University of Science, VNU-HCM

Faculty of Information Technology

ꕥ

REPORT

COURSE: ARTIFICIAL INTELLIGENCE

PROJECT 02: COLORING PUZZLE

Theory Lecurer:	**Nguyen Ngoc Thao**	
Assistant Lecturer:	**Ho Thi Thanh Tuyen**	
List Student:	**Hoang Huu Minh An**	**20127102**
	Nguyen Vo Minh Tri	**20127364**
	Phan Minh Xuan	**20127395**

ꕥ

Contents

I. GROUP INFORMATION

ID	Name	Mail
20127102	Hoang Huu Minh An	20127102@student.hcmus.edu.vn
20127364	Nguyen Vo Minh Tri	20127364@student.hcmus.edu.vn
20127395	Phan Minh Xuan	20127395@student.hcmus.edu.vn

II. ABOUT REQUIREMENTS

1. Problem

We 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.

2. Approaching

+ Analyze how to define CNFs.
+ Design UIs.
+ Use the pysat library and CNFs to solve the problem.
+ Design brute-force and backtracking algorithms.
+ Implement brute-force and backtracking.
+ Connect the logic and how the UI works to emit the application.

III. COMPLETE

No	Content	Complete	Notes
1	Describe the logical principlesfor generating CNFs correctly.	100%	Done
2	Generate CNFs automatically.	100%	Done
3	Use the pysat library to solve CNFs correctly.	100%	Done
4	Use A* to solve CNFs without a library.	80%	We still some shortcomings.
5	Program brute-force algorithm.	100%	Done
6	Program backtracking algorithm.	100%	Done

7	Graphic interface.	100%	Done
8	Give 5 test cases withdifferent sizes.	100%	Done
9	Comparing result and performance.	100%	Done
10	Comply with the regulations of submission requirements.	100%	Done

IV. DESCRIBE CNFs DETAIL

- *Example:*

- Value of each cell start from 1 to 9.

2		
	3	

- Clauses that color red for the cells adjacent to a cell are a set of boolean expressions.
- At the position of No.1 has a value = 2, which means must be 2 adjacent cells it is colored green.
- KB(2):

  Case 1:
  - Normal clause: (1 ∧ 2) ⇒ (¬4 ∧ ¬5)
  - CNF: (¬1 ∨ ¬2 ∨ ¬4) ∧ (¬1 ∨ ¬2 ∨ ¬5)
  - Mean: (1 ∧ 2) is green, No.4 and No.5 are red

  Case 2:
  - Normal clause: (1 ∧ 4) ⇒ (¬2 ∧ ¬5)
  - CNF: (¬1 ∨ ¬4 ∨ ¬2) ∧ (¬1 ∨ ¬2 ∨ ¬5)
  - Mean: (1 ∧ 4) is green, No.2 and No.5 are red

  Case 3:
  - Normal clause: (1 ∧ 5) ⇒ (¬2 ∧ ¬4)
  - CNF: (¬1 ∨ ¬5 ∨ ¬2) ∧ (¬1 ∨ ¬5 ∨ ¬4)

  - Mean: (1 ∧ 5) is green, No.2 and No.4 are red
- At the postion of No.5 has a value= 3, which means must be 3 adjacent cells it is colored green.
- KB(3):
  Case 1:
  - Normal clause: (1 ∧ 2 ∧ 5) ⇒ (¬3 ∧ ¬4 ¬6 ∧ ¬7 ¬8 ∧ ¬9)
  - CNF:(¬1 ∨ ¬2 ∨ ¬5 ∨ ¬3) ∧ (¬1 ∨ ¬2 ∨ ¬5 ∨ ¬4) ∧ (¬1 ∨ ¬2 ∨ ¬5 ∨ ¬6) ∧ (¬1 ∨ ¬2 ∨ ¬5 ∨ ¬7) ∧ (¬1 ∨ ¬2 ∨ ¬5 ∨ ¬8) ∧ (¬1 ∨ ¬2 ∨ ¬5 ∨ ¬9)
  - Mean: (1 ∧ 2 ∧ 5) is green, No.3, No.4, No.6, No.7, No.8, and No.9 are red

  Case 2:
  - Normal clause: (1 ∧ 4 ∧ 5) ⇒ (¬2 ∧ ¬3 ¬6 ∧ ¬7 ¬8 ∧ ¬9)
  - CNF:(¬1 ∨ ¬4 ∨ ¬5 ∨ ¬2) ∧ (¬1 ∨ ¬4 ∨ ¬5 ∨ ¬3) ∧ (¬1 ∨ ¬4 ∨ ¬5 ∨ ¬6) ∧ (¬1 ∨ ¬4 ∨ ¬5 ∨ ¬7) ∧ (¬1 ∨ ¬4 ∨ ¬5 ∨ ¬8) ∧ (¬1 ∨ ¬4 ∨ ¬5 ∨ ¬9)
  - Mean: (1 ∧ 4 ∧ 5) is green, No.2, No.3, No.6, No.7, No.8, and No.9 are red

  In another case in KB (3), we do similar to case 1 and case 2.

➢ **Conclution:**

- Clauses that color green for the cells adjacent to a cell is all the combinations (k-combination of set S) is a set of conjunction of a set of disjunction of a combination:
  - S is a set of all cells adjacent to that cell.
  - N is the number of cells adjacent to that cell (includes ifself).
  - M is the number of cells that need to color green.
  - K is the difference between n and m.

We have KB (2)
KB (C) = KB (2) ∧ KB (3)
KB (D) = KB (C) ∧ KB (4) (with KB(4) in the case knowledge base of the next cell which has value).
.
.
.
.
Finally, we will find the CNFs clause's correct requirements.

V. PYSAT

- Using Glucose 3 in pysat.solver lib.
- Using add_clause( ) to add each clause in a set of CNF clause.
- Using solve( ) function to solve the problem. This function will return True or return False.
- Finally, use get_model( ) to get a value of all variables in CNF clause. Accordingly, it is possible to derive a matrix of numbers 0 and 1, representing the colors, 1 being green and 1 being red.
-

VI. A* ALGORITHM

- Algorithm A* is an optimization algorithm that uses heuristic evaluation to find the next move. So, the A* algorithm is heavily dependent on the heuristic functions. If the difference in heuristic estimates is large enough, it will lead to false results.
- Heristic function is built based on a total number of cell was colored by blue with the number of red cell was valid colored and subtract the number of cells is visual in each clause.

VII. BACKTRACKING

- Loop through all cells that contain a number from left to right and top to bottom.
- At each cell, get all the combinations (k-combination of a set S):
  + S is a set of all the red cells adjacent to that cell (includes itself).
  + k is the number of remaining cells that need to color green.
- Coloring all the cells in a combination (set of numbers) to green and jump to the next cell that contains a number.
- If the current cell has the number of the green cells adjacent to itself that is greater than the number inside or looped through all combinations of that cell, then the backtracking function will go back to the previous cell and try another combination.
- The backtracking function will run until the completion of coloring the last cell that contains a number.

VIII. BRUTEFORCE

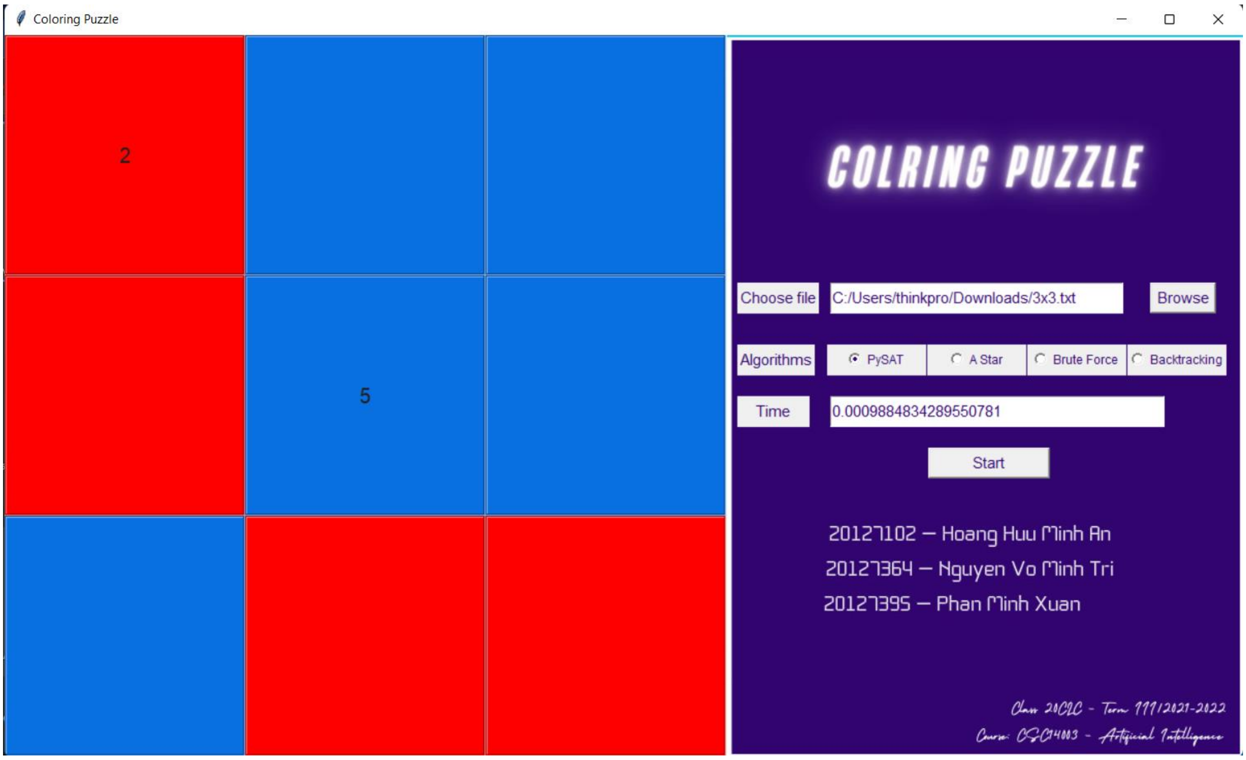
- Coloring all cells in a combination to green and go to the next cell that contains a number.
- If the current cell has the number of the green cells adjoining to itself that is higher than the wide variety interior or looped via all combinations of that cell, then the brute pressure function will go back to the preceding cell and try some other combination.
- The brute force feature will run till the completion of coloring the last cell that includes a variety.

