Ghulam Ishaq Khan Institute of Engineering Sciences and Technology
Department of Computer Science

Course Information

Course Code: CS 351L
Course Title: Artificial Intelligence Lab
Instructor: Mr. Usama Arshad, PhD CS
Program: BS Cybersecurity
Semester: 5th
Reference for Lab Resources:
[CS 351L - AI Lab GitHub Repository]
(https://github.com/usamajanjua9/CS-351L---AI-Lab-)

Lab Task Details

Lab Task: 03

Lab Title: Introduction to Constraint Satisfaction Problems (CSP)

Assigned Date: 18th September 2024

Submission Deadline: 25th September 2024

Task Type: Individual

Submission Instructions

- Make a public repository on GitHub with following name:
  CS 351L - AI Lab GitHub Repository_Your_reg_no.

- Submit each completed lab task on repository and share the link to my email with screenshots of output.
  usama.arshad@giki.edu.pk
- File Naming Convention: [YourName]_CS351L_Lab02.ipynb

Late Submissions: Will incur a deduction of marks unless approved in advance by the instructor.

Task Overview

In this lab, you are provided with a graph coloring problem, implemented using backtracking with two heuristic options: **Minimum Remaining Values (MRV)** and **Degree Heuristic**.

Your task is to come up with a **new scenario** based on real-world graph coloring applications, such as:

- **Scheduling problems** (e.g., exam scheduling where rooms and time slots need to be assigned without conflicts).
- **Map coloring** (e.g., coloring regions of a map where no two adjacent regions can have the same color).
- **Resource allocation** (e.g., allocating limited resources like CPU tasks).

Steps to Follow:

1. **Choose a real-world scenario**: Select a scenario where graph coloring can be applied. For example:
    - You may choose a university timetable scheduling scenario where no two exams should be in the same time slot for students in common.
    - Alternatively, consider map coloring where neighboring countries should have different colors.
2. **Modify the Code**:
    - Adapt the provided code to fit your chosen scenario. You cannot create a new codebase from scratch but must modify the existing code accordingly.
    - Add relevant data (nodes and edges) to represent your chosen scenario.
    - Modify any visualization aspects to fit your scenario better (e.g., change labels to represent rooms, countries, etc.).
3. **Visualize each step**:
    - Ensure the step-by-step visualization is properly shown.
    - The graph coloring process must be displayed for each iteration with clear visuals.
4. **Documentation**:
    - Document your scenario clearly in the comments at the top of your script.
    - Provide explanations of any modifications made to the code, along with explanations of why those changes were necessary.

Requirements:

- **Modular Code**: Ensure your code is well-structured and modular. Each function should serve a clear purpose, with no unnecessary duplication.
- **Clear Output**: Include print statements to display the current step of the algorithm, along with color assignments and conflict checking.

- **Heuristic Choice**: Clearly indicate whether you're using MRV, Degree Heuristic, or sequential assignment. Ensure your reasoning behind the chosen heuristic is explained in your code comments.

Example:

You could transform the original random graph into an **exam scheduling problem**. The graph nodes would represent **courses**, and the edges between them would represent **students enrolled in both courses**. Each color represents a **time slot**, and your goal is to assign courses to different time slots so that no two conflicting courses (connected by an edge) are assigned the same time slot.

```
python
Copy code
# Example: A new scenario for graph coloring applied to exam scheduling
```

Evaluation Criteria:

1. **Creativity of Scenario**: Does your scenario effectively demonstrate graph coloring in a real-world situation?
2. **Code Modifications**: Are your code modifications meaningful and well-implemented?
3. **Visualization**: Is the graph coloring process clearly visualized step by step?
4. **Comments & Explanation**: Are your comments and explanations clear and insightful?
5. **Modularity**: Is your code modular and well-structured?

----------to err is human----------