

运行代码须知：

1. 数码问题没有解出第三个状态的解，而且由于想要调用哈希函数直接用的 java 完成的数码问题的实验，后面数独问题用的 c。
2. 数码问题没有采用文件输入输出，助教给的三组数据存在了 src 目录下的 input.txt 中。

运行 java 源码只需要在命令行下直接

```
java 22 数码问题迭代 A 星搜索.java
```

```
java 22 数码问题纯 A 星搜索.java
```

即可。

数码问题：

1. 启发式函数：当前所有位置与目标位置的曼哈顿距离之和。

2. 运行结果：

纯 A 星：

```
PS F:\作业2020\人工智能作业\PB17111609_盛开_expl\digit\src> java 22数码问题纯A星搜索.java
请输入测试数据，仅仅只用空格隔开数字，不要打逗号！
2 3 11 4 5
0 8 14 9 10
0 7 7 12 13
1 15 7 16 18
6 19 20 17 21
初始状态的曼哈顿距离：
38
success!!!
循环次数k:2723
要采取的行动:lup lup 6up 19left 15down 7left 14down 14down 11down 3right 2right lup 11down 8right 7up 6up 14left 14left 15up 16left 17up 20left 21left 21left
所需要的步数:24
程序运行时间: 156ms
PS F:\作业2020\人工智能作业\PB17111609_盛开_expl\digit\src> java 22数码问题纯A星搜索.java
请输入测试数据，仅仅只用空格隔开数字，不要打逗号！
1 2 3 4 5
6 7 7 8 9
14 15 7 12 10
0 0 11 17 13
19 20 16 21 18
初始状态的曼哈顿距离：
22
success!!!
循环次数k:78
要采取的行动:14down 6down 15down 7left 8left 11up 16up 21left 9left 10up 13up 18up
所需要的步数:12
程序运行时间: 63ms
PS F:\作业2020\人工智能作业\PB17111609_盛开_expl\digit\src> java 22数码问题纯A星搜索.java
请输入测试数据，仅仅只用空格隔开数字，不要打逗号！
0 6 15 7 7
8 9 13 4 7
1 2 3 10 5
14 11 16 12 18
19 20 17 21 0
初始状态的曼哈顿距离：
50
队列不为空且失败，最后一步的曼哈顿距离:44 执行步数:14
循环次数k=100000
问题求解失败，可以尝试增加循环次数以解决问题
程序运行时间: 1921ms
PS F:\作业2020\人工智能作业\PB17111609_盛开_expl\digit\src>
```

迭代 A 星：

```
PS F:\作业2020\人工智能作业\PE17111609_盛开_exp1\digit\src> java 22数码问题迭代A星搜索.java
请输入测试数据, 仅仅只用空格隔开数字, 不要打逗号!
2 3 11 4 5
0 8 14 9 10
0 7 7 12 13
1 15 7 16 18
3 19 20 17 21
初始状态的曼哈顿距离:
332
7提到正确位置所用步数1271
success!!!
循环次数:1461
要采取的行动:lup lup 15left 7left 14down 11down 3right 2right 14down lup 11down 8right 7up 15right 6up 19left 15down 14left 6up 14left 15up 16left 17up 20left 21left 21left
所需要的步数:26
程序运行时间: 109ms
PS F:\作业2020\人工智能作业\PE17111609_盛开_exp1\digit\src> java 22数码问题迭代A星搜索.java
请输入测试数据, 仅仅只用空格隔开数字, 不要打逗号!
1 2 3 4 5
3 7 7 8 9
14 15 7 12 10
0 0 11 17 13
19 20 16 21 18
初始状态的曼哈顿距离:
319
7提到正确位置所用步数71
success!!!
循环次数:95
要采取的行动:14down 6down 15down 7left 8left 9left 10up 13up 11up 18up 16up 21left
所需要的步数:12
程序运行时间: 47ms
PS F:\作业2020\人工智能作业\PE17111609_盛开_exp1\digit\src> java 22数码问题迭代A星搜索.java
请输入测试数据, 仅仅只用空格隔开数字, 不要打逗号!
0 6 15 17 7
8 9 13 4 7
1 2 3 10 5
14 11 16 12 18
19 20 17 21 0
初始状态的曼哈顿距离:
1238
队列不为空且失败, 最后一步的曼哈顿距离:337 执行步数:39
当前状态
0 15 3 4 0
8 7 7 10 13
1 9 7 0 12
14 2 11 18 5
19 20 17 16 21
现在队列长度145155
循环次数=100000
问题未解决, 可以尝试增加循环次数以解决问题
程序运行时间: 1702ms
```

