中国科学技术大学
University of Science and Technology of China

人工智能基础习题课
第1次作业

助教 许悦娇

第一次作业

- **3.7** 给出下列问题的初始状态、目标测试、后继函数和耗散函数。选择精确得足以实现的形式化。
- a. 只用四种颜色对平面地图染色,要求每两个相邻的地区不能染成相同的颜色。
- b. 一间屋子里有一只3英尺高的猴子，屋子的房顶上挂着一串香蕉,离地面8英尺。屋子里有两个可叠放起来、可移动、可攀登的3英尺高的箱子。猴子很想得到香蕉。
- d. 有三个水壶，容量分别为12加仑、8加仑和3加仑,还有一个水龙头。可以把壶装满或者倒空，从一个壶倒进另一个壶或者倒在地上。要求量出刚好1加仑水。

第一次作业

- a. 只用四种颜色对平面地图染色,要求每两个相邻的地区不能染成相同的 颜色。
- ➢ 初始状态：无染色地区
- ➢ 目标测试：所有地区均已着色，且没有两块相邻的地区染有同样的颜色
- ➢ 后继函数：给一个未染色地区染色
- ➢ 耗散函数：染色次数

第一次作业

- b. 一间屋子里有一只3英尺高的猴子，屋子的房顶上挂着一串香蕉,离地面 8英尺。屋子里有两个可叠放起来、可移动、可攀登的3英尺高的箱子。 猴子很想得到香蕉。
- ➢ 初始状态：地面上3英尺高的猴子，离地面8英尺高的香蕉，两个可叠放 起来、可移动、可攀登的3英尺高的箱子
- ➢ 目标测试：猴子可以拿到香蕉
- ➢ 后继函数：爬上箱子，爬下箱子，走到一个箱子处，移动一个箱子，叠放 箱子，拿香蕉
- ➢ 耗散函数：动作次数

第一次作业

- d. 有三个水壶，容量分别为12加仑、8加仑和3加仑,还有一个水龙头。可 以把壶装满或者倒空，从一个壶倒进另一个壶或者倒在地上。要求量出刚 好1加仑水。
- ➢ 初始状态：三个水壶都为空（[0, 0, 0]）
- ➢ 目标测试：某个水壶中存在1加仑水（[1, x, y] or [x, 1, y] or [x, y, 1]）
- ➢ 后继函数：装满某个水壶（[x, y, z] -> [12, y, z] or [x, 8, z] or [x, y, 3]）; 倒空某个水壶（[x, y, z] -> [0, y, z] or [x, 0, z] or [x, y, 0]）;将某一水壶 中的水倒入另一个未满的水壶直到水壶中的水被倒完或另一个水壶装满（ eg. [x, y, z] x->y: if x+y < 8 then [0, x+y, z] else [x+y-8, 8, z]）
- ➢ 耗散函数：动作次数

