Experiment1 Zebra Problem

学号：2211279　姓名：张梓恒

一、问题重述

> （简单描述对问题的理解，从问题中抓住主干，必填）

5 个**不同国家**（英国、西班牙、日本、意大利、挪威）且**工作各不相同**（油漆工、摄影师、外交官、小提琴家、医生）的人分别住在一条街上的 **5 所房子**里，每所房子的**颜色不同**（红色、白色、蓝色、黄色、绿色），每个人都有自己养的**不同宠物**（狗、蜗牛、斑马、马、狐狸），喜欢喝**不同的饮料**（矿泉水、牛奶、茶、橘子汁、咖啡）。

根据以下提示，你能告诉我哪所房子里的人养斑马，哪所房子里的人喜欢喝矿泉水吗?

1. 英国人住在红色的房子里
2. 西班牙人养了一条狗
3. 日本人是一个油漆工
4. 意大利人喜欢喝茶
5. 挪威人住在左边的第一个房子里
6. 绿房子在白房子的右边
7. 摄影师养了一只蜗牛
8. 外交官住在黄房子里
9. 中间那个房子的人喜欢喝牛奶
10. 喜欢喝咖啡的人住在绿房子里
11. 挪威人住在蓝色的房子旁边
12. 小提琴家喜欢喝橘子汁
13. 养狐狸的人所住的房子与医生的房子相邻
14. 养马的人所住的房子与外交官的房子相邻

二、设计思想

> （所采用的方法，有无对方法加以改进，该方法有哪些优化方向（参数调整，框架调整，或者指出方法的局限性和常见问题），伪代码，理论结果验证等...**思考题，非必填**）

在斑马问题中，我们选用了约束满足问题（CSP）策略。CSP是解决分配、排列或选择问题的有效方法，通过系统探索所有可能配置来找到符合所有条件的解决方案。

我们定义了代表房屋属性的变量，并根据题目提示建立了约束条件。利用逻辑编程，我们能够以声明性方式描述问题，使求解过程更直观。

CSP的一大优势是其结构化和可靠性，但它也存在局限，如面对大规模或复杂问题时的效率挑战。优化搜索策略，如采用启发式搜索，可减少搜索空间，提高效率。

尽管存在挑战，CSP为斑马问题提供了一个有效的解决框架，通过优化和算法调整，能够处理更复杂的情境。

三、 代码内容

> （能体现解题思路的主要代码，有多个文件或模块可用多个"===="隔开，必填）

```
# Reference: https://blog.csdn.net/weixin_46291251/article/details/122246347
def left(a, b, list):
    return membero((a, b), zip(list, list[1:]))
# zip() 函数用于将可迭代的对象作为参数，将对象中对应的元素打包成一个个元组，然后返回由这些元组组成的列表
# 如果 list = [1, 2, 3, 4]，那么 zip(list, list[1:]) 将产生 [(1, 2), (2, 3), (3, 4)]
# 相当于 a 元素从 list 里面取，b 元素从 list 的第二位开始取，二者构成一个元组
```

```
def next(a, b, list):
        return conde([left(a, b, list)], [right(a, b, list)])
```

有人喜欢喝矿泉水

```
(membero, (var(), var(), '矿泉水', var(), var()), self.units),
```

英国人住在红色的房子里

```
(membero, ('英国人', var(), var(), var(), '红色'), self.units),
```

挪威人住在左边的第一个房子里

```
(eq,
    (('挪威人', var(), var(), var(), var()), var(), var(), var(), var()),
    self.units),
```

养狐狸的人所住的房子与医生的房子相邻

```
(next,
    (var(), var(), var(), '狐狸', var()),
    (var(), '医生', var(), var(), var()),
    self.units),
```

四、 实验结果

> （实验结果，必填）

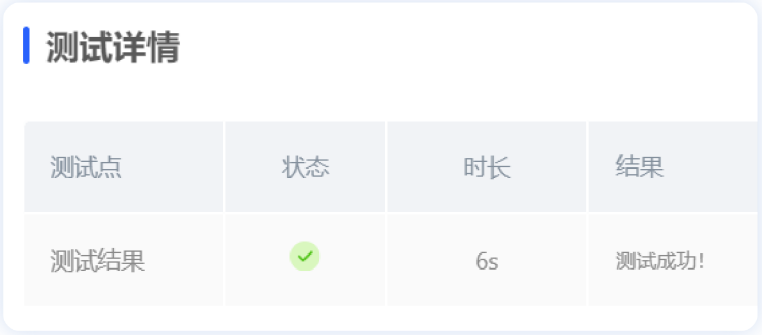
2.3 逻辑推导

[23]:
```
agent = Agent()
solutions = agent.solve()

# 提取解释器的输出
output = [house for house in solutions[0] if '斑马' in house][0][4]
print ('\n{}房子里的人养斑马'.format(output))
output = [house for house in solutions[0] if '矿泉水' in house][0][4]
print ('{}房子里的人喜欢喝矿泉水'.format(output))

# 解释器的输出结果展示
for i in solutions[0]:
    print(i)
```

```
绿色房子里的人养斑马
黄色房子里的人喜欢喝矿泉水
('挪威人', '外交官', '矿泉水', '狐狸', '黄色')
('意大利人', '医生', '茶', '马', '蓝色')
('英国人', '摄影师', '牛奶', '蜗牛', '红色')
('西班牙人', '小提琴家', '橘子汁', '狗', '白色')
('日本人', '油漆工', '咖啡', '斑马', '绿色')
```



五、 总结

> （自评分析（是否达到目标预期，可能改进的方向，实现过程中遇到的困难，从哪些方面可以提升性能，模型的超参数和框架搜索是否合理等），**思考题，非必填**）

- 达到了预期目标
- 可能改进方向：代码是否可以简化重复性的 `(var(), var(), var(), var(), var())` 结构，比如建立新的函数，用位置的顺序数字和字符串来充当参数。
- 多个规则之间记得添加逗号