19024075 陶盛皿 Week8 Assignment

1. 名词解释

    1. 约束传播

        约束传播是特殊的推理：使用约束来减小一个变量的合法取值范围，从而影响到跟此变量有约束关系的另一变量的取值，约束传播与搜索可以交替进行，或者也可以把它作为搜索前的预处理步骤。核心思想是局部相容性。

    2. 路径相容

        路径相容通过观察变量得到隐式约束并以此来加强二元约束。两个变量的集合{Xi，Xj}对于第三个变量Xm是相容的，是指对{Xi，Xj}的每一个赋值{Xi=a，Xj=b}，Xm都有合适的取值同时使得{Xi，Xm}和{Xm，Xj}是相容的。被称为路径相容，是因为这很像是一条从Xi途径Xm到Xj的路径。

    3. 回溯搜索

        回溯搜索用于深度优先搜索中，每次为一个变量选择一个赋值，当没有合法的值可以赋给某变量时就回溯。不断选择未赋值变量，轮流尝试变量值域中的每一个值，试图找到一个解。一旦检测到不相容，BACKTRACK失败，返回上一次调用尝试另一个值。

    4. 最小剩余值启发式

        选择“合法”取值最少的变量，称为最小剩余值启发式。选择最可能很快导致失败的变量，从而对搜索树剪枝，避免无意义的搜索进行。

    5. 度启发式

        度启发式通过选择与其他未赋值变量约束最多的变量来试图降低未来的分支因子。

    6. 最少约束值启发式

        最少约束启发式优先选择的值是给邻居变量留下更多的选择。

    7. 智能回溯

        智能回溯是退回到可能解决当前问题的变量，回跳方法回溯到冲突集中时间最近的赋值，如果没有找到冲突集的合法值就按照失败标记返回冲突集中时间最近的元素。

2. 给出两种方法，将澳大利亚地图问题的约束图简化成树。

```
# Backtracking
from csp import Constraint, CSP
from typing import Dict, List, Optional

class MapColoringConstraint(Constraint[str, str]):
    def __init__(self, place1: str, place2: str) -> None:
        super().__init__([place1, place2])
        self.place1 = place1
        self.place2 = place2

    def satisfied(self, assignment: Dict[str, str]) -> bool:
        # if either place is not in the assignment, then it is not
        # yet possible for their colors to be conflicting
        if self.place1 not in assignment or self.place2 not in assignment:
            return True
        # check the color assigned to place1 is not the same as the
        # color assigned to place2
        return assignment[self.place1] != assignment[self.place2]
```

```python
if __name__ == "__main__":
    variables: List[str] = ["Western Australia", "Northern Territory", "South
Australia",
                            "Queensland", "New South Wales", "Victoria",
"Tasmania"]
    domains: Dict[str, List[str]] = {}
    for variable in variables:
        domains[variable] = ["red", "green", "blue"]
    csp: CSP[str, str] = CSP(variables, domains)
    # 弧相容
    csp.add_constraint(MapColoringConstraint("Western Australia", "Northern
Territory"))
    csp.add_constraint(MapColoringConstraint("Western Australia", "South
Australia"))
    csp.add_constraint(MapColoringConstraint("South Australia", "Northern
Territory"))
    csp.add_constraint(MapColoringConstraint("Queensland", "Northern
Territory"))
    csp.add_constraint(MapColoringConstraint("Queensland", "South Australia"))
    csp.add_constraint(MapColoringConstraint("Queensland", "New South Wales"))
    csp.add_constraint(MapColoringConstraint("New South Wales", "South
Australia"))
    csp.add_constraint(MapColoringConstraint("Victoria", "South Australia"))
    csp.add_constraint(MapColoringConstraint("Victoria", "New South Wales"))
    csp.add_constraint(MapColoringConstraint("Victoria", "Tasmania"))
    # 回溯搜索
    solution: Optional[Dict[str, str]] = csp.backtracking_search()
    if solution is None:
        print("No solution found")
    else:
        print(solution)

# => {'Western Australia': 'red', 'Northern Territory': 'green', 'South
# Australia': 'blue', 'Queensland': 'red', 'New South Wales': 'green',
# 'Victoria': 'red', 'Tasmania': 'green'}
```