IX. DEMO

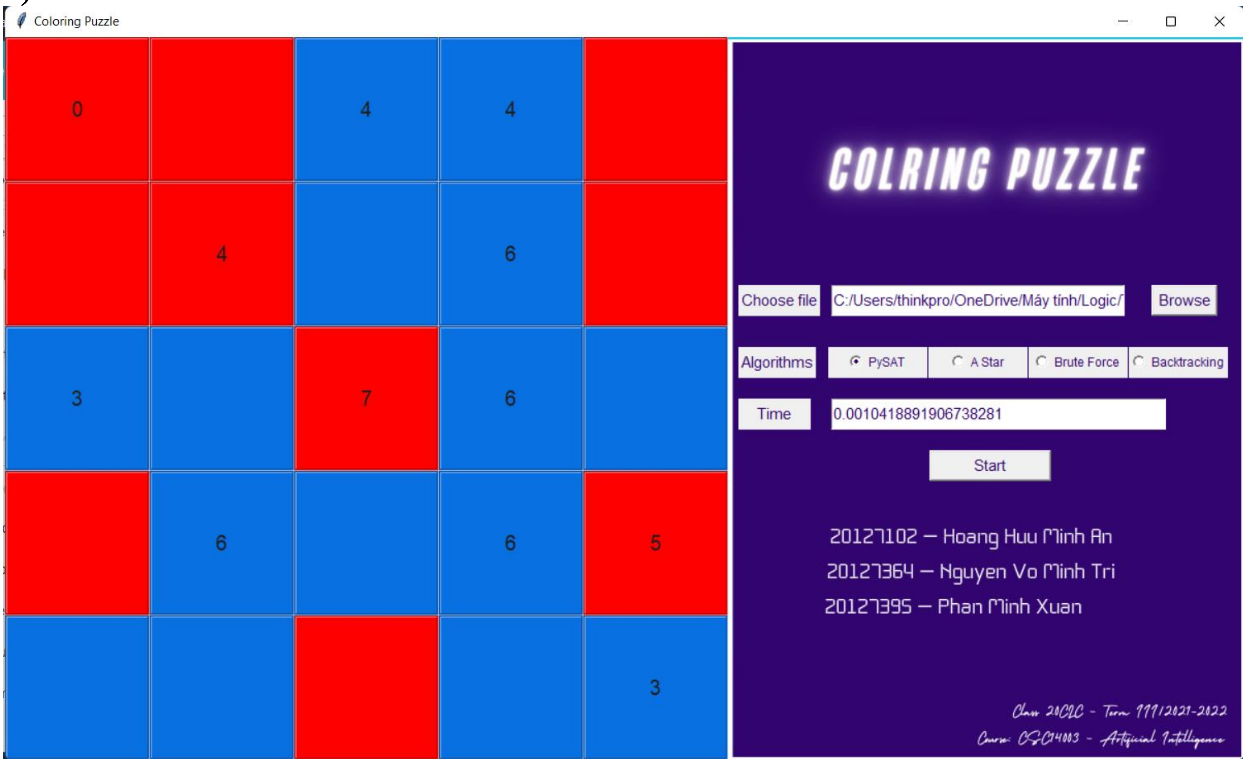
❖ User Interface

1. Using PySat

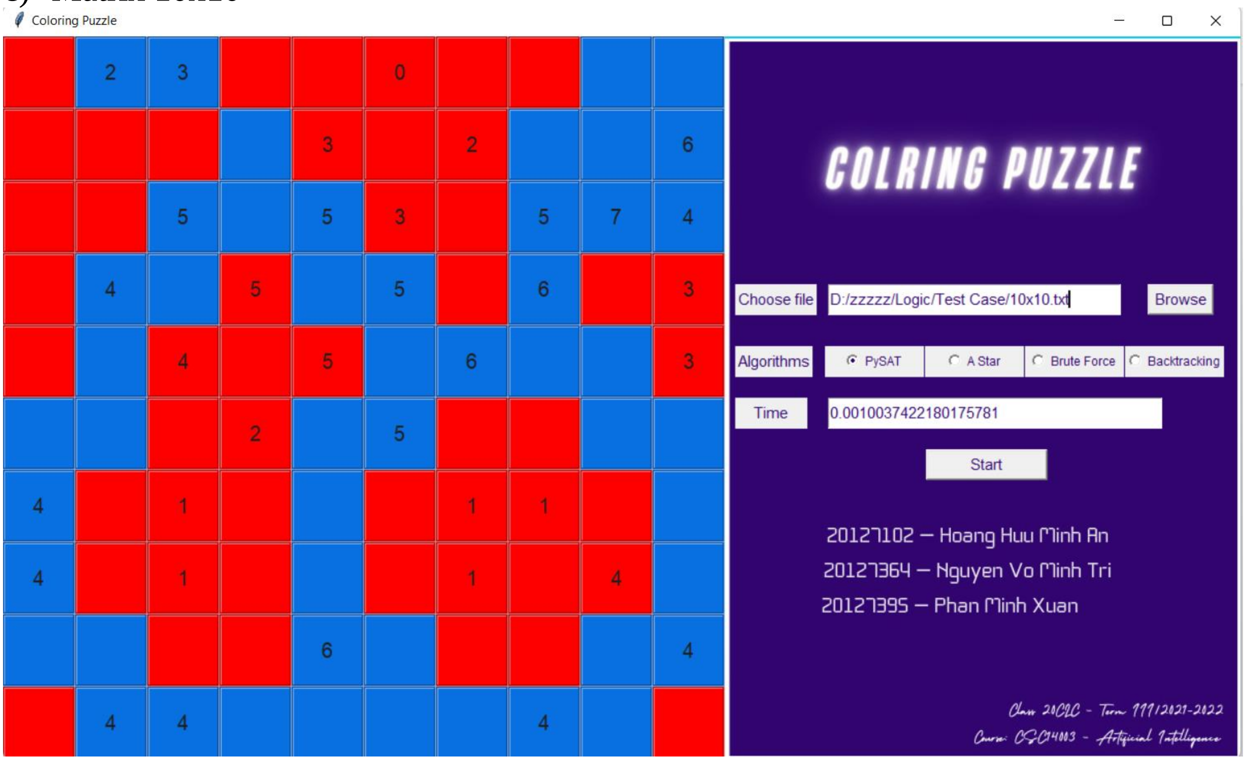
a) Matrix 3x3



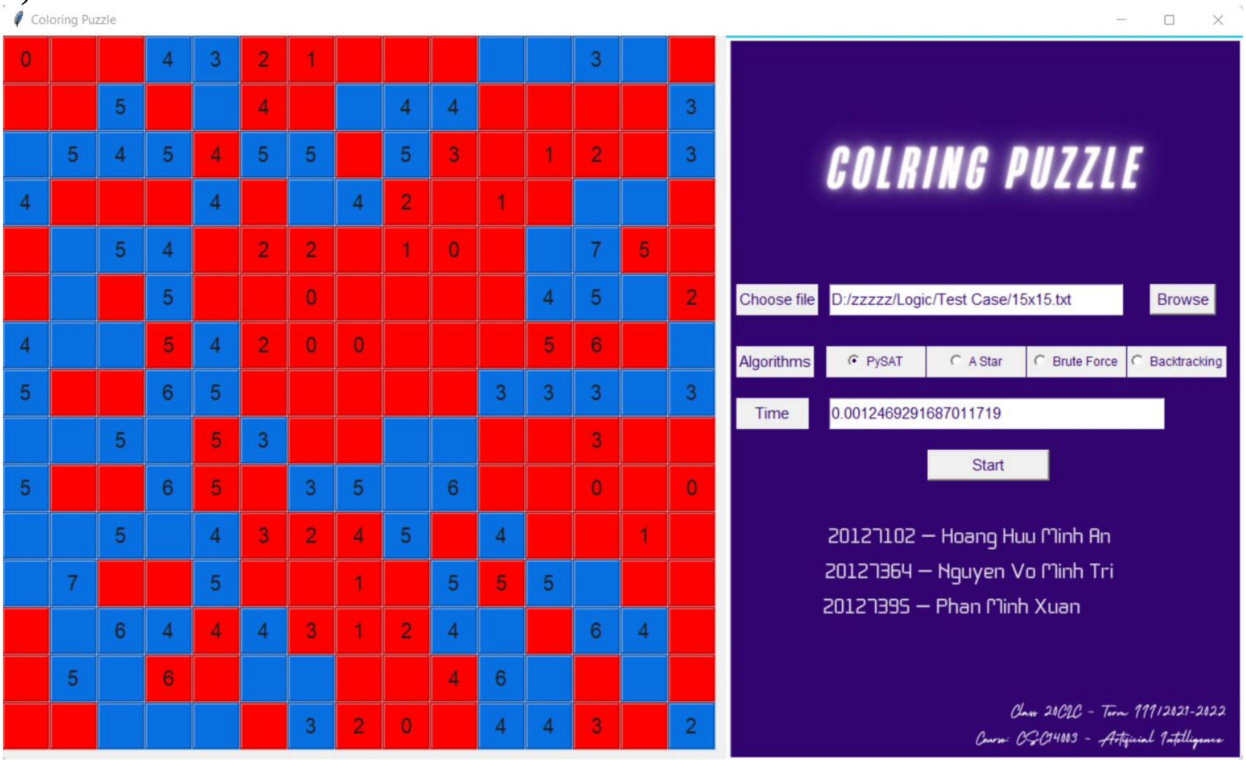
b) Matrix 5x5



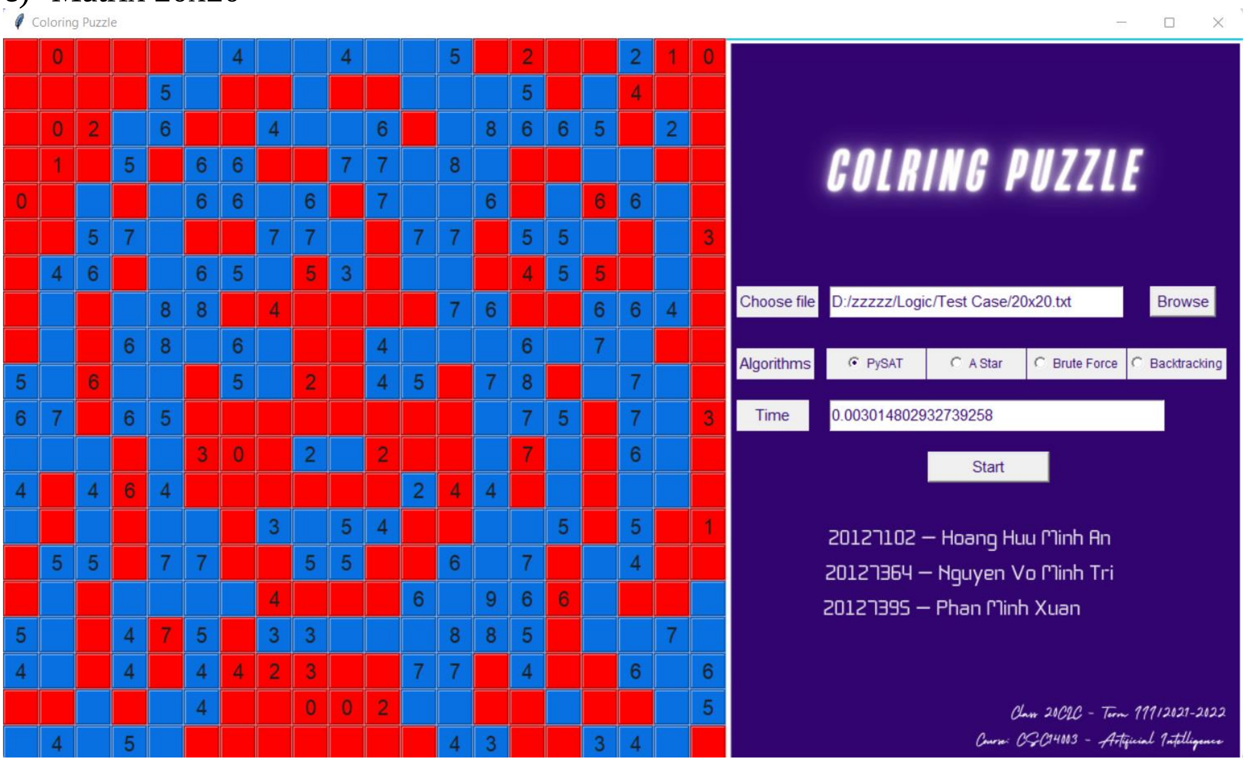
c) Matrix 10x10



d) Matrix 15x15



e) Matrix 20x20

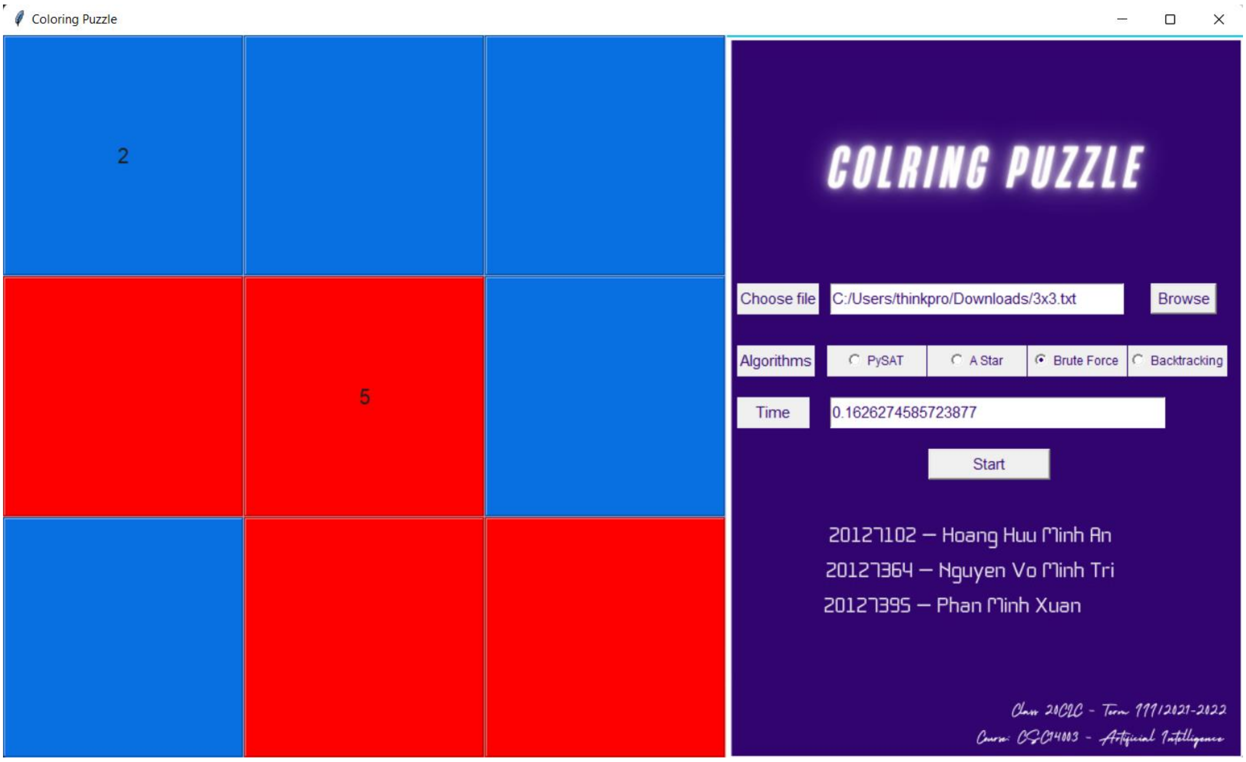


No.	Input	Output
1	3 3 2 -1 -1 -1 5 -1 -1 -1 -1	0 1 1 0 1 1 1 0 0
2	5 5 0 -1 4 4 -1 -1 4 -1 6 -1 3 -1 7 6 -1 -1 6 -1 6 5 -1 -1 -1 -1 3	0 0 1 1 0 0 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1
3	10 10 -1 2 3 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 3 -1 2 -1 -1 6 -1 -1 5 -1 5 3 -1 5 7 4 -1 4 -1 5 -1 5 -1 6 -1 3 -1 -1 4 -1 5 -1 6 -1 -1 3 -1 -1 -1 2 -1 5 -1 -1 -1 -1 4 -1 1 -1 -1 -1 1 1 -1 -1 4 -1 1 -1 -1 -1 1 -1 4 -1 -1 -1 -1 -1 6 -1 -1 -1 -1 4 -1 4 4 -1 -1 -1 -1 4 -1 -1	0 1 1 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1 0 1 1 0 1 1 0 1 0 0 0 1 0 0 0 1 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0 0 0 0 1 1 1 0 0 1 1 0 0 1 1 0 1 1 1 1 1 1 1 1 0
4	15 15 0 -1 -1 4 3 2 1 -1 -1 -1 -1 -1 3 -1 -1 -1 -1 5 -1 -1 4 -1 -1 4 4 -1 -1 -1 -1 3 -1 5 4 5 4 5 5 -1 5 3 -1 1 2 -1 3 4 -1 -1 -1 4 -1 -1 4 2 -1 1 -1 -1 -1 -1 -1 -1 5 4 -1 2 2 -1 1 0 -1 -1 7 5 -1 -1 -1 -1 5 -1 -1 0 -1 -1 -1 -1 4 5 -1 2 4 -1 -1 5 4 2 0 0 -1 -1 -1 5 6 -1 -1 5 -1 -1 6 5 -1 -1 -1 -1 -1 3 3 3 -1 3 -1 -1 5 -1 5 3 -1 -1 -1 -1 -1 -1 3 -1 -1 5 -1 -1 6 5 -1 3 5 -1 6 -1 -1 0 -1 0 -1 -1 5 -1 4 3 2 4 5 -1 4 -1 -1 1 -1 -1 7 -1 -1 5 -1 -1 1 -1 5 5 5 -1 -1 -1 -1 -1 6 4 4 4 3 1 2 4 -1 -1 6 4 -1 -1 5 -1 6 -1 -1 -1 -1 -1 4 6 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 3 2 0 -1 4 4 3 -1 2	0 0 0 1 1 0 0 0 0 0 1 1 1 1 0 0 0 1 0 1 0 0 1 1 1 0 0 0 0 1 1 1 1 1 0 1 1 0 1 0 0 0 0 0 1 1 0 0 0 1 0 1 1 0 0 0 0 1 1 0 0 1 1 1 0 0 0 0 0 0 0 1 1 0 0 0 1 0 1 0 0 0 0 0 0 0 1 1 1 0 1 1 1 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 0 0 0 0 0 1 0 0 1 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 0 0 0 1 0 1 0 0 0 0 1 1 0 0 1 0 0 0 0 1 0 1 1 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 1 0 0 1 1 0 0 1 1 0 0 0 1 1 0 1 0 0 0 1 1 1 0 1 0 0 0 1 1 0 0 1
5	20 20 -1 0 -1 -1 -1 -1 4 -1 -1 4 -1 -1 5 -1 2 -1 -1 2 1 0 -1 -1 -1 -1 5 -1 -1 -1 -1 -1 -1 -1 -1 -1 5 -1 -1 4 -1 -1 -1 0 2 -1 6 -1 -1 4 -1 -1 6 -1 -1 8 6 6 5 -1 2 -1 -1 1 -1 5 -1 6 6 -1 -1 7 7 -1 8 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 6 6 -1 6 -1 7 -1 -1 6 -1 -1 6 6 -1 -1	0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 0 0 0 0 1 1 0 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 1 0 1 1 0 0 1 1 1 1 1 0 0 1 1 0 0 0 0 1 0 1 1 1 1 1 0 1 1 1 1 0 1 0 1 1 0 0 0 1 1 1 0 0 1 1 1 0 1 1 0 1 1 1 0 1 0

