1、（习题）利用状态空间法对汉诺塔或修道士问题进行表示，并给出至少一个解的搜索过程。

1.1 汉诺塔问题解法一

确定状态变量和状态空间： n 阶汉诺塔问题可以用 **n*3 的矩阵表示**。每一行代表一个圆盘的位置分布情况，因为只有三根柱子可以放置圆盘，所以矩阵只有三列。

初始状态： 第一列为从 1 到 n 的数，表示圆盘的大小。

目标状态： 第三列为从 1 到 n 的数。

约束条件： 移动的元素只能是某一列从上往下数的第一个非零元素，也就是最表层的圆盘（比如初始状态只能移动 1 而不能移动 2 和 3）；移动后该元素需为该列第一个非零元素（比如 2 不能直接插进 1 跟 3 中间）；为了避免重复操作，上一步移动的**逆向操作**的下一步是禁止的。

启发式函数设计： 此处展示的是**三阶汉诺塔的 A*搜索**，启发式函数为**第三列未在位圆盘的大小值之和**，并且**需满足第三列从下往上满足大盘至小盘原则**（比如第三列只有 1 跟 2 没有 3，那么 1 跟 2 也无效，h(n)仍为 6，如果只有 3 没有 1 跟 2，h(n)为 3）。

结果： 此处共使用了 **15 次搜索**得出了结果，共需**移动 7 步**。

（矩阵左上方表示搜索的次数，下方表示 g(n)+h(n)）

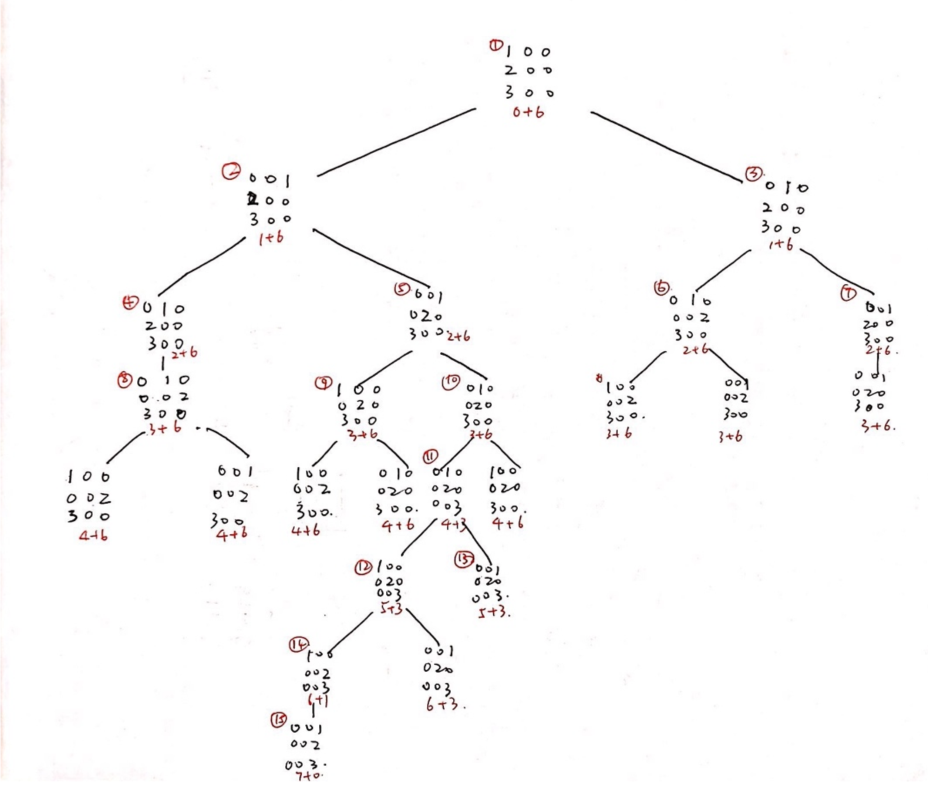


图 1： 汉诺塔问题解法一：状态空间图

1.2 汉诺塔问题解法二

上述 3*3 矩阵的解法在算法实现的过程中会占用较大的内存，使用下面解法二的状态空间表示法可以使算法的效率提高。

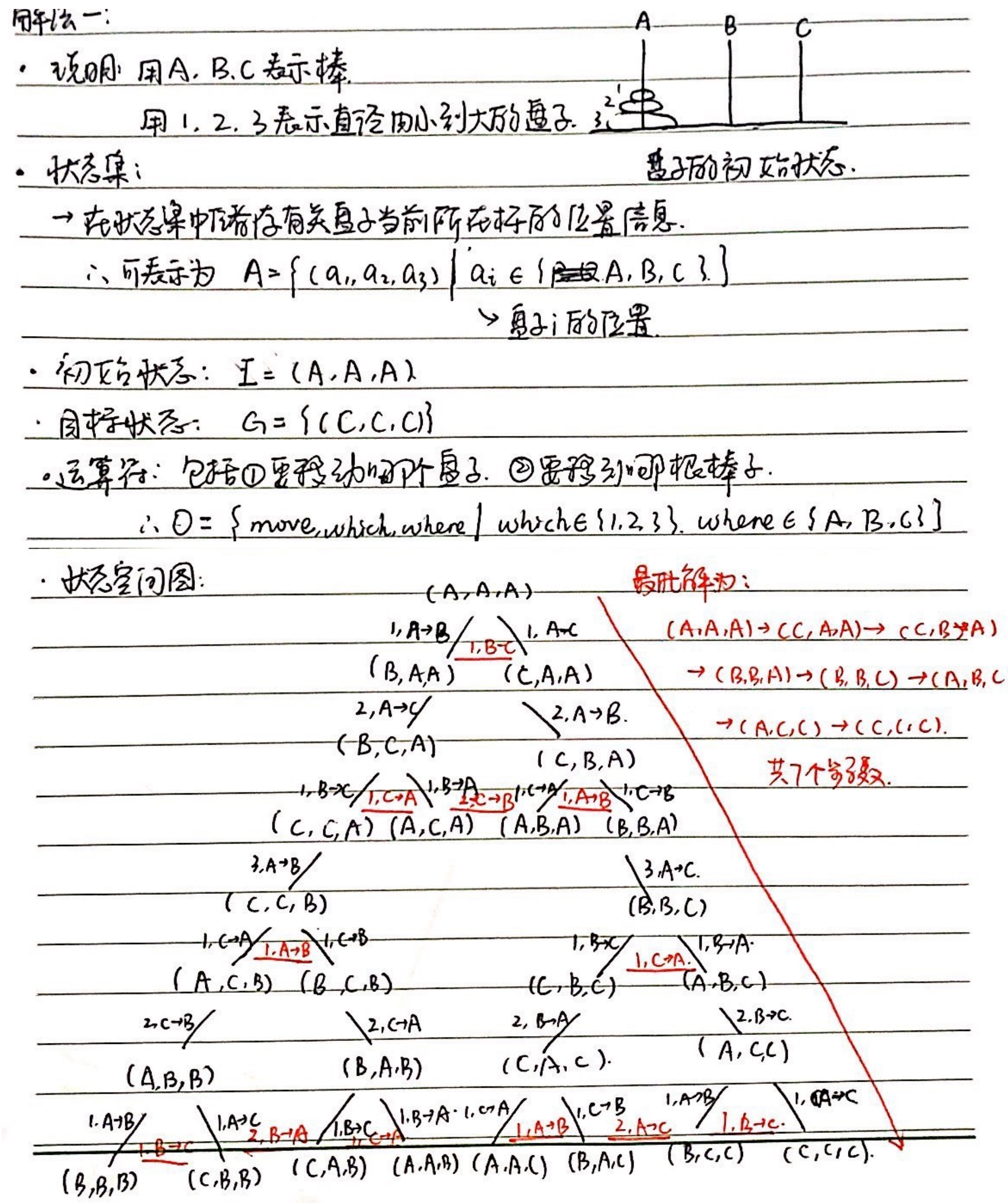


图 2：汉诺塔问题解法二：状态空间图

1.3 修道士问题[1]

