

1. Project Description

This project is an interactive tool designed to solve the map coloring problem using advanced algorithms such as the Four Color Theorem, Constraint satisfaction problem Algorithm (CSP), Greedy Coloring Algorithm and Min-conflicts Heuristics. Regions or nodes are represented as a graph, and different colors are assigned to each region such that no two adjacent regions share the same color. The project provides an easy-to-use graphical user interface (GUI) that allows users to input data about regions and their borders, and automatically generates the coloring.

2. Tools and Libraries Used

Python: Primary programming language.

Tkinter: For creating the graphical user interface (GUI).

NetworkX: For graph representation and data processing.

Matplotlib: For visualizing graphs and maps within the GUI.

MessageBox: For displaying alerts and notifications.

FigureCanvasTkAgg: To integrate Matplotlib plots into the Tkinter interface.

➤ **PEAS** (Performance, Environment, Actuators, Sensors)

Component	Description
Performance	<ul style="list-style-type: none">- Safe and accurate coloring.- Efficient use of minimum colors.- Intuitive GUI.- Fast processing.
Environment	<ul style="list-style-type: none">- Regions and borders entered by the user.- Graph structure representing regions and adjacency.- Predefined color palette.
Actuators	<ul style="list-style-type: none">- Buttons for adding regions, borders, and generating the map.- Coloring algorithm to assign colors.- Real-time visualization.
Sensors	<ul style="list-style-type: none">- Text fields for region and border input.- Button clicks detected by the system.- Graph data processed from user inputs.

➤ **ODESDA** (Observable, Deterministic, Episodic, Static, Discrete, Agent)

Component	Description
O	Fully observable
D	Deterministic
E	Sequential
S	Static
D	Discrete
A	Single

➤ **Agent Type:** Goal-Based Agents

➤ **Team Members:**

- Mahmoud Reda AbdelFattah AbdelRazak.
- Mahmoud Mohamed Sayed Ahmed Sanad.
- Farah Hesham Mokhtar Akl.

Criteria	Backtracking	Greedy Coloring	Min-Conflicts
Optimality	✓ Optimal	✗ Not Optimal	✗ Not Optimal
Speed	✗ Slow	✓ Very fast	✓ Fast to moderate
Time Complexity	$O(k^n)$	$O(V^2 + E)$	$O(\text{max_steps} \times d)$
Scalability (Large Graphs)	✗ Not suitable	✗ Not ideal	✓ Suitable
Minimum Color Guarantee	✓ Yes	✗ No	✗ Not always
Best Use Case	Small graphs where optimality matters	Simple maps requiring fast results	Complex maps with acceptable approximations
Fails when	✗ Graph has many nodes or too few colors → becomes too slow.	✗ Graph has a bad node order or is highly connected, requiring more colors than needed.	✗ Gets stuck in conflict loops or local minima without progress.