(n) 是节点n的估价函数，g(n)是在状态空间中从初始节点到n节点的实际代价，h(n)是从n到目标节点最佳路径的估计代价。 在此八数码问题中，显然g(n)就是从初始状态变换到当前状态所移动的步数，估计函数f(n)我们就可采用当前状态各个数字牌不在目标状态未知的个数，即错位数。

流程图如下：



# 宽度优先搜索

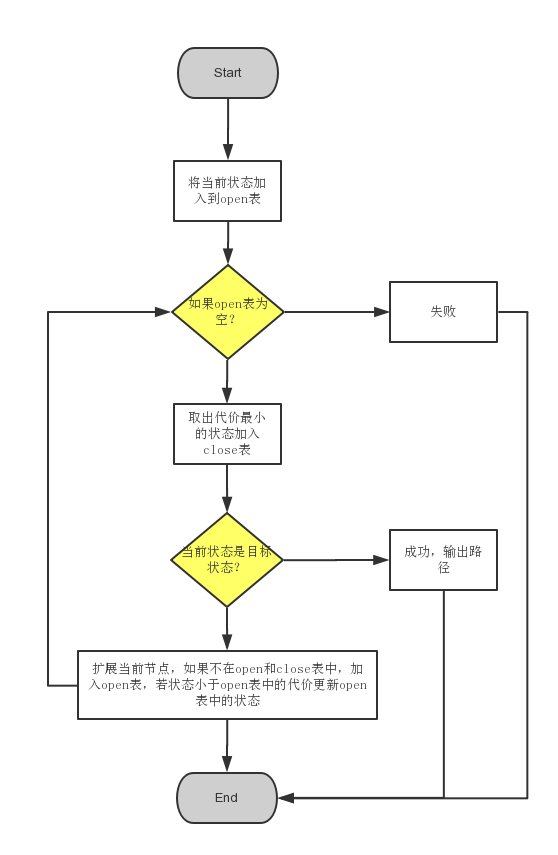


# 八数码问题分析

1.八数码问题，移动数字就是相当于移动空格。这样就将问题简化为空格的移动，空格移动的状态只有4种：上、下、左、右。然而在八数码问题中并不是每次空格的移动都有四种状态，我们要判断在当前位置也移动的状态才能移动，我们还要去掉一种状态就是当前状态的父状态，因为如果我们移动到父状态则相当于回退了一步。

2.然后，我们要关心的就是给定的初始化状态是否能够通过移动而达到目标状态。这就涉及到了数学问题，就是如果初始状态和目标状态的逆序值同为奇数或同为偶数则可以通过有限次数的移动到达目标状态，否则无解。

工作流程图：



# 算法实现：

我是使用JAVA代码来实现的。我接下来会把迷宫比作棋盘，里面的数字比作棋子，这样更方便描述。我在代码同一目录下写了input.txt文件，这样我每次在input.txt里面修改初始和目标状态就可以了。不需要在每个代码里面更改。我把迷宫转换成一维数组来运算的，比如1,2,3,4,5,6,7,8,0就代表图(a) 1,2,3,8,0,4,7,6,5代表图(b)

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 0 |

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 8 | 0 | 4 |
| 7 | 6 | 5 |

( a ) ( b )