确定状态变量和状态空间：选择三元数组作为修道士问题的状态变量，用以描述左岸修道士、野人和船的数量。假设左岸修道士数量为 a，则 a={0, 1, 2, 3}；左岸野人数量为 b, 则 b={0, 1, 2, 3}；左岸船只数量为 c, c={0, 1}；右岸状态不予考虑，此处只选取左岸状态构建状态空间。某个状态可以用三元数组表示为 S=(a, b, c)。

起始状态：S_0 =(3, 3, 1)，表示修道士、野人和船都在左岸准备出发。

目标状态：S_g=(0, 0, 0)，表示左岸的所有人全部完成渡河。

[1] 参考：https://blog.csdn.net/shdhhfhj/article/details/79610233

操作函数：小船最多可承载两人，因此每次渡河时可能只载 1 人，也可能载 2 人。同时，小船还有往返两个操作，记从左至右为操作 P，从右至左记操作 Q。最后得出操作集合：F={P01,P02,P10,P20,P11,Q01,Q02,Q10,Q20,Q11}

启发式函数设计：问题的目的是将左岸的传教士和野人全部运往右岸，因此解搜索过程是朝着右岸人数更多的方向发展，当然，并不是每个右岸人多的状态都是我们想要的，中间状态需要满足两岸传教士数量大于野人数量。因此可得启发式函数为：

$$Fn = \begin{cases} b-a-b & \text{传教士} \geq \text{野人} \\ -\infty & \text{其他.} \end{cases}$$

图 3：修道士问题启发式函数

结果：其中一个解如下图所示，最短路径由 11 次操作构成。

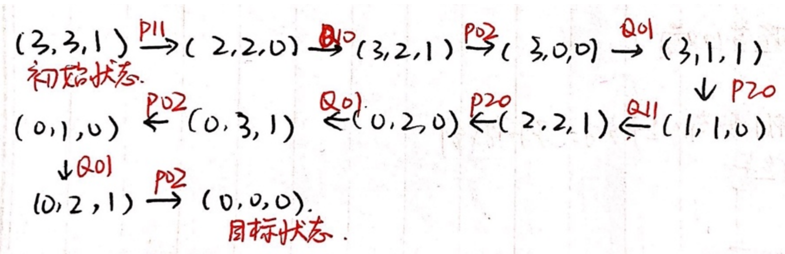
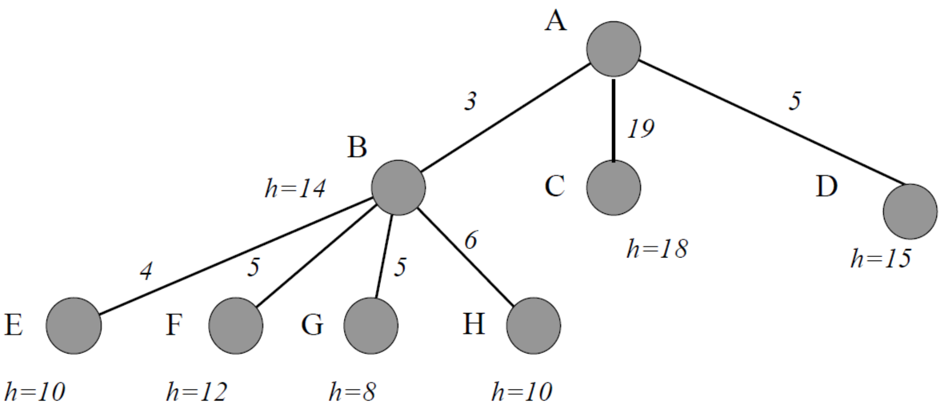


