八数码示例讲解

潘林朝

八数码状态表示

八数码的状态共有9个数字，我使用python的list类型表示，即：

```python
"""
start_state:
    0 1 2
    3 4 5
    6 7 8
"""
start_state = [0, 1, 2, 3, 4, 5, 6, 7, 8]
```

判定状态之间是否存在解

基于逆序数的概念，可以判定八数码的初始状态和目标状态是否存在可行解。可参考：

- python---A*搜索算法解决八数码问题
- 人工智能结课作业-DFS/BFS/Astar解决八数码问题

示例代码：

```python
def have_path(self):
    tmp = list(self.start)
```

```
    tmp1 = list(self.end)
    s = 0
    s1 = 0
    for i in range(1, len(tmp1)):
        t = 0
        if tmp1[i] == 0:
            continue
        for j in range(0, i):
            if tmp1[j] == 0:
                continue
            if tmp1[i] < tmp1[j]:
                t += 1
        s1 += t
    for i in range(1, len(tmp)):
        t = 0
        if tmp[i] == 0:
            continue
        for j in range(0, i):
            if tmp[j] == 0:
                continue
            if tmp[i] < tmp[j]:
                t += 1
        s += t
    if s % 2 == 0 and s1 % 2 == 0:  # 是偶数，即和目标状态同奇偶性质，存在路径
        return True
    if s % 2 != 0 and s1 % 2 != 0:
```

```
            return True
        return False
```

节点类

这也可以称为状态类，即状态空间搜索里的一个状态。我定义的节点类代码如下：

```python
class Node:
    def __init__(self, state: list, gg: int, h_cost, parent=None):
        self.state: list = state
        self.gg: int = gg  # 深度
        self.h_cost = h_cost  # 预估函数
        self.total_cost = self.gg + self.h_cost
        self.parent: Node = parent  # 父节点，默认值是None，即最初始节点的父节点

    def __eq__(self, other):
        return self.state == other.state

    def __lt__(self, other):
        return self.total_cost < other.total_cost

    def __gt__(self, other):
        return self.total_cost > other.total_cost
```

扩展节点

根据八数码中0数字的位置，可以判定上下左右移动后是否合法，以生成后继状态。代码如下：

```python
def expand(cur: list):  # 尝试cur的移动，产生一个列表
    ans = []
    dis = cur.index(0)
    if dis + 3 <= 8:
        ans.append(cur.copy())
        ans[-1][dis], ans[-1][dis + 3] = ans[-1][dis + 3], ans[-1][dis]
        # ("UP")
    if dis - 3 >= 0:
        ans.append(cur.copy())
        ans[-1][dis], ans[-1][dis - 3] = ans[-1][dis - 3], ans[-1][dis]
        # ("DOWN")
    if dis % 3 != 2:
        ans.append(cur.copy())
        ans[-1][dis], ans[-1][dis + 1] = ans[-1][dis + 1], ans[-1][dis]
        # ("LEFT")
    if dis % 3 != 0:
        ans.append(cur.copy())
        ans[-1][dis], ans[-1][dis - 1] = ans[-1][dis - 1], ans[-1][dis]
        # ("RIGHT")
    return ans
```

PYTHON

也可以自定义一个 dict 表，表示 0 的不同位置可以移动的方向，就不用上面的if判断了。

打印路径

类似第一次介绍八数码时给出的搜索示例，利用节点中 parent 的特性，可以迭代打印。

生成状态

为了测试算法性能，可以自定义目标状态，然后随机生成初始状态，进行多次实验。代码如下：

```python
"""
end_state:
    1 2 3
    8 0 4
    7 6 5
"""
end_state = [1, 2, 3, 8, 0, 4, 7, 6, 5]

import random
start_state = random.shuffle(end_state)
```

然后，使用逆序数判断是否有解，进而使用实现的搜索算法求解路径。