	-1 -1 5 7 -1 -1 -1 7 7 -1 -1 7 7 -1 5 5 -1 -1 -1 3 -1 4 6 -1 -1 6 5 -1 5 3 -1 -1 -1 -1 4 5 5 -1 -1 -1 -1 -1 -1 -1 8 8 -1 4 -1 -1 -1 -1 7 6 -1 -1 6 6 4 -1 -1 -1 -1 6 8 -1 6 -1 -1 -1 4 -1 -1 -1 6 -1 7 -1 -1 -1 5 -1 6 -1 -1 -1 5 -1 2 -1 4 5 -1 7 8 -1 -1 7 -1 -1 6 7 -1 6 5 -1 -1 -1 -1 -1 -1 -1 -1 -1 7 5 -1 7 -1 3 -1 -1 -1 -1 -1 3 0 -1 2 -1 2 -1 -1 -1 7 -1 -1 6 -1 -1 4 -1 4 6 4 -1 -1 -1 -1 -1 -1 2 4 4 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 3 -1 5 4 -1 -1 -1 -1 5 -1 5 -1 1 -1 5 5 -1 7 7 -1 -1 5 5 -1 -1 6 -1 7 -1 -1 4 -1 -1 -1 -1 -1 -1 -1 -1 -1 4 -1 -1 -1 6 -1 9 6 6 -1 -1 -1 -1 5 -1 -1 4 7 5 -1 3 3 -1 -1 -1 8 8 5 -1 -1 -1 7 -1 4 -1 -1 4 -1 4 4 2 3 -1 -1 7 7 -1 4 -1 -1 6 -1 6 -1 -1 -1 -1 -1 4 -1 -1 0 0 2 -1 -1 -1 -1 -1 -1 -1 -1 5 -1 4 -1 5 -1 -1 -1 -1 -1 -1 -1 -1 4 3 -1 -1 3 4 -1 -1	0 1 1 0 1 1 1 1 0 1 0 1 1 0 0 1 0 0 1 0 0 1 0 1 1 1 0 0 0 0 0 0 1 1 0 0 1 1 1 0 0 1 0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 0 0 0 0 0 0 0 1 1 1 0 1 1 0 1 1 1 0 1 0 0 0 1 1 0 0 0 1 0 1 0 1 1 0 1 0 1 0 1 0 0 0 0 0 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 1 0 1 1 1 1 0 0 1 1 1 0 1 0 0 0 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 0 0 0 1 0 1 1 1 1 0 0 0 0 1 1 1 1 0 1 0 0 1 1 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 0 1 0 0 0 0 0 1 1 0 1 0 0 1 1 1 0 0 1 0 1 1 0 0 0 0 0 1 1 0 0 1 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 1 1 0 0 1 1 1 0

