# Ch6 习题

1. 如图所示的地图着色问题共有多少个解？如果是四色有多少个解？如果只有两色呢？

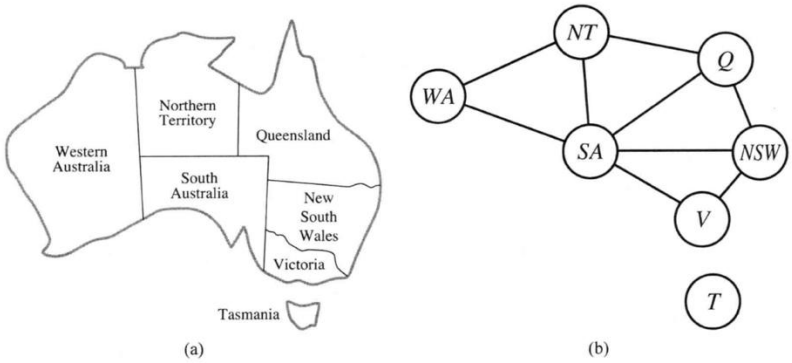


图 6.1 着色问题

（a）澳大利亚的州和行政区。视此地图着色问题为约束满足问题（CSP）。目标是对每个区域分配颜色，使得相邻的区域不同色。（b）将地图着色问题表示成约束图

对于图中显示的澳大利亚地图，每个州都至少与一个其他州相邻，除了塔斯马尼亚（T）。因此，整个图至少需要三种颜色来着色（因为没有任何区域完全被两个区域包围）。如果使用四种颜色，那么解的数量将会是多个，因为一些州可以交换颜色而不违反四色定理。如果只有两种颜色，根据这个地图，无法用两种颜色完成着色，因为至少有三个州相互相邻（例如，南澳大利亚州、西澳大利亚州和北领地），这需要至少三种颜色。因为像上面这样的问题属于 NP 难题，没有任何公式直接来计算一个这样的图有多少个解，所以我们来写个处理简单图的 c++代码来计算
代码核心部分：

```
// 回溯函数来尝试所有可能的颜色组合
void graphColoring(int m, int v) {
    if (v == V) {
        solutions++;
        return;
    }
    for (int c = 1; c <= m; c++) {
        if (isSafe(v, c)) {
            color[v] = c;
            graphColoring(m, v + 1);
            color[v] = 0;
        }
    }
}
```

```cpp
int main() {
    // 为了简化问题，这里直接按图(b)的结构建立图的邻接表
    graph[0].push_back(1); graph[1].push_back(0);
    graph[0].push_back(2); graph[2].push_back(0);
    graph[1].push_back(2); graph[2].push_back(1);
    graph[1].push_back(3); graph[3].push_back(1);
    graph[1].push_back(4); graph[4].push_back(1);
    graph[2].push_back(4); graph[4].push_back(2);
    graph[3].push_back(4); graph[4].push_back(3);
    graph[3].push_back(5); graph[5].push_back(3);
    graph[4].push_back(6); graph[6].push_back(4);

    graphColoring(4, 0);

    cout << "Total solutions: " << solutions << endl;
    return 0;
}
```

运行结果：

```
/tmp/y1NrBtK21K.o
Total solutions: 864
```

2. 考虑下述的逻辑问题：有 5 所不同颜色的房子，住着 5 个来自不同国家的人，每个人都喜欢一种不同牌子的糖果，不同牌子的饮料和不同的宠物。给定下列已知事实，请回答问题“斑马住在哪？哪所房子里的人喜欢喝水？
    - 英国人住在红色房子里。
    - 西班牙人养狗
    - 挪威人住在最左边的第一所房子里
    - 绿房子是象牙色房子的右边邻居
    - 喜欢抽 hershey 牌巧克力的人住在养狐狸的人的旁边
    - 住在黄色房子里的人喜欢 Kit Kats 糖果
    - 挪威人住在蓝房子旁边
    - 喜欢 smarties 糖果的人养了一只蜗牛
    - 喜欢 Snickers 糖果的人喝橘汁
    - 乌克兰人喝茶
    - 日本人喜欢 Milky Ways 糖果
    - 喜欢 Kit Kats 糖果的人住在养马人的隔壁
    - 住在中间房子里的人喜欢喝牛奶
    - 绿房子的主人喝咖啡

