# 习题二：多人搜索和马可夫决策过程（共 60 分）

## 1、约束满足：航空管理（10 分）

我们有五架飞机：A，B，C，D 和 E，以及两条跑道：国际和国内。我们想安排每架飞机降落或起飞的时间段和跑道。每个跑道有四个时段：$\{1, 2, 3, 4\}$，可以安排一架飞机的着陆或起飞。我们必须找到一个符合以下约束的起飞降落安排：

- 飞机 B 失去了一个引擎，必须在时段 1 降落。
- 飞机 D 只能在时段 3 期间或之后到达机场降落。
- 飞机 A 燃料不足，最多可以持续到时段 2。
- 飞机 D 必须在飞机 C 起飞之前降落，因为有些乘客要从 D 转乘到 C。
- 没有两架飞机可以为同一条跑道预订相同的时间段。

（1）在变量、域和约束（variable, domain, and contraint）方面完成此问题作为 约束问题的表述。约束条件应该使用数学或逻辑符号表示，而不是用言语。（4 分）

*Variables: A, B, C, D, E for each plane.*

*Domains: a tuple (runway type, time slot) for runway type 2 {international; domestic} and time slot {1, 2, 3, 4}.*

*Constraints: B[1] = 1, D[1]>= 3, A[1] <=2, D[1] < C[1], A != B != C != D != E*

（2）假如我们添加下面两个约束条件，重新使用变量、域和约束（variable, domain, and contraint）来将此问题表述为约束问题。（3 分）

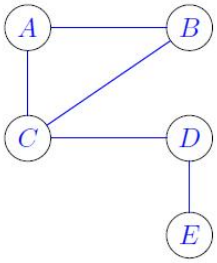
- A、B 和 C 飞机只能使用国际跑道。
- D 和 E 飞机只能使用国内跑道。

*Variables: A, B, C, D, E for each plane.*

*Domains: {1, 2, 3, 4}*

*Explanation of Constraint Graph: A[0]=B[0]=C[0]=international, A != B != C, D[0]=E[0]=domestic, B[1]=1, D[1]>=3, A[1] < 2, D[1] < C[1], D != E*

（3）将问题（2）中的二元（即牵涉到两个变量的）约束用图形方式表达出来，类似于约束问题课件的第八页。（3 分）



## 2、约束满足：数独（10 分）

数独的规则是在每个方格里填 1 到 9 中的某个数字，使得每一行，每一列，每个粗线框里的（3x3）九宫格都包含 1 到 9 九个数字，而且不重复。下面是一个数独的例子。

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 3 | | | | 7 | | 5 | | |
| 1 | | | | | 5 | 6 | | |
| | 4 | 9 | 6 | 1 | 2 | | 7 | |
| 7 | 1 | | | 2 | | 4 | | |
| | | 4 | | 9 | 3 | | 6 | |
| | | 5 | | | | 7 | | |
| 8 | 9 | | | | | | 3 | |
| | | | | | | | | |
| | | | 3 | | | 9 | | 1 |

（1）在变量、域和约束（variable, domain, and constraint）方面完成此问题作为 约束问题的表述。（3 分）

Variables: Vij for each cell where there is no value assigned

Domain: {1, 2, 3, …, 9}

Contraint: Vij != Vik, Vij != Vkj, Vij != Vkl if (i, j) and (k, l) belong to the same 3x3 unit

（2）用原地返回（backtrack）的方法，解决数独问题。请采用 Python 语言回答本题，并计算出上面数独的答案。（6 分）

```
import numpy as np
from itertools import chain

def backtrack(assign, i, j):
    # Backtrack to find Sudoku
    if i == 8 and j == 8: return True

    # Find the next empty spot to fill
    while assign[i,j] != 0:
        # check the next spot
        i, j = i+1, 0 if j == 8 else i, j+1
        if i >= 9: return True

    for v in range(1,10):
        assign[i,j] = v
        if sudoku(assign, i, j):
            res = backtrack(assign, i, j)
            if res: return True
        assign[i,j] = 0
    return False

def sudoku(assign, i0, j0):
    # Check assign at [i,j] satisfies Sudoku conditions
    v = assign[i0,j0]
    for j in chain(range(0,j0), range(j0+1,9)):
        if v == assign[i0,j]:
            return False

    for i in chain(range(0,i0), range(i0+1,9)):
        if v == assign[i,j0]:
            return False
```

```
    i1, j1 = i0//3*3, j0//3*3
    for i in range(i1,i1+3):
        for j in range(j1,j1+3):
            if i == i0 and j == j0:
                continue
            if v == assign[i,j]:
                return False

    return True
```

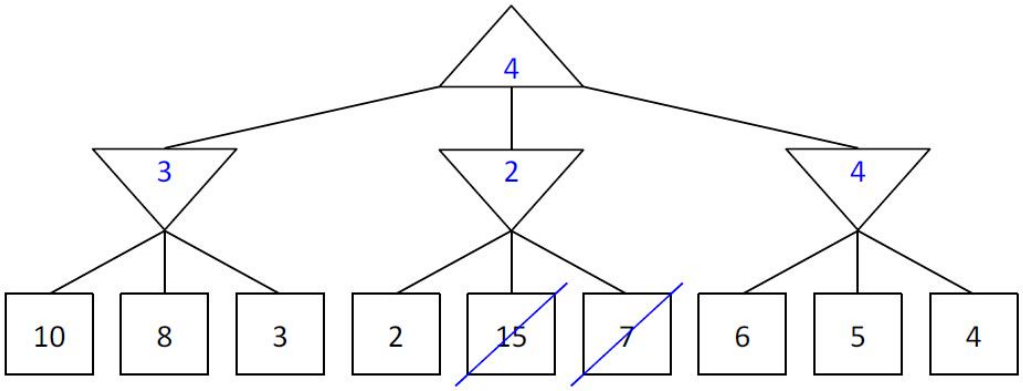
解答：

```
[[3 2 6 9 7 8 5 1 4]
 [1 7 8 4 3 5 6 9 2]
 [5 4 9 6 1 2 3 7 8]
 [7 1 3 5 2 6 4 8 9]
 [2 8 4 7 9 3 1 6 5]
 [9 6 5 8 4 1 7 2 3]
 [8 9 7 1 5 4 2 3 6]
 [4 3 1 2 6 9 8 5 7]
 [6 5 2 3 8 7 9 4 1]]
```