2. Using Brute-Force

a) Matrix 3x3

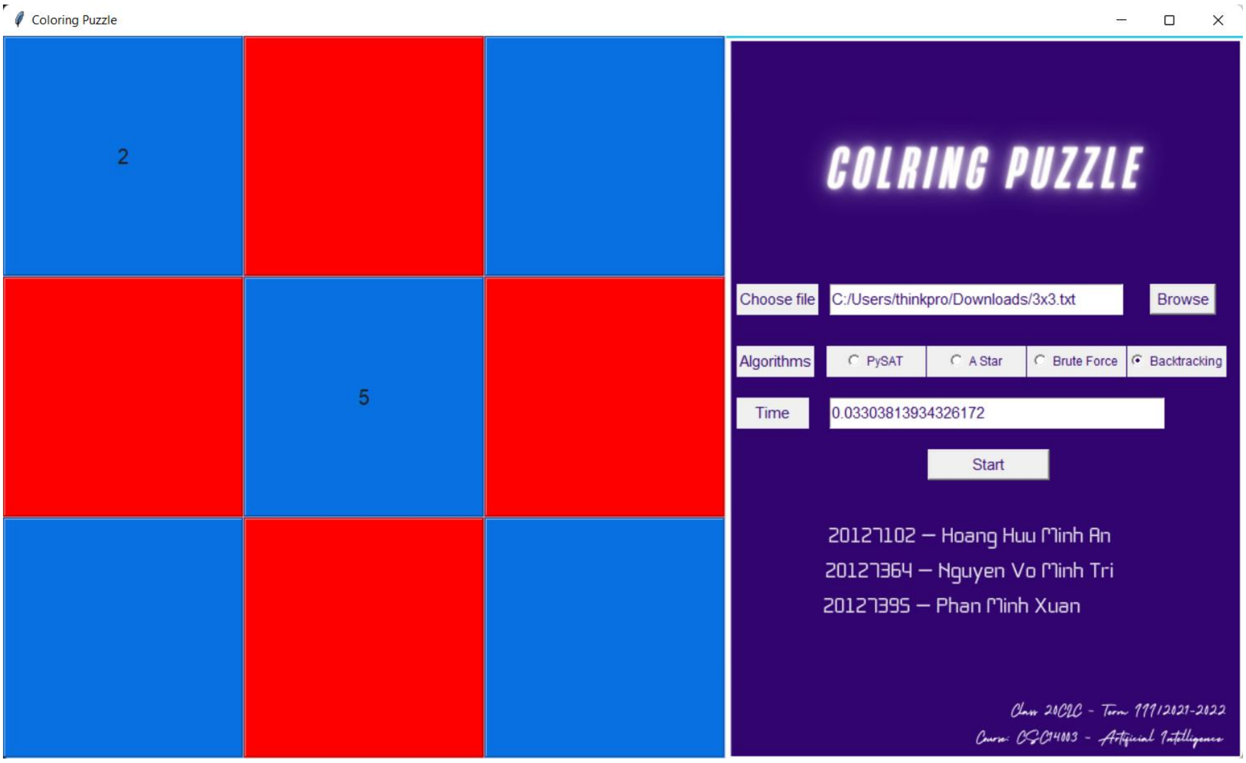


No.	Input	Output
1	3 3 2 -1 -1 -1 5 -1 -1 -1 -1	0 1 1 0 1 1 1 0 0
2	5 5 0 -1 4 4 -1 -1 4 -1 6 -1 3 -1 7 6 -1 -1 6 -1 6 5 -1 -1 -1 -1 3	Running Time: >3600s (1 hour)
3	10 10 -1 2 3 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 3 -1 2 -1 -1 6 -1 -1 5 -1 5 3 -1 5 7 4 -1 4 -1 5 -1 5 -1 6 -1 3 -1 -1 4 -1 5 -1 6 -1 -1 3 -1 -1 -1 2 -1 5 -1 -1 -1 -1 4 -1 1 -1 -1 -1 1 1 -1 -1 4 -1 1 -1 -1 -1 1 -1 4 -1 -1 -1 -1 -1 6 -1 -1 -1 -1 4 -1 4 4 -1 -1 -1 -1 4 -1 -1	Runnung Time: >3600s (1 hour)
4	15 15 0 -1 -1 4 3 2 1 -1 -1 -1 -1 -1 3 -1 -1 -1 -1 5 -1 -1 4 -1 -1 4 4 -1 -1 -1 -1 3 -1 5 4 5 4 5 5 -1 5 3 -1 1 2 -1 3 4 -1 -1 -1 4 -1 -1 4 2 -1 1 -1 -1 -1 -1 -1 -1 5 4 -1 2 2 -1 1 0 -1 -1 7 5 -1 -1 -1 -1 5 -1 -1 0 -1 -1 -1 -1 4 5 -1 2 4 -1 -1 5 4 2 0 0 -1 -1 -1 5 6 -1 -1 5 -1 -1 6 5 -1 -1 -1 -1 -1 3 3 3 -1 3 -1 -1 5 -1 5 3 -1 -1 -1 -1 -1 -1 3 -1 -1 5 -1 -1 6 5 -1 3 5 -1 6 -1 -1 0 -1 0 -1 -1 5 -1 4 3 2 4 5 -1 4 -1 -1 1 -1 -1 7 -1 -1 5 -1 -1 1 -1 5 5 5 -1 -1 -1 -1 -1 6 4 4 4 3 1 2 4 -1 -1 6 4 -1 -1 5 -1 6 -1 -1 -1 -1 -1 4 6 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 3 2 0 -1 4 4 3 -1 2	Runnung Time: >3600s (1 hour)
5	20 20 -1 0 -1 -1 -1 -1 4 -1 -1 4 -1 -1 5 -1 2 -1 -1 2 1 0 -1 -1 -1 -1 5 -1 -1 -1 -1 -1 -1 -1 -1 -1 5 -1 -1 4 -1 -1 -1 0 2 -1 6 -1 -1 4 -1 -1 6 -1 -1 8 6 6 5 -1 2 -1 -1 1 -1 5 -1 6 6 -1 -1 7 7 -1 8 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 6 6 -1 6 -1 7 -1 -1 6 -1 -1 6 6 -1 -1 -1 -1 5 7 -1 -1 -1 7 7 -1 -1 7 7 -1 5 5 -1 -1 -1 3	

	-1 4 6 -1 -1 6 5 -1 5 3 -1 -1 -1 -1 4 5 5 -1 -1 -1 -1 -1 -1 -1 8 8 -1 4 -1 -1 -1 -1 7 6 -1 -1 6 6 4 -1 -1 -1 -1 6 8 -1 6 -1 -1 -1 4 -1 -1 -1 6 -1 7 -1 -1 -1 5 -1 6 -1 -1 -1 5 -1 2 -1 4 5 -1 7 8 -1 -1 7 -1 -1 6 7 -1 6 5 -1 -1 -1 -1 -1 -1 -1 -1 -1 7 5 -1 7 -1 3 -1 -1 -1 -1 -1 3 0 -1 2 -1 2 -1 -1 -1 7 -1 -1 6 -1 -1 4 -1 4 6 4 -1 -1 -1 -1 -1 -1 2 4 4 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 3 -1 5 4 -1 -1 -1 -1 5 -1 5 -1 1 -1 5 5 -1 7 7 -1 -1 5 5 -1 -1 6 -1 7 -1 -1 4 -1 -1 -1 -1 -1 -1 -1 -1 -1 4 -1 -1 -1 6 -1 9 6 6 -1 -1 -1 -1 5 -1 -1 4 7 5 -1 3 3 -1 -1 -1 8 8 5 -1 -1 -1 7 -1 4 -1 -1 4 -1 4 4 2 3 -1 -1 7 7 -1 4 -1 -1 6 -1 6 -1 -1 -1 -1 -1 4 -1 -1 0 0 2 -1 -1 -1 -1 -1 -1 -1 -1 5 -1 4 -1 5 -1 -1 -1 -1 -1 -1 -1 -1 4 3 -1 -1 3 4 -1 -1	Runnung Time: >3600s (1 hour)

