PROJECT 2: Coloring Puzzle

DESCRIPTIONS:

You are asked to build a coloring puzzle solver by using the first order logic to CNF as described below:

Given a matrix of size $m \times n$, where each cell will be a non-negative integer or zero (empty cell). Each cell is considered to be adjacent to itself and 8 surrounding cells.

Your puzzle needs to color all the cells of the matrix with either blue or red, so that the number inside each cell corresponds to the number of blue squares adjacent to that cell (see Figure 1)

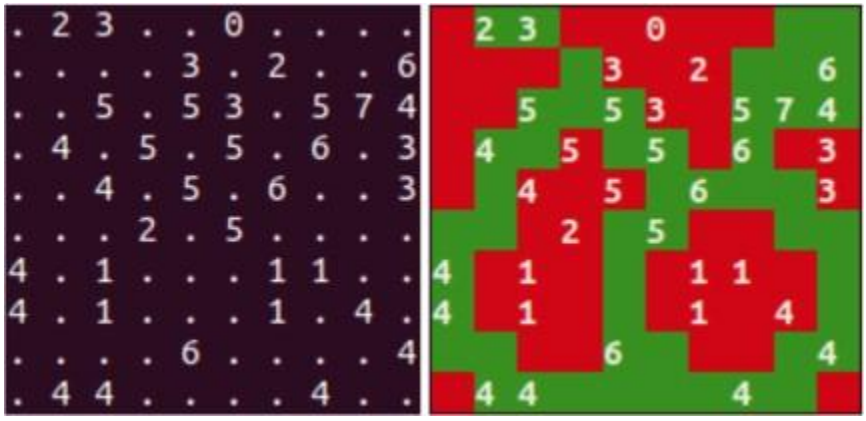

Figure 1 An example of input matrix (left) and output matrix (right)

In order to solve this problem, you can consider some steps:

1. A logical variable is assigned to each cell of the matrix (If the logical variable of that cell is True, it will be colored blue, otherwise it will be red)
2. (Report) Write constraints for cells containing numbers to obtain a set of constraint clauses in CNF (note that you need to remove duplicate clauses)
3. (Implement) Generate CNFs automatically.
4. (Implement) Using the ***pysat*** library to find the value for each variable and infer the result.
5. (Implement) Apply A*, students asked to design the application with the interface as shown that allows users to browse the input file as well as enter the delay time for each step (default is 0.5s). See Figure 2.

6. (Implement) Program brute-force and backtracking algorithm to compare their speed (by measuring running time which is how long it takes for a computer to perform a specific task) and their performance with A* .

