6.7 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“斑马住在哪儿？哪所房子里的人喜欢喝水？”：

英国人住在红色的房子里。

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。

挪威人住在蓝色房子旁边。

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。

讨论把这个问题表示成 CSP 的不同方法。你认为哪种比较好，为什么？

答：

第一种表示：

定义 25 个变量，分别代表不同的颜色、国家、糖果、饮料、宠物，变量集合 X 的大小为 25。

对于每一个变量取值为 1-5 的数字，代表门牌号，Di 值域的大小为 5。

对于约束集合，根据给出的描述抽象出即可。比如“英国人住在红色的房子里”，则“英国人”与“红色”的取值相同。“挪威人住在左边的第一所房子里”，即“挪威人”取值为 1。“住在某某旁边”之类的描述则需要列出详细的可能的约束，比如<(x1,x2),[(1,2),(2,1),(2,3),(3,2)……(4,5)]>之类的描述。

在这里也可以把房子门牌号换成人的序号作为值域，定义 30 个变量，分别代表不同的颜色、国家、糖果、饮料、宠物、门牌号，是同样的道理。

约束满足问题包含三个成分 X、D 和 C：

X 是**变量**集合$\{X_1, \cdots, X_n\}$。

D 是**值域**集合$\{D_1, \cdots, D_n\}$，每个变量有自己的值域。

C 是描述变量取值的**约束**集合。

值域 D_i 是由变量 X_i 的可能取值$\{v_1, \cdots, v_k\}$组成的集合。每个约束 C_i 是有序对$<scope, rel>$，其中 $scope$ 是约束中的变量组，rel 则定义了这些变量取值应满足的关系。关系可以显式地列出所有关系元组，也可以是支持如下两个操作的抽象关系：测试一个元组是否为一个关系的成员和枚举所有关系成员。例如，如果 X_1、X_2 的值域均为{A, B}，约束是二者不能取相同值，关系可如下描述：$<(X_1, X_2), [(A, B), (B, A)]>$或$<(X_1, X_2), X_1 \neq X_2>$。

第二种表示：

对于每个房子都设置五个变量，比如房子 1 有$\{X_{11},X_{12},X_{13},X_{14},X_{15}\}$，分别代表颜色、国家、糖果、饮料、宠物，所以变量集合 X 的大小仍然为 25。

对于其中的每个变量，值域大小为 5，分别代表不同的选择。比如X_{11}代表第一个房子的颜色，值域为{红，象牙，蓝，黄，绿}。X_{21}的值域则与X_{11}相同。

对于约束集合，根据给出的描述，比如，“英国人住在红色的房子里”，则

对于一组的变量中，同时选择英国和红色，有五对这样的约束。“挪威人住在左边的第一所房子里”，即X_{1*}取值为挪威人。“住在某某旁边”之类的描述则需要列出详细的可能的约束，更加难以表示。

我认为第一种方法较好。第一种方法很容易可以表示出描述的所有约束，也是最符合书中定义的方法。效率也相对较高，使用门牌号作值域的方法比人的序号更加简洁。第二种方式的约束关系更难表示。

6.15 试将数独游戏当作 CSP 求解，在部分赋值基础上进行搜索，因为大多数人也是这么求解数独问题的。当然，也可能通过在完整赋值上进行局部搜索来求解这些问题。用最少冲突启发式的局部求解器解决数独游戏有哪些优势？

答：

该问题可见于教材 6.4 节，课上未讲。

局部搜索算法（见 4.1 节）对求解许多 CSP 都是很有效的。它们使用完整状态的形式化：初始状态是给每个变量都赋一个值，搜索过程是一次改变一个变量的取值。例如，在八皇后问题（见图 4.3）中，初始状态是 8 个皇后在 8 列上的一个随机布局，然后每步都是选择一个皇后并把它移动到该列的新位置上。典型地，初始布局会违反一些约束。局部搜索的目的就是要消除这些矛盾。[1]

在为变量选择新值的时候，最明显的启发式是选择与其他变量冲突最少的值——**最少冲突**启发式。图 6.8 给出了该算法，图 6.9 则将算法应用于八皇后问题。

```
function MIN-CONFLICTS(csp, max_steps) returns a solution or failure
  inputs: csp, a constraint satisfaction problem
          max_steps, the number of steps allowed before giving up

  current ← an initial complete assignment for csp
  for i = 1 to max_steps do
    if current is a solution for csp then return current
    var ← a randomly chosen conflicted variable from csp.VARIABLES
    value ← the value v for var that minimizes CONFLICTS(var, v, current, csp)
    set var = value in current
  return failure
```

图 6.8　用局部搜索解决 CSP 的 MIN-CONFLICTS 算法

初始状态可以随机选择或者通过贪婪赋值过程依次为每个变量选择最少冲突的值。函数 CONFLICTS 统计当前赋值情况下某值破坏约束的数量

局部搜索算法：初始状态完全赋值，搜索过程是一次改变一个变量的取值。最小冲突启发式是选择与其他变量冲突最少的值进行改变。它对很多 CSP 都有效且效率高，4.1 节中我们学过的所有局部搜索技术都是 CSP 应用的候选，有些已经被证明非常有效。另一个优势是当问题改变时它可以被应用于联机设置，对调度问题来说非常重要，可以以最小的改动来修正。

数独问题既可以在部分赋值（给出的条件）上进行搜索，也可以在完整赋值（随机赋值）上进行局部搜索。对于最小冲突启发式的局部求解器解决数独问题，它有如下优势：

效率：通过集中精力解决冲突最多的变量，可以更快地引导搜索过程朝着可行的解前进。

灵活性：可以适应不同的初始赋值状态，无论是部分赋值还是完整赋值。

完备性：即使在初始赋值不理想的情况下，也能找到一条通往解的路径。

随机性：可以快速生成一个初始状态，有时可以意外地接近一个有效的解。