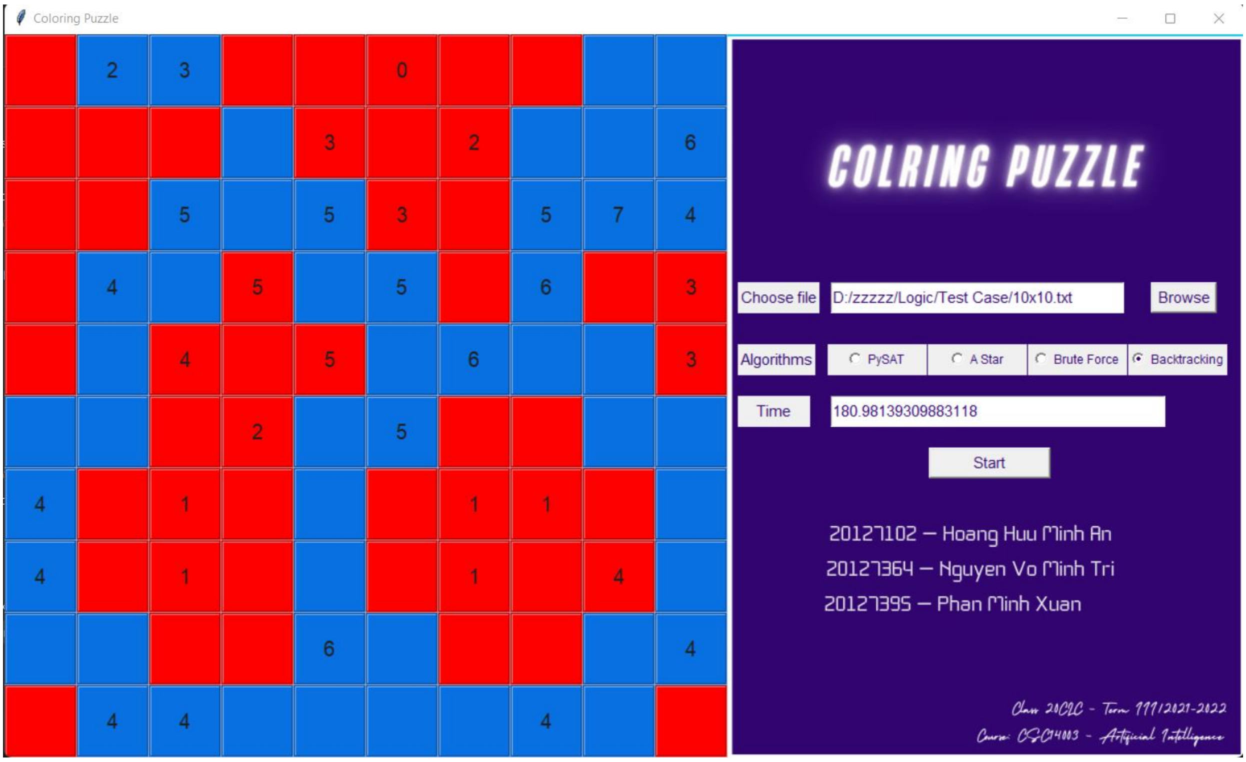
3.Using Backtracking

a. Matrix 3x3

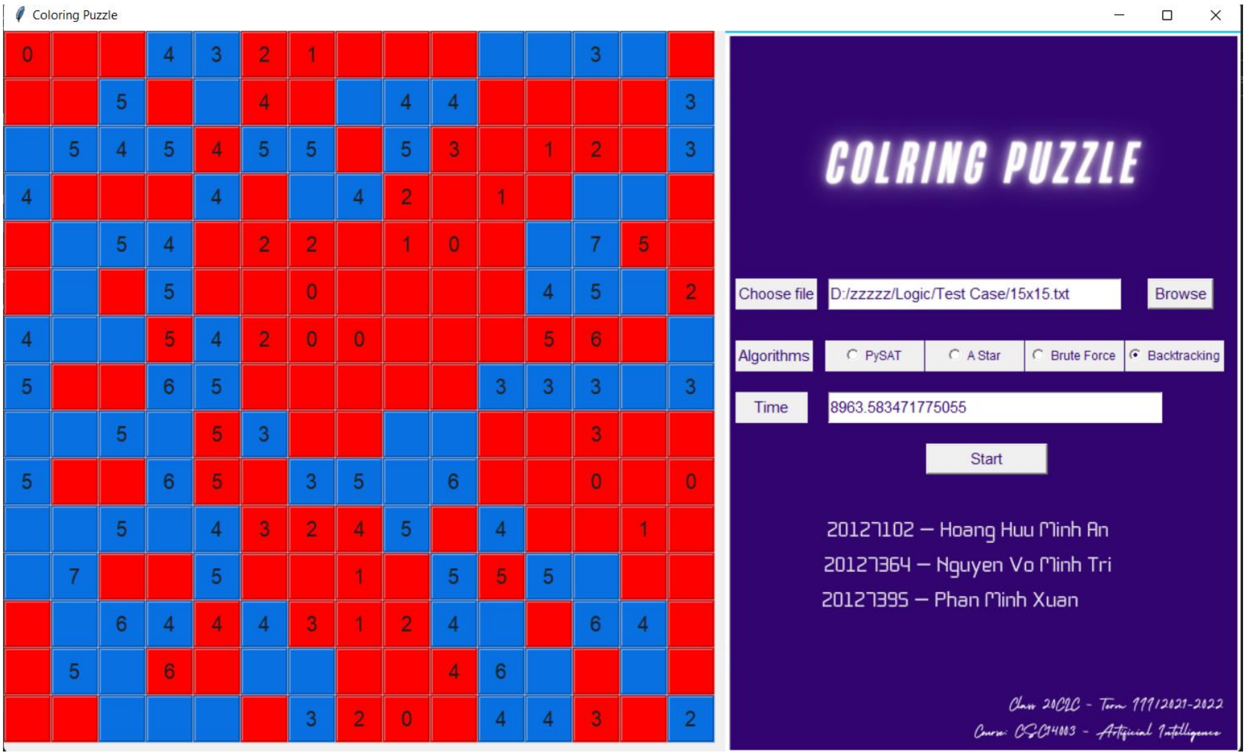


b. Matrix 5x5

c. Matrix 10x10



d. Matrix 15x15

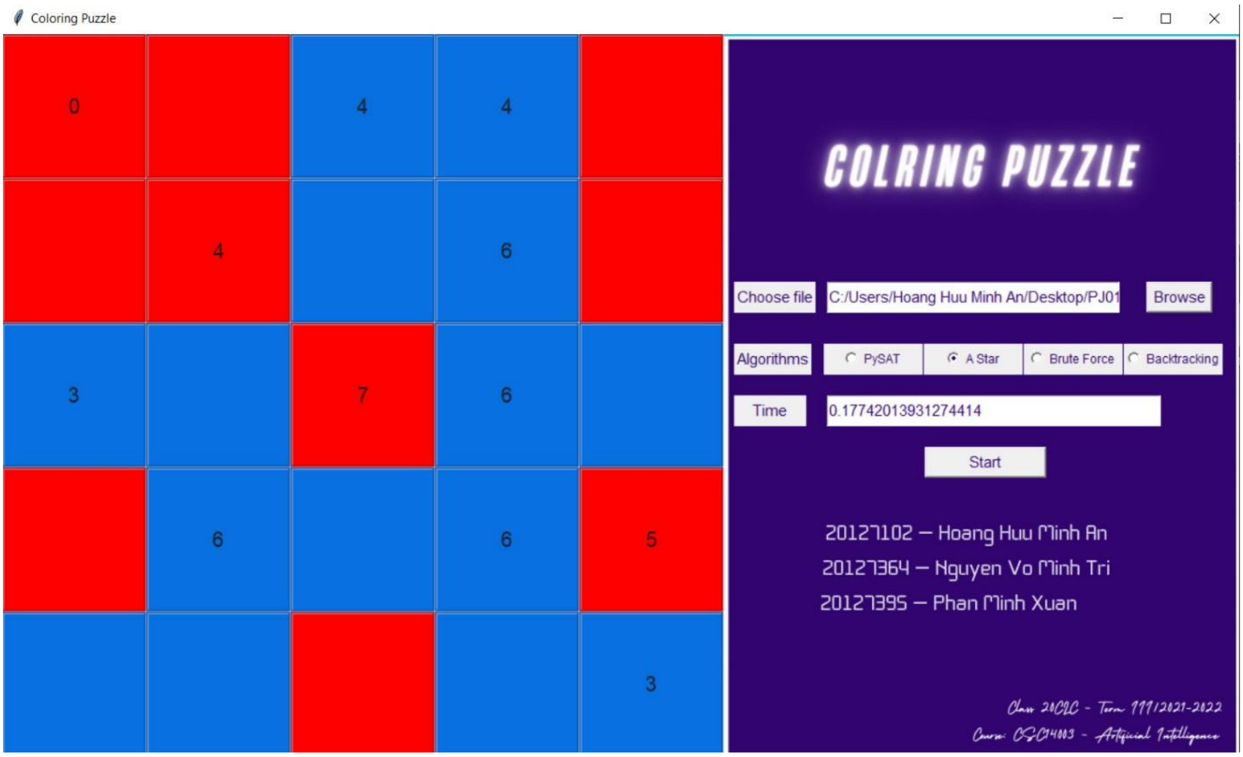


No.	Input	Output
1	3 3 2 -1 -1 -1 5 -1 -1 -1 -1	0 1 1 0 1 1 1 0 0
2	5 5 0 -1 4 4 -1 -1 4 -1 6 -1 3 -1 7 6 -1 -1 6 -1 6 5 -1 -1 -1 -1 3	0 0 1 1 0 0 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1
3	10 10 -1 2 3 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 3 -1 2 -1 -1 6 -1 -1 5 -1 5 3 -1 5 7 4 -1 4 -1 5 -1 5 -1 6 -1 3 -1 -1 4 -1 5 -1 6 -1 -1 3 -1 -1 -1 2 -1 5 -1 -1 -1 -1 4 -1 1 -1 -1 -1 1 1 -1 -1 4 -1 1 -1 -1 -1 1 -1 4 -1 -1 -1 -1 -1 6 -1 -1 -1 -1 4	0 1 1 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1 0 1 1 0 1 1 0 1 0 0 0 1 0 0 0 1 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0 0 0 0 1 1 1 0 0 1 1 0 0 1 1 0 1 1 1 1 1 1 1 1 0