图 4：修道士问题状态空间图

2、（习题）计算宽度优先、深度优先、一致代价、贪婪和 A*算法搜索过程（注：以 H 为目标结点，写出节点搜索次序，并标注结点代价信息）。



宽度优先：

搜索路径 ABCDEFGH

深度优先：

搜索路径 ABEFGH

一致代价：

搜索路径 A-B(3)-D(5)-E(7)-F(8)-G(8)-H(9)

贪婪算法：

搜索路径 A-B(h=14)-G(h=8)-H(h=10)

A*算法：

搜索路径 A-B(3+14)-G(8+8)-E(7+10)-H(9+10)

3、（编程）实现 N 皇后至少一种搜索方法（回溯、爬山、模拟退火，束搜索、遗传算法），并分析其算法的性能（四个搜索算法评价指标）。

代码

请见 NQueen.py，实现了回溯法与随机初始化的爬山法。第一项输入为皇后的数量，第二项输入为算法类型的选择（a 为回溯法，b 为爬山法）；回溯法输出所有解、该解的搜索次数以及解的数量，爬山法输出第一个解以及搜索次数。

回溯法：[2]

回溯法的核心指的是在遍历当前行时，之前行的棋子已经确定了，以此来判断该行的各个位置是否可行。若不可行，则回溯到上一步，改变位置之后再重新调换位置看是否满足约束条件。

算法具有完备性和最优性。时间复杂度为$O(n!)$，空间复杂度$O(n)$，在 N 皇后的解法中效率最低，但能保证找到最优解。

爬山法：[3]

首先在搜索空间随机选取一点作为进行迭代的初始点，然后在其邻域内随机产生一点，计算其函数值，若该点函数值优于当前点，则用当前点替换初始点作为新的初始点继续在邻域内搜索，否则继续在邻域随机产生另一个点与初始点进行比较，直到找到比其优秀一点或连续几次都找不到比其优秀的点则终止搜索过程。

算法不具有完备性和最优性。时间复杂度小于回溯法，空间复杂度为$O(1)$，也小于回溯法。但因为有可能陷入局部最优无法退出，因此不像回溯法那样具有完备性和最优性。

结果截图：

[2] 参考：https://blog.csdn.net/weixin_40411446/article/details/80379952

[3] 参考：https://blog.csdn.net/mago2015/article/details/118549518

图 5：八皇后问题问题（回溯法）

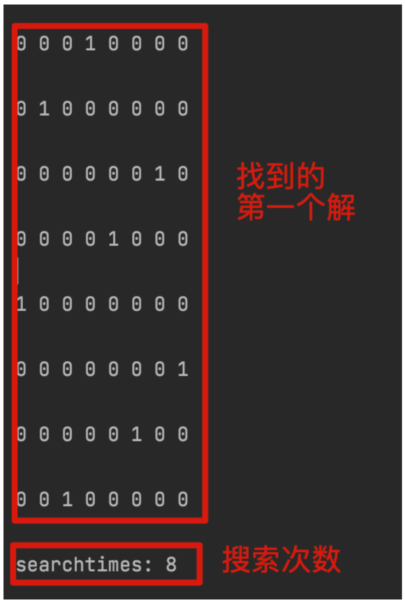


图 6：八皇后问题问题（爬山法）

算法性能：

	回溯算法	爬山算法	模拟退火	束搜索	遗传算法
完备性	完备	不完备	不完备	不完备	不完备
最优性	是	否（可能陷于局部最优）	否（可能陷于局部最优）	否（可能陷于局部最优）	否（可能陷于局部最优）
时间复杂性	$O(n!)$（剪去列冲突的分支） $O(n^n)$（无剪枝）	$O(n^3)$（计算一次所有近邻的冲突对数）	$O(C*n^2)$（C 为找到正确解的步骤或退火结束时的步骤）	$O(Bm)$（B 是束宽度，m 是最大深度）	$O(n^2)$
空间复杂性	$O(n)$	$O(1)$	$O(1)$	$O(Bm)$	$O(n)$