**3、游戏树（20 分）**

考虑下面显示的零和博弈树。指向上方的三角形，例如在顶部节点（根），代表最大化玩家的选择；指向下方的三角形表示最小化玩家的选择。假设两个玩家都以最佳方式行事。

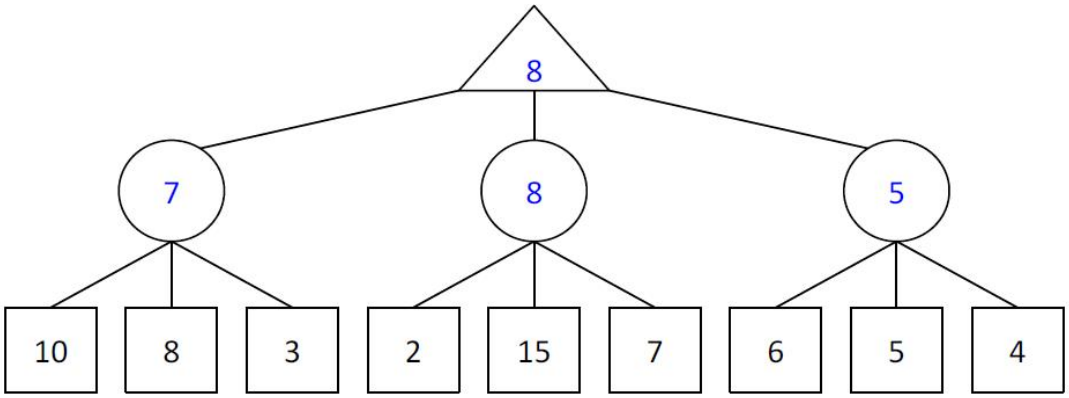
（1）填写每个节点的最小最大值。（5 分）




（2）哪些节点可以通过 alpha-beta 修剪从上面的游戏树中修剪？如果没有节点可以修剪，解释为什么不可以。假设搜索从左到右；选择探望哪个孩子时，选择最左边未访问的孩子。（5 分）

15, 7

（3）考虑相同的零和游戏树，只是现在不再是最小化玩家，而是随机节点（图中的圆圈），它将随机均匀地选择三个值中的一个。填写期望最大值每个节点的值。为方便起见，下面重新绘制了游戏树。（5 分）




（4）哪些节点可以通过 alpha-beta 修剪从上面的游戏树中修剪？如果没有节点可以修剪，解释为什么不可以。（5 分）

*No nodes can be pruned. There will always be the possibility that some leaf further down the branch will have a very high value, which increases the overall average value.*

**4、马可夫决策过程：微型黑杰克（20 分）**

在微型黑杰克的游戏中，您反复抽出一张同样可能是 2、3 或 4 的牌。如果您抽出的牌的总分低于 6，那你可以选择抽牌（Draw）或停止（Stop）。如果您的总分等于或超出 6 分，那游戏结束，您获得 0 分。如果你选择抽牌，那游戏继续。如果您选择停止，您的得分等于您目前牌的总分（最多 5），并且游戏结束。假设没有折扣（discount，$\gamma = 1$）。让我们将此问题表述为具有以下几个状态的 MDP：0，2，3，4，5 和 一个 Done 状态，表示游戏结束。

（1）这个 MDP 的转换函数（transition function）是什么？（7 分）

（2）这个 MDP 的奖励函数（reward function）是什么？（3 分）

$$
\begin{aligned}
T(s, Stop, Done) &= 1 \\
T(0, Draw, s') &= 1/3 \text{ for } s' \in \{2, 3, 4\} \\
T(2, Draw, s') &= 1/3 \text{ for } s' \in \{4, 5, Done\} \\
T(3, Draw, s') &= \begin{matrix} 1/3 \text{ if } s' = 5 \\ 2/3 \text{ if } s' = Done \end{matrix} \\
T(4, Draw, Done) &= 1 \\
T(5, Draw, Done) &= 1 \\
T(s, a, s') &= 0 \text{ otherwise}
\end{aligned}
$$

The reward function is

$$
\begin{aligned}
R(s, Stop, Done) &= s, s \leq 5 \\
R(s, a, s') &= 0 \text{ otherwise}
\end{aligned}
$$

（3）填写下表，其中包含 头 4 次值迭代（value iteration）的结果。（5 分）

| States | 0 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $V_0$ | 0 | 0 | 0 | 0 | 0 |
| $V_1$ | 0 | 2 | 3 | 4 | 5 |
| $V_2$ | 3 | 3 | 3 | 4 | 5 |
| $V_3$ | 10/3 | 3 | 3 | 4 | 5 |
| $V_4$ | 10/3 | 3 | 3 | 4 | 5 |

（3）注意到，上面的值迭代已经收敛了。MDP 的最优策略是什么？（5 分）

| States | 0 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $\pi^*$ | Draw | Draw | Stop | Stop | Stop |