人工智能实验

week5&6　　A* & IDA*

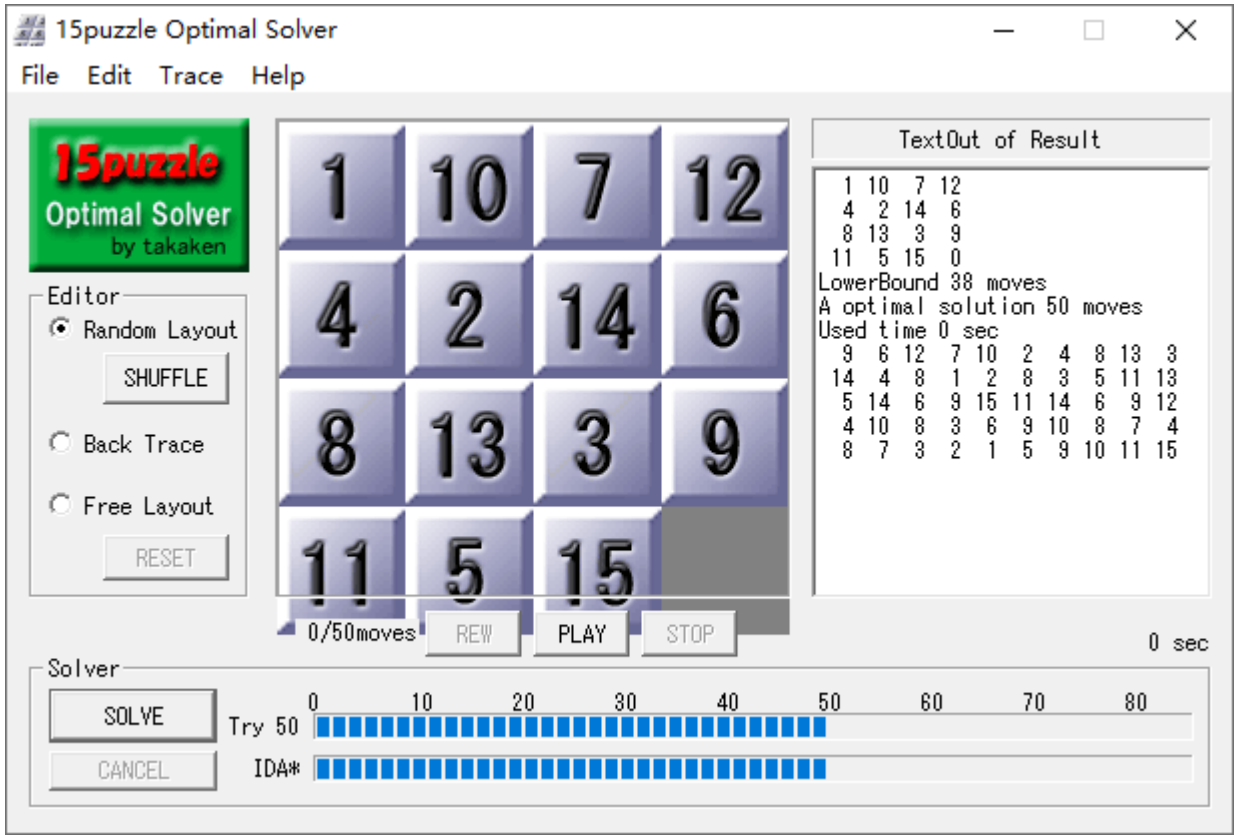
实验任务·项目

- 使用A*与IDA*算法求15-Puzzle问题的**最优解**，启发式函数可以自己选取，可以多尝试几种不同的启发式函数，完成**压缩包的4个测例**及**ppt上的4个测例**（最后两个难度较高）
- 结果分析要求
  - 对比分析A*和IDA*的性能和结果（Memory Out请附上内存限制）
  - 如果使用了多种启发式函数也可以进行对比和分析
- 加分项（供参考）
  - 算法实现优化分析（使用数据结构、利用性质的剪枝等）
  - 未提及的启发式函数实现、对比和分析
- 启发式函数参考
  - 曼哈顿距离 $|x_1 - x_2| + |y_1 - y_2|$
  - 切比雪夫距离 $\max(|x_1 - x_2|, |y_1 - y_2|)$
- Deadline
  - 两周，**4月8日 [周一] 23:59**
- 提交命名
  - E3_学号.zip

15 Puzzle

输入格式
4*4矩阵，0表示空位

输入示例
1 10 7 12
4 2 14 6
8 13 3 9
11 5 15 0



输出格式

1. 最优解步数（如上图中的50）
2. 对应的动作序列（如上图中的9 6 12 7 10 2 4 8 13 …）
3. 每一步的局面（动作序列中每执行一个动作后，呈现的4*4局面）