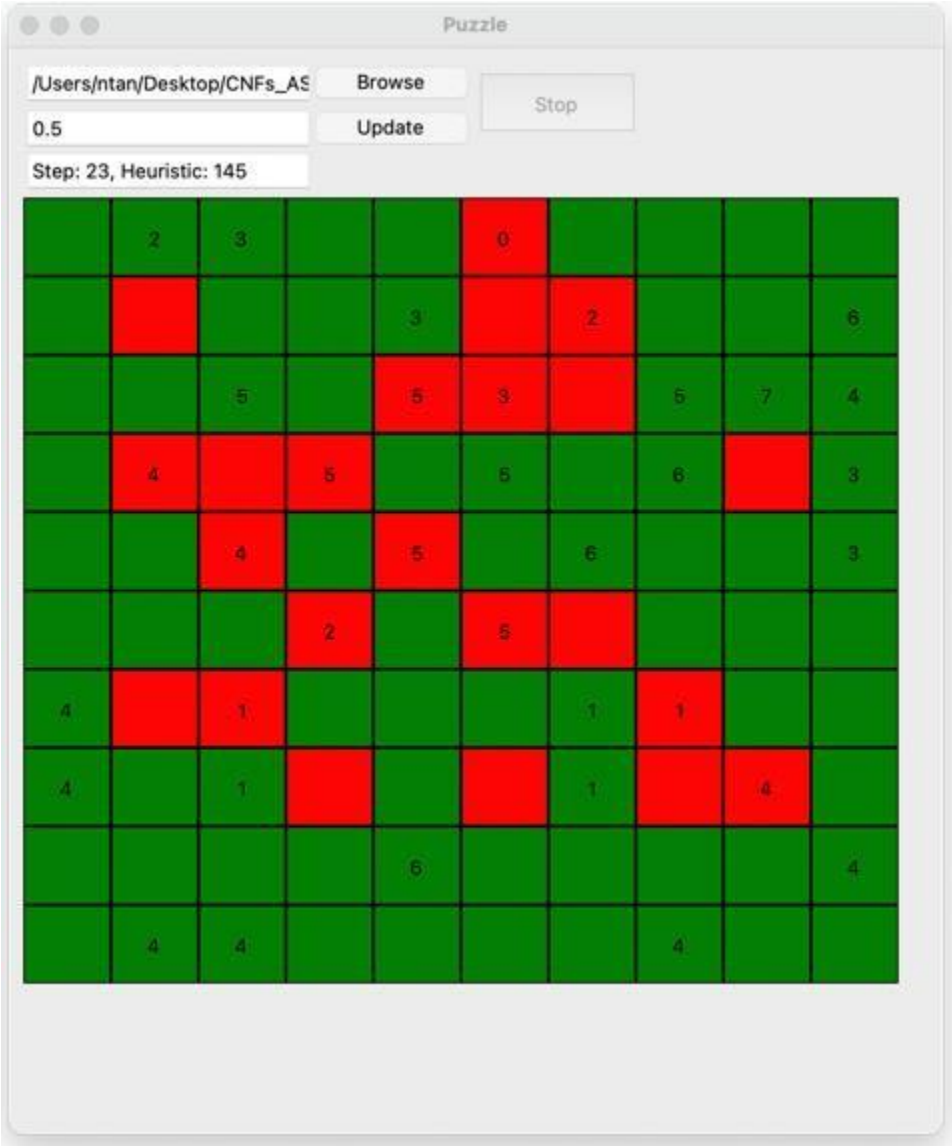

Figure 2 A Sample GUI for coloring puzzle

- When the user presses the Start button, runs the A* algorithm, at each step, the state of the current State is displayed on the form, the step order and the current State's heuristic value are also displayed on the form.
- Students can use Python's tkinter library to create GUI

REQUIREMENTS:

1. Collaboration Policy

You can work individually or in groups. Each group has maximum 3 students.

2. Submitted documents

a. Group Roster: Who are the group members? What did each person do? Completion of how many percent of the work is assigned?

b. Source code: entire your source code with running instructions.

c. Video demo: Video that recording the process of running the tests and the results of your program.

d. Report:

The following is a suggested structure for the report:

- Introduction: this section introduces your problem, and the overall plan for approaching your problem.
- Approach: This section details the framework of your project. Be specific, which means you might want to include equations, figures, plots, etc. In this section, you have to describe the logical principles for generating CNFs, the methods or tools you use and how you solve the problem. You should describe the algorithm you use as well as the highlights of your solution.
- Experiment: In this section, you can present experimental conditions, obtained results as well as limitations of your proposed solution. You can also compare the results of different solutions (You are encouraged to use tables, and charts to show the difference between methods)
- Conclusion: What have you learned? What would you like to share about approaching this problem?
- References: You need to state all references used in your project. This is absolutely necessary. Note that, without this list, your work may be considered as plagiarism.

Grading Policy:

We have some rubric for assessing your work.

No.	Criteria	%
1	**Solution description** Describe the logical principles for generating CNFs correctly.	30%
2	**Implementation:** Generate CNFs automatically (10%) Use pysat library to solve CNFs correctly (10%) Implement A* to solve CNFs without a library 10%) Program brute-force algorithm to compare with A* (speed) (5%) Program backtracking algorithm to compare with A* (speed) (5%) Graphic interface (10%)	40%-50%
3	**Documents and other resources that you need to write and analysis in your report** - Write your answer for step #2 - Give at least 5 testcases with difference sizes (5x5, 9x9, 11x11, 15x15, 20x20…) to check your solution (10%) - Thoroughness in analysis and experimentation (5%) - Comparing results and performance (5%) - Comply with the regulations of submission requirements (10%)	30%

Note:

- *0 points will be given for the same content submissions.*
- *Only submissions on the Moodle link are accepted.*
- *All submission files are packaged in a compressed file. The file name submitted must be set according to the following syntax: StudentID1_StudentID2.zip*
- *Any plagiarism will be get 0 point.*