首先定义：

f(n):估计从初始状态到目标状态的代价。

　　g(n):从初始状态到当前状态的实际代价。

h(n):三种代码的估计函数不一样，在最后会进行比较

三个代码里都用到了都重写了compareTo函数，这样就保open表的排序是从f(n)从小到大的顺序

## 算法一：估价函数h(n)为当前状态与目标状态的错位数

h(n)会一直减小，最后直到h(n)为0，此时就找到了目标节点

代码：

**import** java**.**io**.**BufferedReader**;**

**import** java**.**io**.**FileReader**;**

**import** java**.**io**.**IOException**;**

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**Arrays**;**

**import** java**.**util**.**Collections**;**

@SuppressWarnings**(**"rawtypes"**)**

public class EightFigure1 **implements** Comparable**{**

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

//这里的g为深度，h为各棋子不在正确位置的数目

//当然h也可以设置为个棋子与目标位置的距离，但是太麻烦

private int g**;** //当前的深度即走到当前状态的步骤

private int f**;** //从起始状态到目标的最小估计值

private int h**;** //到目标的最小估计

private EightFigure1 parent**;** //当前状态的父状态

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

**return** num**;**

**}**

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

**this.**num **=** num**;**

**}**

public int getg**()** **{**

**return** g**;**

**}**

public void setg**(**int g**)** **{**

**this.**g **=** g**;**

System**.**out**.**print**(**"g(n)="**+**g**+**" "**);**

**}**

public int getf**()** **{**

**return** f**;**

**}**

public void setf**(**int f**)** **{**

**this.**f **=** f**;**

System**.**out**.**print**(**"f(n)="**+**f**+**" "**);**

**}**

public int geth**()** **{**

**return** h**;**

**}**

public void seth**(**int h**)** **{**

**this.**h **=** h**;**

System**.**out**.**print**(**"h(n)="**+**h**+**" "**);**

**}**

public EightFigure1 getParent**()** **{**

**return** parent**;**

**}**

public void setParent**(**EightFigure1 parent**)** **{**

**this.**parent **=** parent**;**

**}**

/\*\*

\* 判断当前状态是否为目标状态

\* **@param** target

\* **@return**

\*/

public boolean isTarget**(**EightFigure1 target**){**

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

**}**

/\*\*

\* 求f(n) = g(n)+h(n);

\* 初始化状态信息

\* **@param** target

\*/

public void init**(**EightFigure1 target**){**

int temp **=** 0**;**

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

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

temp**++;**

**}**

**this.**seth**(**temp**);**

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

**this.**setg**(**0**);**//如果没有父节点

**}else{**

**this.**g **=** **this.**parent**.**getg**()+**1**;**

setg**(**g**);**

**}**

**this.**setf**(this.**getg**()+this.**geth**());**

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

**}**

/\*\*

\* 数学问题如果初始状态和目标状态的逆序值同为奇数或同为偶数

\* 则可以通过有限次数的移动到达目标状态，否则无解。

\* @TODO求逆序值并判断是否有解

\* **@param** target

\* **@return** 有解：true 无解：false

\*/

public boolean isSolvable**(**EightFigure1 target**){**

int reverse **=** 0**;**

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

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

//这里的this就是start，因为用了start.isSolvable(target)

**if(this.**num**[**j**]>this.**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**)** **{**

EightFigure1 c **=** **(**EightFigure1**)** o**;**

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

**}**

/\*\*

\* **@return** 返回0在八数码中的位置

\*/

public int getZeroPosition**(){**

int position **=** **-**1**;**

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

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

position **=** i**;**

**}**

**}**

**return** position**;**

**}**

/\*\*

\*

\* **@param** open 状态集合

\* **@return** 判断当前状态是否存在于open表中

\*/

public int isContains**(**ArrayList**<**EightFigure1**>** open**){**

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

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

**return** i**;**

**}**

**}**

**return** **-**1**;**

**}**

/\*\*

\*

\* **@return** 小于3的不能上移返回false

\*/

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

int position **=** getZeroPosition**();**

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

**return** **false;**

**}**

**return** **true;**

**}**

/\*\*

\*

\* **@return** 大于6返回false

\*/

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

int position **=** getZeroPosition**();**

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

**return** **false;**

**}**

**return** **true;**

**}**

/\*\*

\* 是否可以左移，如果空格(0)的位置在

\* 数组的0，3，6就不行，及在最左边一排

\* **@return** 0，3，6返回false

\*/

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

int position **=** getZeroPosition**();**

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

**return** **false;**

**}**

**return** **true;**

**}**

/\*\*

\*

\* **@return** 2，5，8不能右移返回false

\*/

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

int position **=** getZeroPosition**();**

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

**return** **false;**

**}**

**return** **true;**

**}**

/\*\*

\* **@param** move 0：上，1：下，2：左，3：右

\* **@return** 返回移动后的状态

\*/

public EightFigure1 Move**(**int choice**){**

EightFigure1 temp **=** **new** EightFigure1**();**

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(**choice**){**

//0：上，1：下，2：左，3：右:

**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**;**

**}**

/\*\*

\* 按照八数码的格式输出

\*/

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**]+**" "**);**

**}**

**}**

**}**

/\*\*

\* 反序列的输出状态

\*/

public void printRoute**(){**

System**.**out**.**println**(**"因为是反序列输出，请从下往上看"**);**

EightFigure1 temp **=** **null;**

int count **=** 0**;**

temp **=** **this;**

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

temp**.**print**();**

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

temp **=** temp**.**getParent**();**

count**++;**

**}**

System**.**out**.**println**(**"生成节点数："**+(**count**-**1**));**

**}**

/\*\*

\*

\* **@param** open open表

\* **@param** close close表

\* **@param** parent 父状态

\* **@param** target 目标状态

\*/

public void operation**(**ArrayList**<**EightFigure1**>** open**,** ArrayList**<**EightFigure1**>** close**,** EightFigure1 parent**,** EightFigure1 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.**getg**()** **<** open**.**get**(**position**).**getg**()){**

open**.**remove**(**position**);**

**this.**parent **=** parent**;**

**this.**init**(**target**);**

open**.**add**(this);**

**}**

**}**

**}**

**}**

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

//定义open表

ArrayList**<**EightFigure1**>** open **=** **new** ArrayList**<**EightFigure1**>();**

ArrayList**<**EightFigure1**>** close **=** **new** ArrayList**<**EightFigure1**>();**

//创建初始节点和目标节点，这里面包含棋子顺序,f,g,h，指针···属性信息

EightFigure1 start **=** **new** EightFigure1**();**

EightFigure1 target **=** **new** EightFigure1**();**

String lineContent **=** **null;**

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

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

int startnum**[]** **=** **new** int**[**9**];**

int targetnum**[]** **=** **new** int**[**9**];**

int order **=** 0**;**

**try** **{**

BufferedReader br**;**

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

**while((**lineContent**=**br**.**readLine**())!=null){**//读取一行

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

//System.out.println("input.txt每一行个数"+str.length);

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

**if(**order**==**0**)**

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

**else**

targetnum**[**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**(**startnum**);**

target**.**setNum**(**targetnum**);**

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**);** //会按照f的值从小到大排序，因为重写了compareTo()

EightFigure1 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 endTime**=**System**.**currentTimeMillis**();** //获取结束时间

System**.**out**.**println**(**"拓展节点数："**+**close**.**size**()** **);**

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

System**.**out**.**println**((**"转成分钟为："**+(**endTime**-**startTime**)/**60000.0**)+**"min"**);**