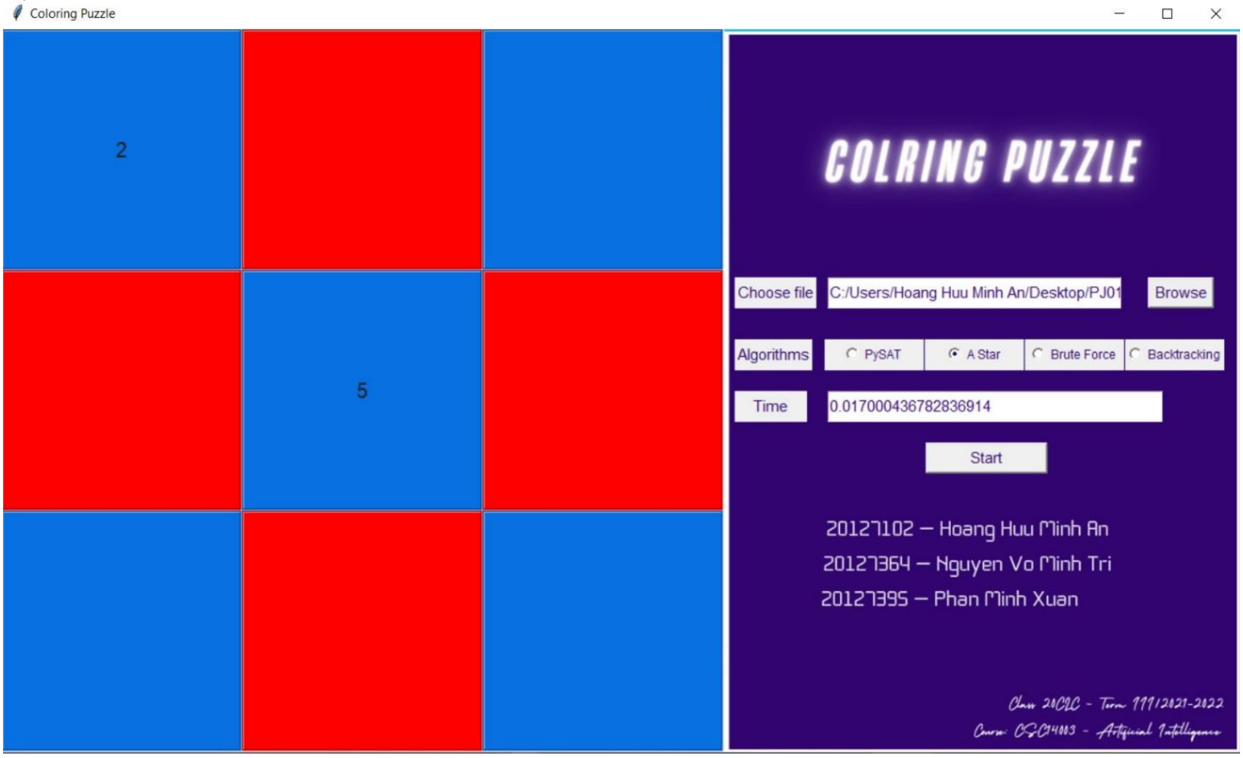
	-1 4 4 -1 -1 -1 -1 4 -1 -1	
4	15 15 0 -1 -1 4 3 2 1 -1 -1 -1 -1 -1 3 -1 -1 -1 -1 5 -1 -1 4 -1 -1 4 4 -1 -1 -1 -1 3 -1 5 4 5 4 5 5 -1 5 3 -1 1 2 -1 3 4 -1 -1 -1 4 -1 -1 4 2 -1 1 -1 -1 -1 -1 -1 -1 5 4 -1 2 2 -1 1 0 -1 -1 7 5 -1 -1 -1 -1 5 -1 -1 0 -1 -1 -1 -1 4 5 -1 2 4 -1 -1 5 4 2 0 0 -1 -1 -1 5 6 -1 -1 5 -1 -1 6 5 -1 -1 -1 -1 -1 3 3 3 -1 3 -1 -1 5 -1 5 3 -1 -1 -1 -1 -1 -1 3 -1 -1 5 -1 -1 6 5 -1 3 5 -1 6 -1 -1 0 -1 0 -1 -1 5 -1 4 3 2 4 5 -1 4 -1 -1 1 -1 -1 7 -1 -1 5 -1 -1 1 -1 5 5 5 -1 -1 -1 -1 -1 6 4 4 4 3 1 2 4 -1 -1 6 4 -1 -1 5 -1 6 -1 -1 -1 -1 -1 4 6 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 3 2 0 -1 4 4 3 -1 2	0 0 0 1 1 0 0 0 0 0 1 1 1 1 0 0 0 1 0 1 0 0 1 1 1 0 0 0 0 1 1 1 1 1 0 1 1 0 1 0 0 0 0 0 1 1 0 0 0 1 0 1 1 0 0 0 0 1 1 0 0 1 1 1 0 0 0 0 0 0 0 1 1 0 0 0 1 0 1 0 0 0 0 0 0 0 1 1 1 0 1 1 1 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 0 0 0 0 0 1 0 0 1 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1 1 0 0 0 1 0 1 0 0 0 0 1 1 0 0 1 0 0 0 0 1 0 1 1 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 1 0 0 1 1 0 0 1 1 0 0 0 1 1 0 1 0 0 0 1 1 1 0 1 0 0 0 1 1 0 0 1
5	20 20 -1 0 -1 -1 -1 -1 4 -1 -1 4 -1 -1 5 -1 2 -1 -1 2 1 0 -1 -1 -1 -1 5 -1 -1 -1 -1 -1 -1 -1 -1 -1 5 -1 -1 4 -1 -1 -1 0 2 -1 6 -1 -1 4 -1 -1 6 -1 -1 8 6 6 5 -1 2 -1 -1 1 -1 5 -1 6 6 -1 -1 7 7 -1 8 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 6 6 -1 6 -1 7 -1 -1 6 -1 -1 6 6 -1 -1 -1 -1 5 7 -1 -1 -1 7 7 -1 -1 7 7 -1 5 5 -1 -1 -1 3 -1 4 6 -1 -1 6 5 -1 5 3 -1 -1 -1 -1 4 5 5 -1 -1 -1 -1 -1 -1 -1 8 8 -1 4 -1 -1 -1 -1 7 6 -1 -1 6 6 4 -1 -1 -1 -1 6 8 -1 6 -1 -1 -1 4 -1 -1 -1 6 -1 7 -1 -1 -1 5 -1 6 -1 -1 -1 5 -1 2 -1 4 5 -1 7 8 -1 -1 7 -1 -1 6 7 -1 6 5 -1 -1 -1 -1 -1 -1 -1 -1 -1 7 5 -1 7 -1 3 -1 -1 -1 -1 -1 3 0 -1 2 -1 2 -1 -1 -1 7 -1 -1 6 -1 -1 4 -1 4 6 4 -1 -1 -1 -1 -1 -1 2 4 4 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 3 -1 5 4 -1 -1 -1 -1 5 -1 5 -1 1 -1 5 5 -1 7 7 -1 -1 5 5 -1 -1 6 -1 7 -1 -1 4 -1 -1 -1 -1 -1 -1 -1 -1 -1 4 -1 -1 -1 6 -1 9 6 6 -1 -1 -1 -1 5 -1 -1 4 7 5 -1 3 3 -1 -1 -1 8 8 5 -1 -1 -1 7 -1 4 -1 -1 4 -1 4 4 2 3 -1 -1 7 7 -1 4 -1 -1 6 -1 6 -1 -1 -1 -1 -1 4 -1 -1 0 0 2 -1 -1 -1 -1 -1 -1 -1 -1 5 -1 4 -1 5 -1 -1 -1 -1 -1 -1 -1 -1 4 3 -1 -1 3 4 -1 -1	Runnung Time: >3600s (1 hour)

4. Using A*

a) Matrix 3x3


Coloring Puzzle
0
4
4
4
6
3
7
6
6
6
5
3
COLRING PUZZLE
Choose file
C:/Users/Hoang Huu Minh An/Desktop/PJ01
Browse
Algorithms
PySAT
A Star
Brute Force
Backtracking
Time
0.17742013931274414
Start
20127102 – Hoang Huu Minh An
20127364 – Nguyen Vo Minh Tri
20127395 – Phan Minh Xuan

b) Matrix 5x5


Coloring Puzzle
2
5
COLRING PUZZLE
Choose file
C:/Users/Hoang Huu Minh An/Desktop/PJ01
Browse
Algorithms
PySAT
A Star
Brute Force
Backtracking
Time
0.017000436782836914
Start
20127102 – Hoang Huu Minh An
20127364 – Nguyen Vo Minh Tri
20127395 – Phan Minh Xuan

