CS4408 Learning Journal 4

For this assignment, I implemented a map coloring solution using a Constraint Satisfaction Problem (CSP) backtracking algorithm in Python. The goal was to assign colors to a map’s regions so that no two adjacent regions share the same color, using a fixed order of seven colors. My approach involved reading a graph representation from a file, constructing a data structure that maps each vertex to its adjacent vertices, and then recursively attempting to assign colors to each vertex while obeying the constraints.

I began by analyzing the assignment requirements carefully and breaking down the task into manageable steps. First, I focused on designing a method to read the graph data from a text file, ensuring that the program could handle a range of scenarios—from small graphs with just a few vertices to larger graphs with up to 20 vertices. I chose to use a dictionary to represent the graph because it allowed efficient lookup of each vertex's neighbors. Next, I implemented the backtracking algorithm. I was particularly interested in the backtracking approach because it’s a clear example of how recursion and constraint propagation can be applied to solve combinatorial problems, as discussed by Russell and Norvig (2010).

During the implementation, I experienced a mix of excitement and challenge. Initially, I felt confident about the structure of my program, but as I tested the code, especially with edge cases like a complete graph (K8) that cannot be colored with only seven colors, I encountered some difficulties. Debugging the recursive calls and ensuring the algorithm correctly backtracked when a conflict was encountered required careful thought and numerous print statements to trace the flow of execution. However, once I identified the problem areas, I was able to refine my logic, and the satisfaction of seeing a successful run was very rewarding.

I actively participated in discussion forums throughout the process. Feedback from peers helped me understand different ways to optimize the backtracking function and manage recursive calls more efficiently. For example, one colleague suggested reordering the vertices based on the number of adjacent vertices (the degree heuristic), although I eventually decided to keep the order as per the assignment’s specification to maintain clarity and conformity. These interactions not only improved my code but also deepened my understanding of CSP and algorithm design.

My feelings throughout this assignment evolved from initial curiosity to a sense of achievement. I appreciated the problem-solving aspect of the task, especially when the code finally handled both typical cases and the more complex scenarios where seven colors weren’t enough. The challenge pushed me to think critically and adopt a systematic approach to debugging and testing. In hindsight, the process taught me valuable lessons about the importance of clear problem decomposition, testing with varied scenarios, and iterative refinement—a mindset that is crucial for both academic and professional software development.

I also learned that writing clean, maintainable code is just as important as solving the problem. I took extra care to document my code and structure it in a way that made each component understandable. This reflective exercise not only reinforced my technical skills but also highlighted the importance of self-assessment and iterative improvement in my learning journey.

In summary, this assignment was a practical exercise in implementing a backtracking algorithm to solve a CSP. It challenged me to refine my coding skills, engage with peers for feedback, and critically reflect on my learning process. The experience has enhanced my understanding of recursive algorithms and provided me with strategies for tackling complex programming problems in the future.

**Reference**

Russell, S., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach* (3rd ed.). Prentice Hall.