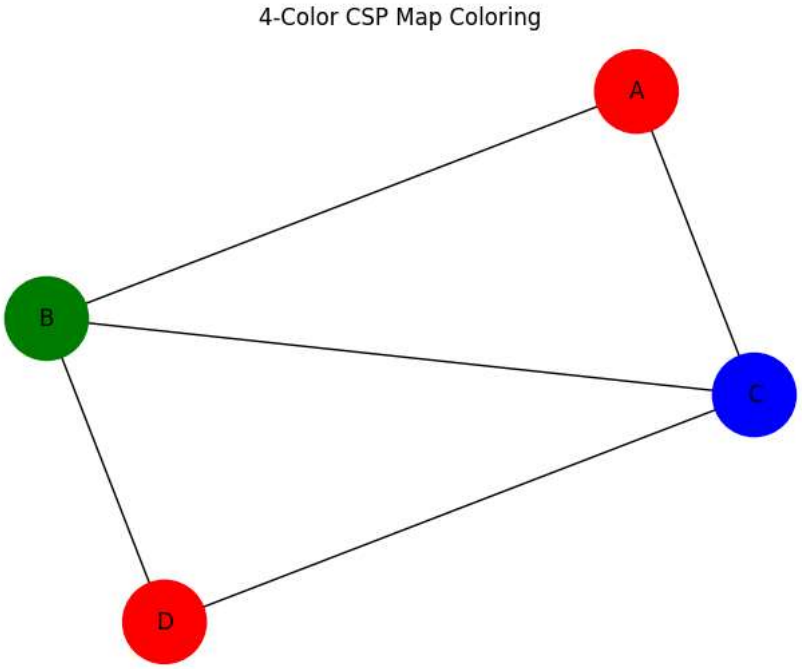
```
import networkx as nx
import matplotlib.pyplot as plt

class ConstraintSatisfactionProblem:
    def __init__(self, variables, domains, neighbors):
        self.variables = variables
        self.domains = domains
        self.neighbors = neighbors

    def is_consistent(self, variable, value, assignment):
        for neighbor in self.neighbors[variable]:
            if neighbor in assignment and assignment[neighbor] == value:
                return False
        return True

    def backtrack(self, assignment):
        if len(assignment) == len(self.variables):
            return assignment

        var = self.select_unassigned_variable(assignment)

        for value in self.domains[var]:
            if self.is_consistent(var, value, assignment):
                assignment[var] = value
                result = self.backtrack(assignment)
                if result is not None:
                    return result
                assignment.pop(var)

        return None

    def select_unassigned_variable(self, assignment):
        return next(v for v in self.variables if v not in assignment)



variables = ['A', 'B', 'C', 'D']

domains = {
    var: ['Red', 'Green', 'Blue', 'Yellow']
    for var in variables
}

neighbors = {
    'A': ['B', 'C'],
    'B': ['A', 'C', 'D'],
    'C': ['A', 'B', 'D'],
    'D': ['B', 'C']
}

csp = ConstraintSatisfactionProblem(variables, domains, neighbors)
solution = csp.backtrack({})

print("Solution:", solution)


G = nx.Graph()

for var in variables:
    G.add_node(var)

for var in neighbors:
    for neighbor in neighbors[var]:
        G.add_edge(var, neighbor)

pos = nx.spring_layout(G)

node_colors = [solution[node].lower() for node in G.nodes()]

plt.figure()
nx.draw(G, pos, with_labels=True, node_color=node_colors, node_size=2000)

plt.title("4-Color CSP Map Coloring")
plt.show()
```

```
Solution: {'A': 'Red', 'B': 'Green', 'C': 'Blue', 'D': 'Red'}
```




```
Start coding or generate with AI.
```