3. 算法说明：

变量说明：

1) 声明了一个名为 `state` 的 `class`，其中存有

```
public String array;    //数组当前状态
public int    distant;  //当前的曼哈顿距离
public String action;   //当前用过了哪些动作
public int    step;     //初始状态到当前状态有多少步
```

每个 `state` 就是在遍历图搜索上的一个节点。

2) 定义一个全局的哈希表用来存储 `closed` 的节点。

3) 定义一个优先队列 `q`, 其按照曼哈顿距离+步数*2 对各个状态进行排序，小的放前面，以便于优先求解。

算法说明：

将初始状态计算完后推知后面的状态并将后面的可行状态都入队（曼哈顿距离小于初始状态并且不在 `closed` 的哈希表中），每次循环取出队头并计算该头之后的状态。

直到队头曼哈顿距离为 0 或者循环次数到达上限或队列为空才停止。

（这个实验各种尝试求解第三个输入都失败了，随便给个 7, 8 分满分 10 分的话意思一下就 ok）

数独问题：

文件说明： 最终优化版是最好的。其他两个效率都挺差的，要等个 3 秒才能出结果。

1. 算法思想：（仅说明最终优化版的）

初级方法：最开始是直接挨个搜索，找到一个当前值为 0 的位置就直接尝试赋值为任意 1-9 中的数字然后检验是否可行，如果可行就继续对下一个位置进行尝试。如果一直运行完之后不可行就把尝试值赋值为 0 后继续尝试。1-9 都不行之后就返回上一层函数继续尝试。

1) 综合了度启发和前向搜索。

2) 核心是一个数组 `int a[9][9][11]`;

`a[9][9][10]`代表这个位置有多少个可填数字，其他 9 个位置为 0 时代表该位置可填这个数，为 1 时代表不可，而且还可以多次否定。`a[9][9][0]`代表这个位置填的什么数
每次填一个数之后更新 `a` 数组。

3) 每次尝试填数的时候填限制最多的那个点即 `a[9][9][10]`最小的那个点。

4) 失败之后去掉该位置填的数，同时也要更新 `a` 数组，并继续尝试。

2.实验结果说明与分析：

最终优化版直接确定每个位置可以填的数，而且综合了度启发式和前向搜索，最大化了效率，仅执行了 3000 多次递归就得到了 `input3` 的最终解。运行时间也得到了大幅优化。

```
6 7 0 0 0 0 0 0 9
0 9 0 6 0 0 0 0 3
0 3 0 0 7 0 0 0 0
0 0 1 0 0 0 0 0 0
7 0 0 0 0 0 0 0 1
0 0 0 0 0 0 4 0 0
0 0 0 0 6 0 0 2 0
4 0 0 0 0 3 0 8 0
3 0 0 0 0 0 0 5 7
剩余空位数61
数独的解为：
得出最终解所需递归次数：3475
6 7 5 3 1 8 2 4 9
8 9 4 6 2 5 7 1 3
1 3 2 4 7 9 5 6 8
9 4 1 5 3 6 8 7 2
7 5 3 8 4 2 6 9 1
2 8 6 7 9 1 4 3 5
5 1 8 9 6 7 3 2 4
4 2 7 1 5 3 9 8 6
3 6 9 2 8 4 1 5 7

程序运行时间109 ms
Process returned 0 (0x0)   execution time : 0.187 s
Press any key to continue.
```

```
6 7 0 0 0 0 0 0 9
0 9 0 6 0 0 0 0 3
0 3 0 0 7 0 0 0 0
0 0 1 0 0 0 0 0 0
7 0 0 0 0 0 0 0 1
0 0 0 0 0 0 4 0 0
0 0 0 0 6 0 0 2 0
4 0 0 0 0 3 0 8 0
3 0 0 0 0 0 0 5 7
数独的解为：
搜索次数:4384596
第1个填法为：
6 7 5 3 1 8 2 4 9
8 9 4 6 2 5 7 1 3
1 3 2 4 7 9 5 6 8
9 4 1 5 3 6 8 7 2
7 5 3 8 4 2 6 9 1
2 8 6 7 9 1 4 3 5
5 1 8 9 6 7 3 2 4
4 2 7 1 5 3 9 8 6
3 6 9 2 8 4 1 5 7
程序运行时间2390 ms
Logs & others
```

3.思考题：

- 1) 爬山算法可以利用度启发式，每次选择受最大约束的点来尝试。可能会走进死胡同，而我当前的算法已经用回溯解决了这个问题。
- 2) 遗传算法可以先随机填上所有的空，但是要把输入的数字给固定住，不能动这些数字。然后把产生冲突的次数当作适应度函数来产生后代。
(不过感觉由于条件有点苛刻，可能运算次数会很多，存不下。)
- 3) 模拟退火搜索：(不太清楚怎么用这个实现)。