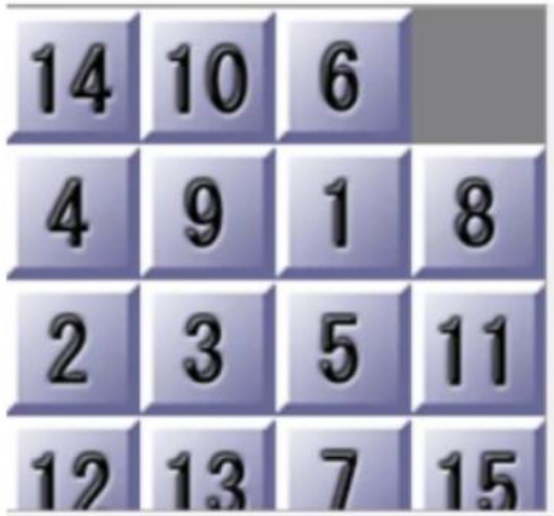
PS: 实验报告的结果展示中不用包括第三点，但代码验收时需要

测例

较难，建议先做压缩包的测例

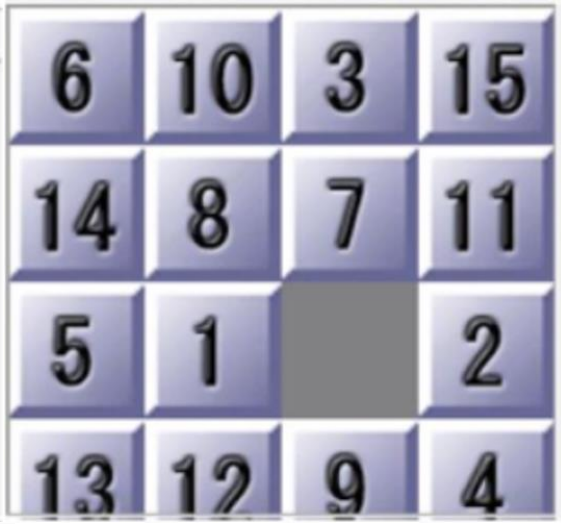
例1

14 10 6 0
4 9 1 8
2 3 5 11
12 13 7 15



TextOut of Result

```
 14 10  6  0
  4  9  1  8
  2  3  5 11
 12 13  7 15
LowerBound 37 moves
A optimal solution 49 moves
Used time 0 sec
  6 10  9  4 14  9  4  1 10  4
  1  3  2 14  9  1  3  2  5 11
  8  6  4  3  2  5 13 12 14 13
 12  7 11 12  7 14 13  9  5 10
  6  8 12  7 10  6  7 11 15
```



TextOut of Result

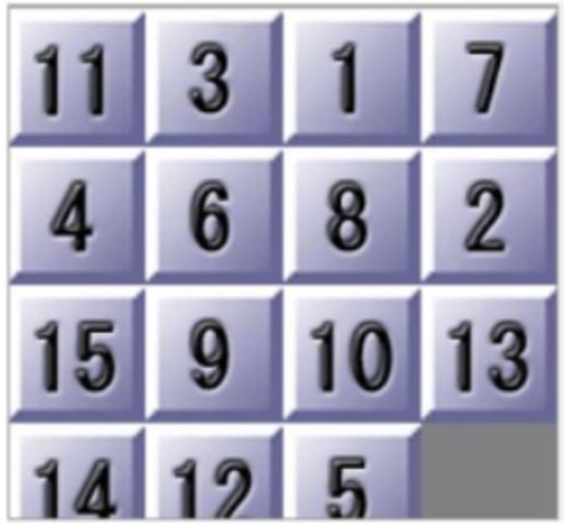
```
  6 10  3 15
 14  8  7 11
  5  1  0  2
 13 12  9  4
LowerBound 32 moves
A optimal solution 48 moves
Used time 0 sec
  9 12 13  5  1  9  7 11  2  4
 12 13  9  7 11  2 15  3  2 15
  4 11 15  8 14  1  5  9 13 15
  7 14 10  6  1  5  9 13 14 10
  6  2  3  4  8  7 11 12
```

