

1. 使用启发式函数 **h1**, 动态显示 **OPEN** 表的结点数、总扩展的结点数和评估函数最小的结点. 若 **de** 出结果, 则输出 **OPEN** 表中在最佳路径上的结点及其评估函数值

```
C:\WINDOWS\system32\cmd.exe
OPEN表节点数: 3
总扩展节点数: 4
评估函数最小的结点:
2 8 3
1 6 4
7 0 5
+-----+

OPEN表节点数: 6
总扩展节点数: 8
评估函数最小的结点:
2 8 3
1 0 4
7 6 5
+-----+

OPEN表节点数: 8
总扩展节点数: 11
评估函数最小的结点:
2 8 3
0 1 4
7 6 5
+-----+

OPEN表节点数: 10
总扩展节点数: 14
评估函数最小的结点:
2 0 3
1 8 4
7 6 5
+-----+

OPEN表节点数: 11
总扩展节点数: 16
评估函数最小的结点:
0 2 3
1 8 4
7 6 5
+-----+

OPEN表节点数: 13
总扩展节点数: 19
评估函数最小的结点:
1 2 3
0 8 4
7 6 5
+-----+
```

找到结果，成功退出

最佳路径的逆序输出如下

```
+-----+
1 2 3   评估涵值为5
8 0 4
7 6 5

1 2 3   评估涵值为4
0 8 4
7 6 5

0 2 3   评估涵值为3
1 8 4
7 6 5

2 0 3   评估涵值为2
1 8 4
7 6 5

2 8 3   评估涵值为1
1 0 4
7 6 5
```

2. 使用启发式函数 **h2**, 动态显示 **OPEN** 表的结点数、总扩展的结点数和评估函数最小的结点. 若 **de** 出结果, 则输出 **OPEN** 表中在最佳路径上的结点及其评估函数值

```
OPEN表节点数: 3
总扩展节点数: 4
评估函数最小的结点:
2 8 3
1 6 4
7 0 5
+-----+

OPEN表节点数: 6
总扩展节点数: 8
评估函数最小的结点:
2 8 3
1 0 4
7 6 5
+-----+

OPEN表节点数: 8
总扩展节点数: 11
评估函数最小的结点:
2 0 3
1 8 4
7 6 5
+-----+

OPEN表节点数: 9
总扩展节点数: 13
评估函数最小的结点:
0 2 3
1 8 4
7 6 5
+-----+

OPEN表节点数: 11
总扩展节点数: 16
评估函数最小的结点:
1 2 3
0 8 4
7 6 5
+-----+
```

找到结果，成功退出

最佳路径的逆序输出如下

```
+-----+
1 2 3   评估涵值为5
8 0 4
7 6 5

1 2 3   评估涵值为4
0 8 4
7 6 5

0 2 3   评估涵值为3
1 8 4
7 6 5

2 0 3   评估涵值为2
1 8 4
7 6 5

2 8 3   评估涵值为1
1 0 4
7 6 5
```

### 3. 比较启发函数 $h_1$ 和 $h_2$ 的搜索效率

由上述的结果可以看出，为了搜索到结果， $h_1$  执行了 6 步， $h_2$  执行了 5 步，因此  $h_2$  的搜索效率相对较高

### 4. 验证凡 A\* 算法挑选出来求后继的点 $n$ 必定满足: $f(n) \leq f^*(S_0)$

令  $S_0 = n_0, n_1, n_2, \dots, n_k = S_g$  为一条最优路径，设  $n' \in \text{path}(n_0, n_1, n_2, \dots, n_k)$  中最后一个出现在 Open 表上的元素。显然， $n'$  一定存在，因为至少有  $S_0 = n_0$  必然在 Open 表上，只考虑当  $n_k$  还未出现在 Closed 表中时，因为若  $n_k$  已在 Closed 表中时，则  $n_k = S_g$ ，A\* 算法将终止于成功退出。

由定义有

$$\begin{aligned} f(n') &= g(n') + h(n') = g^*(n') + h(n') && (\text{因为 } n' \text{ 在最优路径上}) \\ &\leq g^*(n') + h^*(n') = f^*(n') = f^*(S_0) \end{aligned}$$