把这个问题表示成 CSP 问题，并进行求解。

这段代码使用了 python-constraint 库的 Problem 类来创建一个约束满足问题。我们定义了五所房子的不同属性（颜色、国籍、宠物、糖果、饮料）作为变量，并为每个变量指定了范围（即每个属性可以分配给 1 至 5 号房子中的任意一个）。

接着，我们通过 addConstraint 方法添加了一系列的约束条件，这些条件基于题目中提供的信息。例如，英国人住在红色房子里、挪威人住在第一所房子里、绿房子紧挨着象牙色房子的右边等等。每个约束都是一个函数，它接受对应属性的房子号作为参数，并返回一个布尔值，表示这个约束是否被满足

```
from constraint import Problem

# 创建问题实例
problem = Problem()

# 为五所房子中的每一所分配属性
houses = range(1, 6)
problem.addVariables(["Red", "Green", "Ivory", "Yellow", "Blue"], houses)
problem.addVariables(["English", "Spanish", "Ukrainian", "Norwegian", "Japanese"],
    houses)
problem.addVariables(["Dog", "Snail", "Fox", "Horse", "Zebra"], houses)
problem.addVariables(["Hershey", "KitKat", "Smarties", "Snickers", "MilkyWay"], houses)
problem.addVariables(["OrangeJuice", "Tea", "Coffee", "Milk", "Water"], houses)

```

```
# 添加约束条件
# 1. 英国人住在红色房子里
problem.addConstraint(lambda e, r: e == r, ("English", "Red"))
# 2. 西班牙人养狗
problem.addConstraint(lambda s, d: s == d, ("Spanish", "Dog"))
# 3. 挪威人住在第一所房子里
problem.addConstraint(lambda n: n == 1, ["Norwegian"])
# 4. 绿房子紧挨着象牙色房子的右边
problem.addConstraint(lambda g, i: g == i + 1, ("Green", "Ivory"))
# 5. Hershey糖果的邻居养狐狸
problem.addConstraint(lambda h, f: abs(h - f) == 1, ("Hershey", "Fox"))
# 6. 黄色房子的人喜欢KitKat
problem.addConstraint(lambda y, k: y == k, ("Yellow", "KitKat"))
# 7. 挪威人住在蓝房子旁边
problem.addConstraint(lambda n, b: abs(n - b) == 1, ("Norwegian", "Blue"))
# 8. Smarties糖果的人养蜗牛
problem.addConstraint(lambda s, sn: s == sn, ("Smarties", "Snail"))
# 9. Snickers糖果的人喝橘汁
problem.addConstraint(lambda s, o: s == o, ("Snickers", "OrangeJuice"))
# 10. 乌克兰人喝茶
problem.addConstraint(lambda u, t: u == t, ("Ukrainian", "Tea"))
# 11. 日本人喜欢MilkyWay糖果
problem.addConstraint(lambda j, m: j == m, ("Japanese", "MilkyWay"))
# 12. KitKat糖果的人住在养马的人旁边
problem.addConstraint(lambda k, h: abs(k - h) == 1, ("KitKat", "Horse"))
# 13. 中间的房子喝牛奶
problem.addConstraint(lambda m: m == 3, ["Milk"])
```

```
problem.addConstraint(lambda g, c: g == c, ("Green", "Coffee"))

# 每个属性在不同的房子
problem.addConstraint(lambda *args: len(set(args)) == 5, ["Red", "Green", "Ivory",
    "Yellow", "Blue"])
problem.addConstraint(lambda *args: len(set(args)) == 5, ["English", "Spanish",
    "Ukrainian", "Norwegian", "Japanese"])
problem.addConstraint(lambda *args: len(set(args)) == 5, ["Dog", "Snail", "Fox", "Horse"
    , "Zebra"])
problem.addConstraint(lambda *args: len(set(args)) == 5, ["Hershey", "KitKat",
    "Smarties", "Snickers", "MilkyWay"])
problem.addConstraint(lambda *args: len(set(args)) == 5, ["OrangeJuice", "Tea", "Coffee"
    , "Milk", "Water"])
```

```
51  # 求解CSP问题
52  solution = problem.getSolution()
53
54  # 打印结果
55  print("The Zebra is in the House No.", solution["Zebra"])
56  print("The person who drinks water lives in the
```