System**.**exit**(**0**);**

**}**

//best不是目标节点，继续：

int choice**;**

//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**()){**

choice **=** 0**;**

EightFigure1 temp **=** best**.**Move**(**choice**);**

temp**.**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**()){**

choice **=** 1**;**

EightFigure1 temp **=** best**.**Move**(**choice**);**

temp**.**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**()){**

choice **=** 2**;**

EightFigure1 temp **=** best**.**Move**(**choice**);**

temp**.**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**()){**

choice **=** 3**;**

EightFigure1 temp **=** best**.**Move**(**choice**);**

temp**.**operation**(**open**,** close**,** best**,** target**);**

**}**

**}**

**}else**

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

**}**

**}**

## 算法二: 估价函数h(n)为当前状态与目标状态的差开根号的和

具体解释：因为我是以一维数组来计算的，例如

数组1(前继节点)：a,b,c,d,e 数组2(当前节点)：h,i,j,k,l

此时估价函数就为：h(n)=

使用此估价函数是自己想的，因为这个估价函数满足了逼近的条件，h(n)整体上会越来越小，最后到达0时会停止。

代码：

**import** java**.**io**.**BufferedReader**;**

**import** java**.**io**.**FileReader**;**

**import** java**.**io**.**IOException**;**

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**Arrays**;**

**import** java**.**util**.**Collections**;**

@SuppressWarnings**(**"rawtypes"**)**

public class EightFigure2 **implements** Comparable**{**

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

//这里的g为深度，h为各棋子不在正确位置的数目

//当然h也可以设置为个棋子与目标位置的距离，但是太麻烦

private int g**;** //当前的深度即走到当前状态的步骤

private int f**;** //从起始状态到目标的最小估计值

private int h**;** //到目标的最小估计

private EightFigure2 parent**;** //当前状态的父状态

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

**return** num**;**

**}**

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

**this.**num **=** num**;**

**}**

public int getg**()** **{**

**return** g**;**

**}**

public void setg**(**int g**)** **{**

**this.**g **=** g**;**

System**.**out**.**print**(**"g(n)="**+**g**+**" "**);**

**}**

public int getf**()** **{**

**return** f**;**

**}**

public void setf**(**int f**)** **{**

**this.**f **=** f**;**

System**.**out**.**print**(**"f(n)="**+**f**+**" "**);**

**}**

public int geth**()** **{**

**return** h**;**

**}**

public void seth**(**int h**)** **{**

**this.**h **=** h**;**

System**.**out**.**print**(**"h(n)="**+**h**+**" "**);**

**}**

public EightFigure2 getParent**()** **{**

**return** parent**;**

**}**

public void setParent**(**EightFigure2 parent**)** **{**

**this.**parent **=** parent**;**

**}**

/\*\*

\* 判断当前状态是否为目标状态

\* **@param** target

\* **@return**

\*/

public boolean isTarget**(**EightFigure2 target**){**

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

**}**

/\*\*

\* 求f(n) = g(n)+h(n);

\* 初始化状态信息

\* **@param** target

\*/

public void init**(**EightFigure2 target**){**

int sum **=** 0**;**

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

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

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

int x **=** 0**;**

x**=**Math**.**abs**(**j**-**i**);**

sum **+=** Math**.**pow**(**x**,**1.0**/**4**);**

**}**

**}**

**}**

seth**(**sum**);**

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

**this.**setg**(**0**);**//如果没有父节点

**}else{**

**this.**g **=** **this.**parent**.**getg**()+**1**;**

setg**(**g**);**

**}**

**this.**setf**(this.**getg**()+this.**geth**());**

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

**}**

/\*\*

\* 数学问题如果初始状态和目标状态的逆序值同为奇数或同为偶数

\* 则可以通过有限次数的移动到达目标状态，否则无解。

\* @TODO求逆序值并判断是否有解

\* **@param** target

\* **@return** 有解：true 无解：false

\*/

public boolean isSolvable**(**EightFigure2 target**){**

int reverse **=** 0**;**

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

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

//这里的this就是start，因为用了start.isSolvable(target)

**if(this.**num**[**j**]>this.**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**)** **{**

EightFigure2 t **=** **(**EightFigure2**)** o**;**

**return** **this.**f**-**t**.**getf**();**//默认排序为f(n)由小到大排序

**}**

/\*\*

\* **@return** 返回0在八数码中的位置

\*/

public int getZeroPosition**(){**

int position **=** **-**1**;**

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

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

position **=** i**;**

**}**

**}**

**return** position**;**

**}**

/\*\*

\*

\* **@param** open 状态集合

\* **@return** 判断当前状态是否存在于open表中

\*/

public int isContains**(**ArrayList**<**EightFigure2**>** open**){**

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

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

**return** i**;**

**}**

**}**

**return** **-**1**;**

**}**

/\*\*

\*

\* **@return** 小于3的不能上移返回false

\*/

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

int position **=** getZeroPosition**();**

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

**return** **false;**

**}**

**return** **true;**

**}**

/\*\*

\*

\* **@return** 大于6返回false

\*/

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

int position **=** getZeroPosition**();**

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

**return** **false;**

**}**

**return** **true;**

**}**

/\*\*

\* 是否可以左移，如果空格(0)的位置在

\* 数组的0，3，6就不行，及在最左边一排

\* **@return** 0，3，6返回false

\*/

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

int position **=** getZeroPosition**();**

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

**return** **false;**

**}**

**return** **true;**

**}**

/\*\*

\*

\* **@return** 2，5，8不能右移返回false

\*/

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

int position **=** getZeroPosition**();**

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

**return** **false;**

**}**

**return** **true;**

**}**

/\*\*

\* **@param** move 0：上，1：下，2：左，3：右

\* **@return** 返回移动后的状态

\*/

public EightFigure2 Move**(**int choice**){**

EightFigure2 temp **=** **new** EightFigure2**();**

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(**choice**){**

//0：上，1：下，2：左，3：右:

**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**;**

**}**

/\*\*

\* 按照八数码的格式输出

\*/

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**]+**" "**);**

**}**

**}**

**}**

/\*\*

\* 反序列的输出状态

\*/

public void printRoute**(){**

System**.**out**.**println**(**"因为是反序列输出，请从下往上看"**);**

EightFigure2 temp **=** **null;**

int count **=** 0**;**

temp **=** **this;**

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

temp**.**print**();**

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

temp **=** temp**.**getParent**();**

count**++;**

**}**

System**.**out**.**println**(**"生成节点数："**+(**count**-**1**));**

**}**

/\*\*

\*

\* **@param** open open表

\* **@param** close close表

\* **@param** parent 父状态

\* **@param** target 目标状态

\*/

public void operation**(**ArrayList**<**EightFigure2**>** open**,** ArrayList**<**EightFigure2**>** close**,** EightFigure2 parent**,** EightFigure2 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.**getg**()** **<** open**.**get**(**position**).**getg**()){**

open**.**remove**(**position**);**

**this.**parent **=** parent**;**

**this.**init**(**target**);**

open**.**add**(this);**

**}**

**}**

**}**

**}**

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

//定义open表

ArrayList**<**EightFigure2**>** open **=** **new** ArrayList**<**EightFigure2**>();**

ArrayList**<**EightFigure2**>** close **=** **new** ArrayList**<**EightFigure2**>();**

//创建初始节点和目标节点，这里面包含棋子顺序,f,g,h，指针···属性信息

EightFigure2 start **=** **new** EightFigure2**();**

EightFigure2 target **=** **new** EightFigure2**();**

String lineContent **=** **null;**

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

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

int startnum**[]** **=** **new** int**[**9**];**

int targetnum**[]** **=** **new** int**[**9**];**

int order **=** 0**;**

**try** **{**

BufferedReader br**;**

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

**while((**lineContent**=**br**.**readLine**())!=null){**//读取一行

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

//System.out.println("input.txt每一行个数"+str.length);

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

**if(**order**==**0**)**

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

**else**

targetnum**[**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**(**startnum**);**

target**.**setNum**(**targetnum**);**

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**);** //会按照f的值从小到大排序，因为重写了compareTo()

EightFigure2 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 endTime**=**System**.**currentTimeMillis**();** //获取结束时间

System**.**out**.**println**(**"拓展节点数："**+**close**.**size**());**

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

System**.**out**.**println**((**"转成分钟为："**+(**endTime**-**startTime**)/**60000.0**)+**"min"**);**