第一次作业

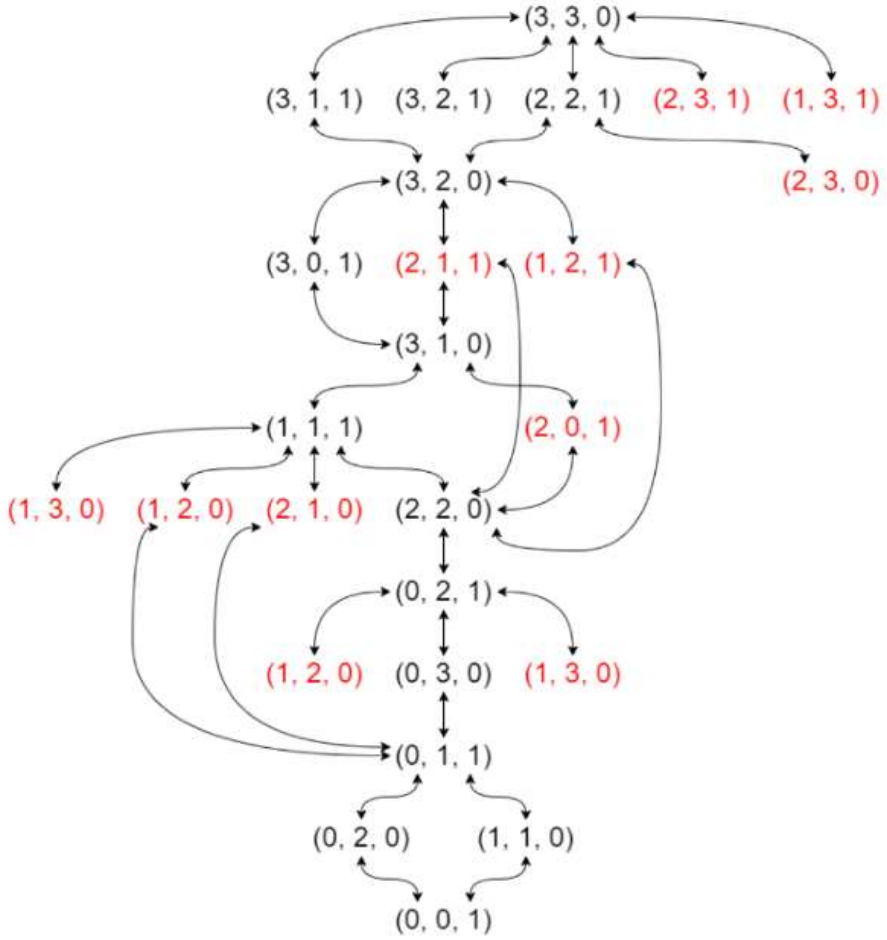
- 3.9 传教士和野人问题通常描述如下：三个传教士和三个野人在河的一边，还有一条能载一个人或者两个人的船。找到一个办法让所有的人都渡到 河的另一岸，要求在任何地方野人数都不能多于传教士的人数(可以只有 野人没有传教士)。这个问题在AI领域中很著名,因为它是第一篇从分析的 观点探讨问题形式化的论文的主题（Amarel, 1968）
- a. 精确地形式化该问题，只描述确保该问题有解所必需的特性。画出该 问题的完全状态空间图。
- ➢ 初始状态：3个传教士和3个野人以及一条船在岸上，另一岸为空
- ➢ 目标状态：3个传教士和3个野人都到达另一岸
- ➢ 耗散函数：动作次数
- ➢ 后继函数：移动1个或者2个人以及一条船到另一岸去。

- ➢ 定义状态表示三元组(A, B, C)，A表示在左岸传教士的数量，B表示在左岸野人的 数量，C为二元指示函数，C=0表示船在左岸，C=1表示船在右岸，则可以形式 化的表示为：
- ➢ 初始状态：[3, 3, 0]
- ➢ 目标状态：[0, 0, 1]（注意，最终结束时船不可能在左岸）
- ➢ 耗散函数：动作次数
- ➢ 后继函数：[a, b, 0] -> [a-1, b, 1] or [a-2, b, 1] or [a, b-1, 1] or [a, b-2, 1] or [a-1, b-1, 1]

  [a, b, 1] -> [a+1, b, 0] or [a+2, b, 0] or [a, b+1, 0] or [a, b+2, 0] or [a+1, b+1, 0]

第一次作业

完全状态空间图：



红色标记的为不合法状态，搜索时达到不合法状态立即回溯

第一次作业

- b.用一个合适的搜索算法实现和最优地求解该问题。检查重复状态是个好主意吗?
- 由于问题规模有限，几乎所有的最优搜索算法都可以。下面给出一个BFS（广度优先搜索算法）的例子。

```
BFS(G)
    // G: the graph of states and actions;
    // G.adj[u]: a set of nodes that can be reached from u by an action;
    // frontier: a FIFO queue;
    // if no solution can be found, return -1; else return cost
    // the optimal path can be infered from u.parent
    cost = 0
    for each node u in G
        u.visited = false
    initial_node.visited = true
    EnQueue(initial_node, frontier)
    while !isEmpty(frontier)
        cost = cost + 1
        u = DeQueue(frontier)
        for each node v in G.f[u] && !v.visited
            if v == goal_node
                return cost
            v.visited = true;
            EnQueue(v, frontier)
    return -1
```

一种可行的方案是：去2个野人，回1个野人，去2个野人，回1个野人，去2个传教士，回1个传教士和1个野人，去2个传教士，回1个野人，去2个野人，回1个野人，去2个野人。

最低路径耗散为11。

本题状态空间有限，不检查重复状态也是可以搜索出结果的，但就效率而言检查重复状态显然是有利于提高算法搜索速度的

第一次作业

- c. 这个问题的状态空间如此简单，你认为为什么人们求解它却很困难？
- 因为几乎所有的移动要么是非法的，要么就需要返回到上一状态，因此状态空间规模将会变大。这也是为什么检查重复状态有助于提高搜索效率的原因

谢谢大家！