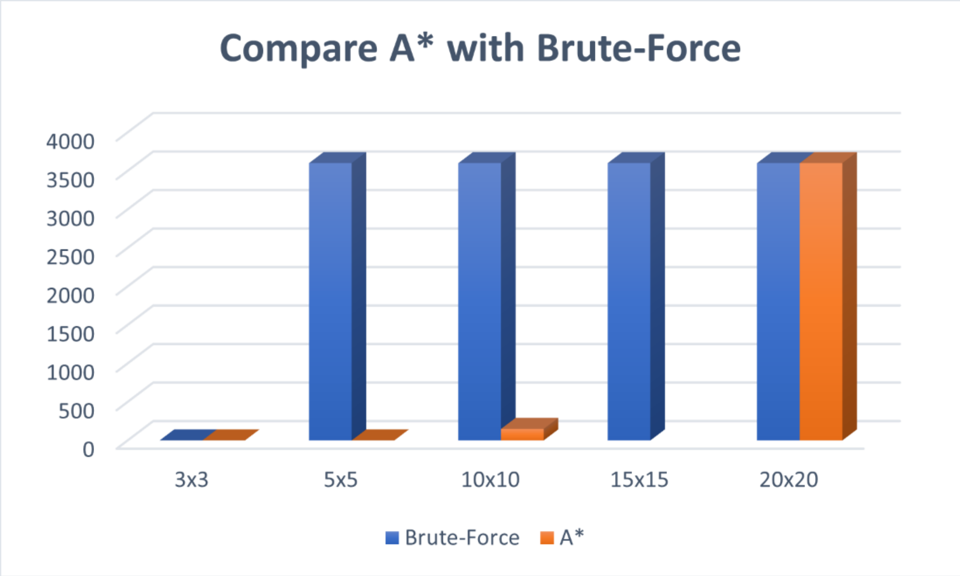
c) Matrix 10x10

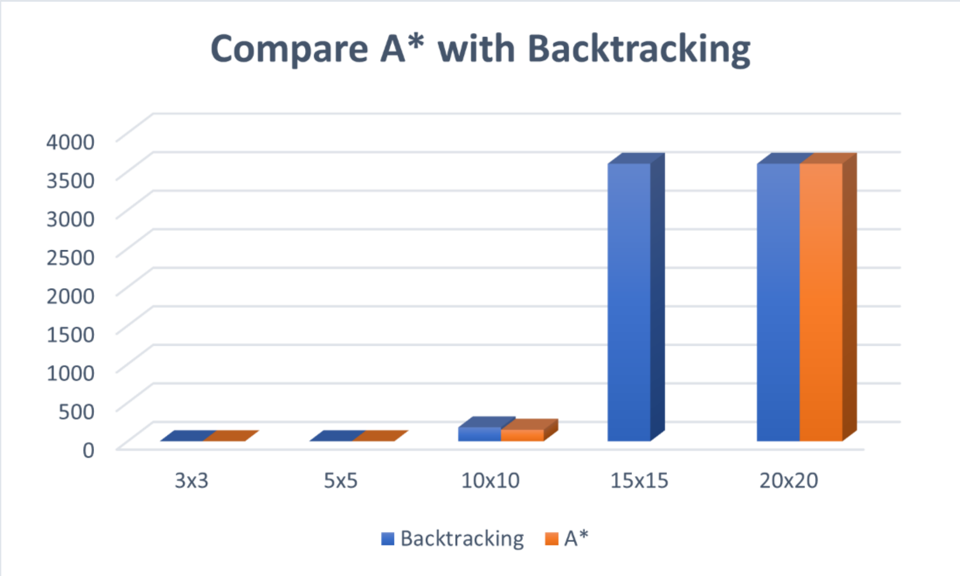
No.	Input	Output
1	3 3 2 -1 -1 -1 5 -1 -1 -1 -1	0 1 1 0 1 1 1 0 0
2	5 5 0 -1 4 4 -1 -1 4 -1 6 -1 3 -1 7 6 -1 -1 6 -1 6 5 -1 -1 -1 -1 3	0 0 1 1 0 0 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1
3	10 10 -1 2 3 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 3 -1 2 -1 -1 6 -1 -1 5 -1 5 3 -1 5 7 4 -1 4 -1 5 -1 5 -1 6 -1 3 -1 -1 4 -1 5 -1 6 -1 -1 3 -1 -1 -1 2 -1 5 -1 -1 -1 -1 4 -1 1 -1 -1 -1 1 1 -1 -1 4 -1 1 -1 -1 -1 1 -1 4 -1	0 1 1 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1 0 1 1 0 1 1 0 1 0 0 0 1 0 0 0 1 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 0 0 1 0 0 0 0 1 1 0 0 0 1 0 0 0 0 1 1 1 0 0 1 1 0 0 1 1

	-1 -1 -1 -1 6 -1 -1 -1 -1 4 -1 4 4 -1 -1 -1 -1 4 -1 -1	0 1 1 1 1 1 1 1 1 0
4	15 15 0 -1 -1 4 3 2 1 -1 -1 -1 -1 -1 3 -1 -1 -1 -1 5 -1 -1 4 -1 -1 4 4 -1 -1 -1 -1 3 -1 5 4 5 4 5 5 -1 5 3 -1 1 2 -1 3 4 -1 -1 -1 4 -1 -1 4 2 -1 1 -1 -1 -1 -1 -1 -1 5 4 -1 2 2 -1 1 0 -1 -1 7 5 -1 -1 -1 -1 5 -1 -1 0 -1 -1 -1 -1 4 5 -1 2 4 -1 -1 5 4 2 0 0 -1 -1 -1 5 6 -1 -1 5 -1 -1 6 5 -1 -1 -1 -1 -1 3 3 3 -1 3 -1 -1 5 -1 5 3 -1 -1 -1 -1 -1 -1 3 -1 -1 5 -1 -1 6 5 -1 3 5 -1 6 -1 -1 0 -1 0 -1 -1 5 -1 4 3 2 4 5 -1 4 -1 -1 1 -1 -1 7 -1 -1 5 -1 -1 1 -1 5 5 5 -1 -1 -1 -1 -1 6 4 4 4 3 1 2 4 -1 -1 6 4 -1 -1 5 -1 6 -1 -1 -1 -1 -1 4 6 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 3 2 0 -1 4 4 3 -1 2	Running Time: >3600s (1 hour)
5	20 20 -1 0 -1 -1 -1 -1 4 -1 -1 4 -1 -1 5 -1 2 -1 -1 2 1 0 -1 -1 -1 -1 5 -1 -1 -1 -1 -1 -1 -1 -1 -1 5 -1 -1 4 -1 -1 -1 0 2 -1 6 -1 -1 4 -1 -1 6 -1 -1 8 6 6 5 -1 2 -1 -1 1 -1 5 -1 6 6 -1 -1 7 7 -1 8 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 6 6 -1 6 -1 7 -1 -1 6 -1 -1 6 6 -1 -1 -1 -1 5 7 -1 -1 -1 7 7 -1 -1 7 7 -1 5 5 -1 -1 -1 3 -1 4 6 -1 -1 6 5 -1 5 3 -1 -1 -1 -1 4 5 5 -1 -1 -1 -1 -1 -1 -1 8 8 -1 4 -1 -1 -1 -1 7 6 -1 -1 6 6 4 -1 -1 -1 -1 6 8 -1 6 -1 -1 -1 4 -1 -1 -1 6 -1 7 -1 -1 -1 5 -1 6 -1 -1 -1 5 -1 2 -1 4 5 -1 7 8 -1 -1 7 -1 -1 6 7 -1 6 5 -1 -1 -1 -1 -1 -1 -1 -1 -1 7 5 -1 7 -1 3 -1 -1 -1 -1 -1 3 0 -1 2 -1 2 -1 -1 -1 7 -1 -1 6 -1 -1 4 -1 4 6 4 -1 -1 -1 -1 -1 -1 2 4 4 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 3 -1 5 4 -1 -1 -1 -1 5 -1 5 -1 1 -1 5 5 -1 7 7 -1 -1 5 5 -1 -1 6 -1 7 -1 -1 4 -1 -1 -1 -1 -1 -1 -1 -1 -1 4 -1 -1 -1 6 -1 9 6 6 -1 -1 -1 -1 5 -1 -1 4 7 5 -1 3 3 -1 -1 -1 8 8 5 -1 -1 -1 7 -1 4 -1 -1 4 -1 4 4 2 3 -1 -1 7 7 -1 4 -1 -1 6 -1 6 -1 -1 -1 -1 -1 4 -1 -1 0 0 2 -1 -1 -1 -1 -1 -1 -1 -1 5 -1 4 -1 5 -1 -1 -1 -1 -1 -1 -1 -1 4 3 -1 -1 3 4 -1 -1	Runnung Time: >3600s (1 hour)

X. EXPERIMENT

Algorithms	Running Time (seconds)				
	3x3	5x5	10x10	15x15	20x20
Pysat	0.000988	0.001041	0.001003	0.0012469	0.003014
Brute-force	0.126274	> 3600s	> 3600s	> 3600s	> 3600s
Backtracking	0.033038	0.112219	180.98139	8963.5834	> 3600s
A*	0.017	0.17742	149.39893	>3600s	>3600s





XI. CONCLUSION

- Algorithm A* is an optimization algorithm that uses heuristic evaluation to find the next move. So, the A* algorithm is heavily dependent on the heuristic functions. If the difference in heuristic estimates is large enough, it will lead to false results. The disadvantage of the A* algorithm is that it consumes a lot of memory.
- Algorithms Brute-Force and Backtracking applied in this problem will always have the right results, but their disadvantages is that the running time is very long, depending on the size of the matrix. We are able to use CNF clause with pysat module of Python to solve satisfied problems.

XII. REFERENCE

- Material of lecturer: https://l.facebook.com/l.php?u=https%3A%2F%2Fdrive.google.com%2Fdrive%2Ffolders%2F1OmW_a959zfyCfWP__agipFzfEZX_eV1S%3Ffbclid%3DIwAR2Zfdd40XKkkqm9w-BgKgYdVy8HuAT1EXkR4BNO9IGSArloavoF9JtE52g&h=AT2Jk4i7ueaCNwHenSrJE111W4IlTnmh3XvSw3iMnynnDwfhCUiAci6zEMf8Y0XCT1DKBLU3nKF1fKl8qc6Z4YHGsaun8WSNtTDmzTgsFgUFzNMvjEvRRT4z049bAgsRvs4y0ALzVrxBBO5sQY6iYg
- A* Search Brilliant Math & Science Wifi: https://brilliant.org/wiki/a-star-search/?fbclid=IwAR0DRY9FXAyIxPArXYJv6itfMEwFfoUxe6UHasLyGrtHf5LNbWveNvEFYpE
- Python – GUI Programming (Tkinter) https://www.tutorialspoint.com/python/python_gui_programming.htm?fbclid=IwAR0WHvK015v37WH6LUcUHKs_rUofJCZrLlGmH-bXE2P2gyxXH6TZTGee-8Y
- Backtracking – Wifipedia https://en.wikipedia.org/wiki/Backtracking?fbclid=IwAR2Zfdd40XKkkqm9w-BgKgYdVy8HuAT1EXkR4BNO9IGSArloavoF9JtE52g
- Brute-force Wikipedia https://en.wikipedia.org/wiki/Brute-force_search?fbclid=IwAR0oRD_PviqRbwoKuQcT-mxby3tCOZDODZ6cSo9BnsagnMt3olUDiMEvHOM

☙END❧