例2

6 10 3 15
14 8 7 11
5 1 0 2
13 12 9 4

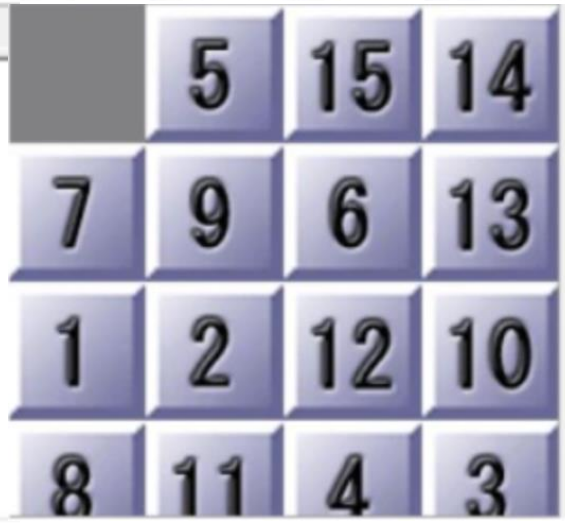
例3

11 3 1 7
4 6 8 2
15 9 10 13
14 12 5 0



TextOut of Result

```
 11  3  1  7
  4  6  8  2
 15  9 10 13
 14 12  5  0
LowerBound 36 moves
A optimal solution 56 moves
Used time 3 sec
 13 10  8  6  9 12  5 13 10  8
 12 15 14  5 13 12 15 14  5 13
 14  9  4 11  3  1  6  4 11  3
  1  6  4  2  8 10 12 15 10  8
  7  4  2 11  3  5  9 10 11  3
  6  2  3  7  8 12
```



TextOut of Result

```
  0  5 15 14
  7  9  6 13
  1  2 12 10
  8 11  4  3
LowerBound 44 moves
A optimal solution 62 moves
Used time 4 sec
  7  9  2  1  9  2  5  7  2  5
  1 11  8  9  5  1  6 12 10  3
  4  8 11 10 12 13  3  4  8 12
 13 15 14  3  4  8 12 13 15 14
  7  2  1  5 10 11 13 15 14  7
  3  4  8 12 15 14 11 10  9 13
 14 15
```

例4

0 5 15 14
7 9 6 13
1 2 12 10
8 11 4 3

A*算法（仅供参考）

- 估价函数
  - $f(x) = h(x) + g(x)$

- 算法描述
  - 从起始状态开始，每次选取未进行拓展的状态中估价函数最小的状态进行拓展，不断搜索新的可达状态并计算它们的估价函数值，或更新待进行拓展的状态的$g(x)$，对已进行拓展的状态进行剪枝，直到找到目标状态。

- 步骤
  1. 从起始状态开始，把其当成待拓展状态存入一个“开启列表”
  2. 从“开启列表”中找到估价函数最小的状态C，并将它从开启列表中删除，添加到“关闭列表”中（列表中保存所有不需要再次搜索的状态）
  3. 检查C所有相邻状态，
     1. 如果状态在关闭列表中，则剪枝；
     2. 否则，则计算h, g, f，将C作为“父状态”，并加入开启列表
  4. 重复步骤2,3，直到找到目标状态（步骤2抽取的是目标状态）或者开启列表为空

IDA*算法（仅供参考）

- 估价函数
  - $f(x) = h(x) + g(x)$

- 算法描述
  - 对估价函数值的阈值进行迭代，在迭代的每一步都进行深度优先搜索，选择相邻状态中的一个状态进行递归，若当前状态的估价函数值大于当前的阈值，则进行剪枝，直到到达目标状态或无解，否则更新阈值继续迭代。

- 步骤
  1. 对估价函数值的阈值进行迭代增加，对于给定的一个阈值，定义递归过程
     1. 从起始状态开始，计算所有相邻状态的估价函数值，选取其中一个状态进行递归
     2. 如果当前状态的估价函数值大于阈值，则记录当前状态的估价函数值（以确定阈值的下一个迭代值），回溯
     3. 如果当前状态是目标状态，则记录答案，结束递归
  2. 如果递归找到目标状态，则算法结束
  3. 如果递归过程中所有到达状态的估价函数值均小于等于阈值，则目标状态无法到达，算法结束;
  4. 根据递归过程中记录的超出阈值的估价函数值更新阈值（如最小值），再次进行步骤1的递归过程，直到找到目标状态，或无解

提示

- 如何实现Open列表中的排序?
  - heapq/PriorityQueue
  - __lt__函数
  - …
- Closed列表用什么数据结构?
- 如何尽可能减少每次扩展出的状态?
  - 是否上下左右四个方向都要考虑?
  - …
- 交换list中的两个元素的位置:
  - list1[index0], list1[index1] = list1[index1], list1[index0]
- 启发式函数的选取
- 改进算法
- …

PS: 未优化的A*可能会带来内存耗尽问题，请留意设备的内存限制