所以， $f(n') \leq f^*(S_0)$  成立

### 5. 验证 $h_1(n)$ 的单调性，显示凡 A\* 算法挑选出来求后继的点 $n_i$ 扩展

的一个子结点  $n_j$ ，检查是否满足:  $h(n_i) \leq 1 + h(n_j)$

$h(n_i)$ 表示从节点  $n_i$ 到目标节点的最佳路径的代价,  $c(n_i, n_j)$ 表示节点  $n_i$ 到节点  $n_j$ 的最佳路径的代价, 根据三角不等式可知:

$$h(n_i) \leq h(n_j) + c(n_i, n_j)$$

又因为  $n_j$  是  $n_i$  的后继节点, 所以  $c(n_i, n_j) = 1$ , 带入得

$$h(n_i) \leq h(n_j) + 1$$

6. 对于九数码问题, 启发函数  $h_1(n)$ 的结果如下

```
OPEN表节点数: 3
总扩展节点数: 4
评估函数值最小的结点:
2 8 3
1 6 4
7 0 5
+-----+

OPEN表节点数: 6
总扩展节点数: 8
评估函数值最小的结点:
2 8 3
1 0 4
7 6 5
+-----+

OPEN表节点数: 8
总扩展节点数: 11
评估函数值最小的结点:
2 8 3
0 1 4
7 6 5
+-----+

OPEN表节点数: 11
总扩展节点数: 15
评估函数值最小的结点:
2 8 3
1 0 4
7 6 5
+-----+
```

```
OPEN表节点数: 12
总扩展节点数: 17
评估函值最小的结点:
2 0 3
1 8 4
7 6 5
```

```
OPEN表节点数: 13
总扩展节点数: 19
评估函值最小的结点:
0 2 3
1 8 4
7 6 5
```

```
OPEN表节点数: 15
总扩展节点数: 22
评估函值最小的结点:
1 2 3
0 8 4
7 6 5
```

找到结果，成功退出

最佳路径的逆序输出如下

```
1 2 3    评估涵值为5
8 0 4
7 6 5
```

```
1 2 3    评估涵值为4
0 8 4
7 6 5
```

```
0 2 3    评估涵值为3
1 8 4
7 6 5
```

```
2 0 3    评估涵值为2
1 8 4
7 6 5
```

```
2 0 3    评估涵值为2
1 8 4
7 6 5
```

```
2 8 3    评估涵值为1
1 0 4
7 6 5
```

7. 对于九数码问题, 启发函数  $h_2(n)$ 的结果如下

```
OPEN表节点数: 3
总扩展节点数: 4
评估函数值最小的结点:
2 8 3
1 6 4
7 0 5
+-----+

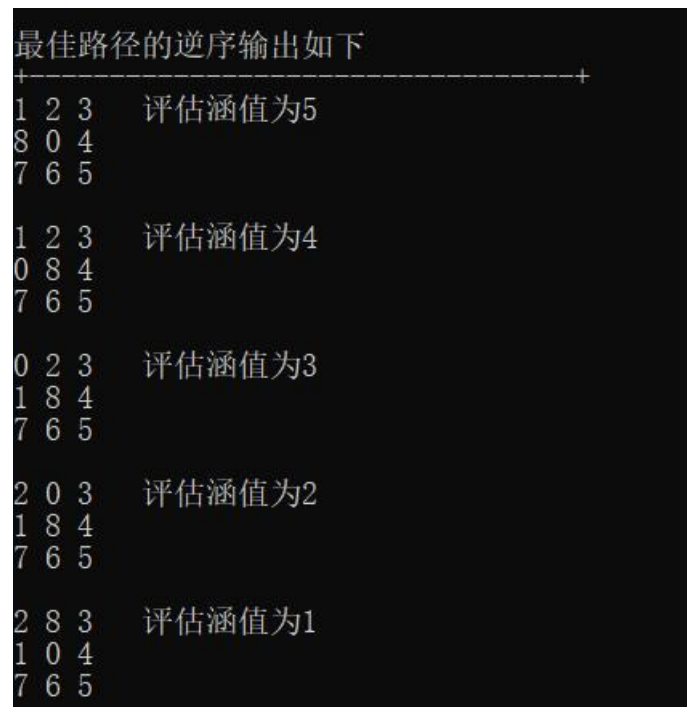
OPEN表节点数: 6
总扩展节点数: 8
评估函数值最小的结点:
2 8 3
1 0 4
7 6 5
+-----+

OPEN表节点数: 8
总扩展节点数: 11
评估函数值最小的结点:
2 0 3
1 8 4
7 6 5
+-----+

OPEN表节点数: 9
总扩展节点数: 13
评估函数值最小的结点:
0 2 3
1 8 4
7 6 5
+-----+

OPEN表节点数: 11
总扩展节点数: 16
评估函数值最小的结点:
1 2 3
0 8 4
7 6 5
+-----+

找到结果, 成功退出
```



由上述的结果可以看出，八数码和九数码问题的搜索图不同

8. 当除 0 外的数字的位置都没有错误时，到达目标状态