人工智能原理与技术第 4 周作业

2154312 郑博远

传教士和野人问题。三个传教士和三个野人在河的一岸，有一条能载一个人或者两个人的船。请设法使所有人都渡到河的另一岸，要求在任何地方野人数都不能多于传教士的人数。这个问题在 AI 领域中很有名，是因为它是第一个从分析的观点探讨问题形式化的论文的主题(Amarel，1968)。

a. 请对该问题进行详细形式化，只描述确保该问题求解所必需的特性。画出完整的状态空间图。

b. 应用合适的搜索算法求出该问题的最优解。对于这个问题检查重复状态是个好主意吗?

c. 这个问题的状态空间很简单，你认为是什么导致人们求解它很困难?

解答：

a.

以三元组 T = (M, S, B)表示当前所在的状态，其中 M 表示原河岸一侧的传教士个数，S 表示原河岸一侧的野人个数，B 表示停靠在原河岸一侧的船只数。

初始状态：对应三元组 T = (3, 3, 1)。即原河岸侧有传教士、野人各 3 个，船停靠在原河岸一侧。

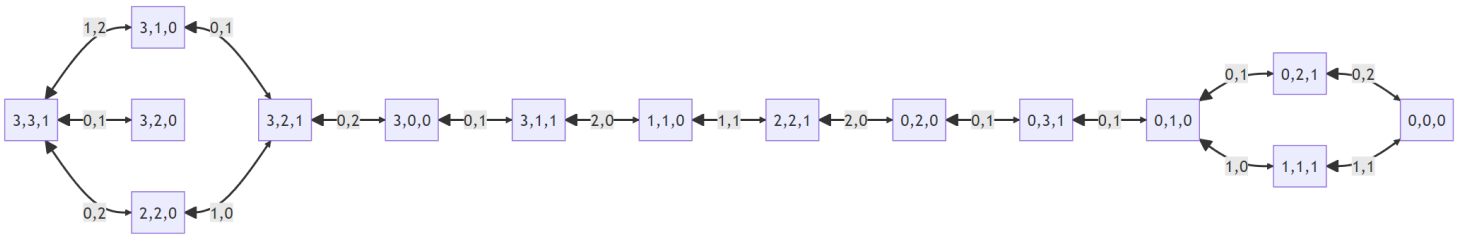
行　　动：一到两人乘船从河岸一侧到另一侧（B 在 0 与 1 之间切换）。

转移模型：行动产生对应的结果（任何地方野人数不得多于传教士的人数）。

目标检测：检测是否是三元组 T = (0, 0, 0)。即目标河岸侧有传教士、野人各 3 个，船停靠在目标河岸一侧。

路径消耗：每一步耗散值为 1，因此整个解路径的耗散值是船的来往次数。

完整的状态空间图如下：



b.

可以用深度优先搜索、宽度优先搜索、A*搜索进行求解。下面以深度优先搜索举例：

```
def dfs(T: tuple, stack:list, visit:set):
    if T not in visit:
        visit.add(T)
    else:
        return

    stack.append(T)

    if(T == (0, 0, 0)):
        print(stack)
        return

    m, s, b = T

    # 枚举当前选择
    alt = [(1, 0), (0, 1), (0, 2), (2, 0), (1, 1)]
    for dm, ds in alt:
        if(b == 1):
            m1 = m - dm
            s1 = s - ds
        else:
            m1 = m + dm
            s1 = s + ds
        # 排除非法情况
        if(m1 < 0 or m1 > 3 or s1 < 0 or s1 > 3):
            continue
        elif(m1 != 0 and m1 < s1):
            continue
        elif(3 - m1 != 0 and 3 - m1 < 3 - s1):
            continue

        dfs((m1, s1, 1-b), stack, visit)

    stack.pop()


dfs((3,3,1), [], set())
```

搜索序列为：(3, 3, 1) (3, 2, 0) (3, 1, 0) (3, 2, 1) (3, 0, 0) (3, 1, 1) (1, 1, 0) (2, 2, 1) (0, 2, 0) (0, 3, 1) (0, 1, 0) (1, 1, 1) (0, 0, 0)。

程序打印解序列如下：

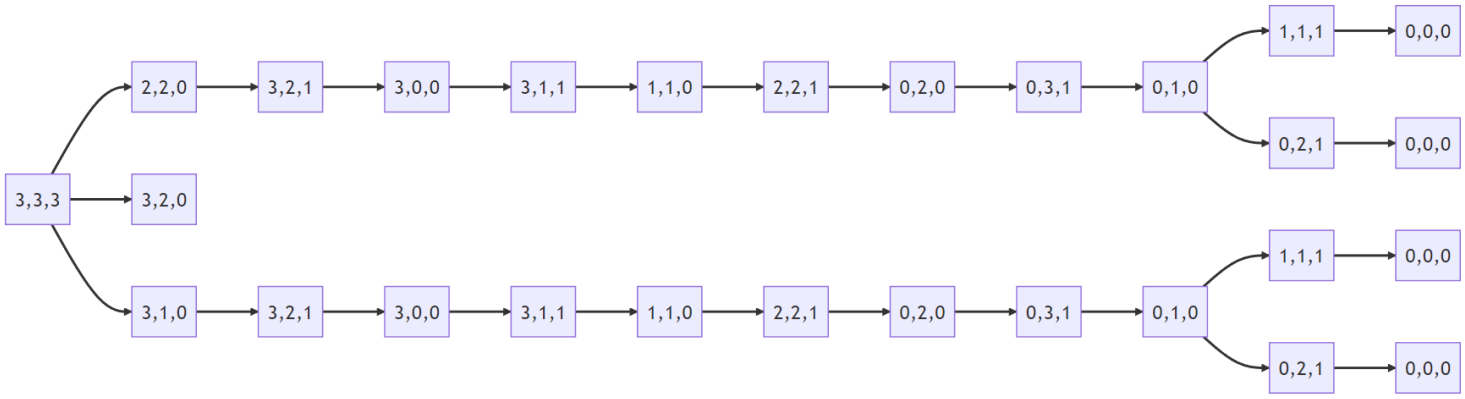
(3, 3, 1), (3, 1, 0), (3, 2, 1), (3, 0, 0), (3, 1, 1), (1, 1, 0), (2, 2, 1), (0, 2, 0), (0, 3, 1), (0, 1, 0), (1, 1, 1), (0, 0, 0)

还有其他可能的解序列如下：

(3, 3, 1), (2, 2, 0), (3, 2, 1), (3, 0, 0), (3, 1, 1), (1, 1, 0), (2, 2, 1), (0, 2, 0), (0, 3, 1), (0, 1, 0), (1, 1, 1), (0, 0, 0)

(3, 3, 1), (3, 1, 0), (3, 2, 1), (3, 0, 0), (3, 1, 1), (1, 1, 0), (2, 2, 1), (0, 2, 0), (0, 3, 1), (0, 1, 0), (0, 2, 1), (0, 0, 0)

(3, 3, 1), (2, 2, 0), (3, 2, 1), (3, 0, 0), (3, 1, 1), (1, 1, 0), (2, 2, 1), (0, 2, 0), (0, 3, 1), (0, 1, 0), (0, 2, 1), (0, 0, 0)



对重复状态的检查是十分重要的，否则程序会陷入死循环。

c.

尽管状态空间十分简单，但若不对重复状态进行检查，则搜索无法终止，程序会陷入死循环之中。比如，前一次将 1 个传教士和 1 个野人运往目标河岸，紧接着又将 1 个传教士和 1 个野人送回原河岸。因此要不断检查重复状态，并进行去除。