System**.**exit**(**0**);**

**}**

//best不是目标节点，继续：

int choice**;**

//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**()){**

choice **=** 0**;**

EightFigure2 temp **=** best**.**Move**(**choice**);**

temp**.**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**()){**

choice **=** 1**;**

EightFigure2 temp **=** best**.**Move**(**choice**);**

temp**.**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**()){**

choice **=** 2**;**

EightFigure2 temp **=** best**.**Move**(**choice**);**

temp**.**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**()){**

choice **=** 3**;**

EightFigure2 temp **=** best**.**Move**(**choice**);**

temp**.**operation**(**open**,** close**,** best**,** target**);**

**}**

**}**

**}else**

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

**}**

**}**

## 算法三：宽度优先搜索，即令估价函数h(n)=0,此时f(n)=g(n)

代码：

**import** java**.**io**.**BufferedReader**;**

**import** java**.**io**.**FileReader**;**

**import** java**.**io**.**IOException**;**

**import** java**.**util**.**ArrayList**;**

**import** java**.**util**.**Arrays**;**

**import** java**.**util**.**Collections**;**

@SuppressWarnings**(**"rawtypes"**)**

public class EightFigure2 **implements** Comparable**{**

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

//这里的g为深度，h为各棋子不在正确位置的数目

//当然h也可以设置为个棋子与目标位置的距离，但是太麻烦

private int g**;** //当前的深度即走到当前状态的步骤

private int f**;** //从起始状态到目标的最小估计值

private int h**;** //到目标的最小估计

private EightFigure2 parent**;** //当前状态的父状态

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

**return** num**;**

**}**

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

**this.**num **=** num**;**

**}**

public int getg**()** **{**

**return** g**;**

**}**

public void setg**(**int g**)** **{**

**this.**g **=** g**;**

System**.**out**.**print**(**"g(n)="**+**g**+**" "**);**

**}**

public int getf**()** **{**

**return** f**;**

**}**

public void setf**(**int f**)** **{**

**this.**f **=** f**;**

System**.**out**.**print**(**"f(n)="**+**f**+**" "**);**

**}**

public int geth**()** **{**

**return** h**;**

**}**

public void seth**(**int h**)** **{**

**this.**h **=** h**;**

System**.**out**.**print**(**"h(n)="**+**h**+**" "**);**

**}**

public EightFigure2 getParent**()** **{**

**return** parent**;**

**}**

public void setParent**(**EightFigure2 parent**)** **{**

**this.**parent **=** parent**;**

**}**

/\*\*

\* 判断当前状态是否为目标状态

\* **@param** target

\* **@return**

\*/

public boolean isTarget**(**EightFigure2 target**){**

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

**}**

/\*\*

\* 求f(n) = g(n)+h(n);

\* 初始化状态信息

\* **@param** target

\*/

public void init**(**EightFigure2 target**){**

int sum **=** 0**;**

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

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

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

int x **=** 0**;**

x**=**Math**.**abs**(**j**-**i**);**

sum **+=** Math**.**pow**(**x**,**1.0**/**4**);**

**}**

**}**

**}**

seth**(**sum**);**

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

**this.**setg**(**0**);**//如果没有父节点

**}else{**

**this.**g **=** **this.**parent**.**getg**()+**1**;**

setg**(**g**);**

**}**

**this.**setf**(this.**getg**()+this.**geth**());**

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

**}**

/\*\*

\* 数学问题如果初始状态和目标状态的逆序值同为奇数或同为偶数

\* 则可以通过有限次数的移动到达目标状态，否则无解。

\* @TODO求逆序值并判断是否有解

\* **@param** target

\* **@return** 有解：true 无解：false

\*/

public boolean isSolvable**(**EightFigure2 target**){**

int reverse **=** 0**;**

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

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

//这里的this就是start，因为用了start.isSolvable(target)

**if(this.**num**[**j**]>this.**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**)** **{**

EightFigure2 t **=** **(**EightFigure2**)** o**;**

**return** **this.**f**-**t**.**getf**();**//默认排序为f(n)由小到大排序

**}**

/\*\*

\* **@return** 返回0在八数码中的位置

\*/

public int getZeroPosition**(){**

int position **=** **-**1**;**

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

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

position **=** i**;**

**}**

**}**

**return** position**;**

**}**

/\*\*

\*

\* **@param** open 状态集合

\* **@return** 判断当前状态是否存在于open表中

\*/

public int isContains**(**ArrayList**<**EightFigure2**>** open**){**

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

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

**return** i**;**

**}**

**}**

**return** **-**1**;**

**}**

/\*\*

\*

\* **@return** 小于3的不能上移返回false

\*/

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

int position **=** getZeroPosition**();**

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

**return** **false;**

**}**

**return** **true;**

**}**

/\*\*

\*

\* **@return** 大于6返回false

\*/

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

int position **=** getZeroPosition**();**

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

**return** **false;**

**}**

**return** **true;**

**}**

/\*\*

\* 是否可以左移，如果空格(0)的位置在

\* 数组的0，3，6就不行，及在最左边一排

\* **@return** 0，3，6返回false

\*/

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

int position **=** getZeroPosition**();**

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

**return** **false;**

**}**

**return** **true;**

**}**

/\*\*

\*

\* **@return** 2，5，8不能右移返回false

\*/

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

int position **=** getZeroPosition**();**

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

**return** **false;**

**}**

**return** **true;**

**}**

/\*\*

\* **@param** move 0：上，1：下，2：左，3：右

\* **@return** 返回移动后的状态

\*/

public EightFigure2 Move**(**int choice**){**

EightFigure2 temp **=** **new** EightFigure2**();**

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(**choice**){**

//0：上，1：下，2：左，3：右:

**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**;**

**}**

/\*\*

\* 按照八数码的格式输出

\*/

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**]+**" "**);**

**}**

**}**

**}**

/\*\*

\* 反序列的输出状态

\*/

public void printRoute**(){**

System**.**out**.**println**(**"因为是反序列输出，请从下往上看"**);**

EightFigure2 temp **=** **null;**

int count **=** 0**;**

temp **=** **this;**

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

temp**.**print**();**

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

temp **=** temp**.**getParent**();**

count**++;**

**}**

System**.**out**.**println**(**"生成节点数："**+(**count**-**1**));**

**}**

/\*\*

\*

\* **@param** open open表

\* **@param** close close表

\* **@param** parent 父状态

\* **@param** target 目标状态

\*/

public void operation**(**ArrayList**<**EightFigure2**>** open**,** ArrayList**<**EightFigure2**>** close**,** EightFigure2 parent**,** EightFigure2 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.**getg**()** **<** open**.**get**(**position**).**getg**()){**

open**.**remove**(**position**);**

**this.**parent **=** parent**;**

**this.**init**(**target**);**

open**.**add**(this);**

**}**

**}**

**}**

**}**

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

//定义open表

ArrayList**<**EightFigure2**>** open **=** **new** ArrayList**<**EightFigure2**>();**

ArrayList**<**EightFigure2**>** close **=** **new** ArrayList**<**EightFigure2**>();**

//创建初始节点和目标节点，这里面包含棋子顺序,f,g,h，指针···属性信息

EightFigure2 start **=** **new** EightFigure2**();**

EightFigure2 target **=** **new** EightFigure2**();**

String lineContent **=** **null;**

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

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

int startnum**[]** **=** **new** int**[**9**];**

int targetnum**[]** **=** **new** int**[**9**];**

int order **=** 0**;**

**try** **{**

BufferedReader br**;**

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

**while((**lineContent**=**br**.**readLine**())!=null){**//读取一行

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

//System.out.println("input.txt每一行个数"+str.length);

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

**if(**order**==**0**)**

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

**else**

targetnum**[**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**(**startnum**);**

target**.**setNum**(**targetnum**);**

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**);** //会按照f的值从小到大排序，因为重写了compareTo()

EightFigure2 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 endTime**=**System**.**currentTimeMillis**();** //获取结束时间

System**.**out**.**println**(**"拓展节点数："**+**close**.**size**());**

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

System**.**out**.**println**((**"转成分钟为："**+(**endTime**-**startTime**)/**60000.0**)+**"min"**);**

System**.**exit**(**0**);**

**}**

//best不是目标节点，继续：

int choice**;**

//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**()){**

choice **=** 0**;**

EightFigure2 temp **=** best**.**Move**(**choice**);**

temp**.**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**()){**

choice **=** 1**;**

EightFigure2 temp **=** best**.**Move**(**choice**);**

temp**.**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**()){**

choice **=** 2**;**

EightFigure2 temp **=** best**.**Move**(**choice**);**

temp**.**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**()){**

choice **=** 3**;**

EightFigure2 temp **=** best**.**Move**(**choice**);**

temp**.**operation**(**open**,** close**,** best**,** target**);**

**}**

**}**

**}else**

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

**}**

**}**

# 运行结果分析：

input,第一行是初始状态，第二行是目标状态

## 第一次测试:

输入： 1,2,3,4,5,6,7,8,0

2,1,3,4,6,5,7,8,0

输出结果：

算法一：生成节点数：18

拓展节点数：1691

程序运行时间： 220ms

转成分钟为：0.0036666666666666666min

算法二：生成节点数：18

拓展节点数：1691

程序运行时间： 341ms

转成分钟为：0.005683333333333334min

算法三：生成节点数：18

拓展节点数：25587

程序运行时间： 21687ms

转成分钟为：0.36145min

## 第二次测试：

输入：5,6,7,4,0,8,3,2,1

1,2,3,8,0,4,7,6,5

算法一：生成节点数：30

拓展节点数：127370

程序运行时间： 476370ms

转成分钟为：7.9395min

算法二：生成节点数：30

拓展节点数：127370

程序运行时间： 457565ms

转成分钟为：7.626083333333334min

算法三：生成节点数：30

拓展节点数：181369

程序运行时间： 868977ms

转成分钟为：14.48295min