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

exams = ["Exam1", "Exam2", "Exam3", "Exam4"]

conflicts = [
    ("Exam1", "Exam2"),
    ("Exam1", "Exam3"),
    ("Exam1", "Exam4"),
    ("Exam2", "Exam3"),
    ("Exam2", "Exam4"),
    ("Exam3", "Exam4"),
]

colors = ["red", "green", "blue", "yellow"]

def is_safe(node, color, assignment, graph):
    for neighbor in graph.neighbors(node):
        if neighbor in assignment and assignment[neighbor] == color:
            return False
    return True

def backtrack(graph, nodes, assignment):
    if len(assignment) == len(nodes):
        return assignment

    node = nodes[len(assignment)]

    for color in colors:
        if is_safe(node, color, assignment, graph):
            assignment[node] = color
            result = backtrack(graph, nodes, assignment)
            if result:
                return result
            del assignment[node]
    return None

G = nx.Graph()
G.add_nodes_from(exams)
G.add_edges_from(conflicts)

solution = backtrack(G, exams, {})

print("Proctor Assignment:")
for exam in solution:
    print(exam, "->", solution[exam])

pos = nx.spring_layout(G)
nx.draw(G, pos,
        with_labels=True,
        node_color=[solution[node] for node in G.nodes()],
        node_size=3000,
        font_weight="bold")

plt.title("CSP Backtracking - 4 Colour Proctor Assignment")
plt.show()
```

```
Proctor Assignment:
Exam1 -> red
Exam2 -> green
Exam3 -> blue
Exam4 -> yellow
```