FCDS – Cybersecurity department – lev 2

8-Puzzle solver project

Team members :

- Ahmed Hosni Fahmi ID : 20221455470
- Mohamed Moataz ID : 20221446731
- Adham Ehab Ahmed ID : 2106124

Overview :

Our program is written in python and consists of 3 AI agents that solve the 8-puzzle problem using 3 different algorithms; BFS, DFS, A*.

Our program also consists of Puzzle, main and GUI classes.

- Puzzle class :

  A puzzle object takes a string of integers as an input.

  Ex : 643512780

  The puzzle has a function to check whether the puzzle is solved or not. It also has a function that generates all possible swaps, a function that swaps a given number with the '0' mark if possible, and a function to find the position of a given number in the puzzle's matrix.

- B_D_solvingAgents file :
  Consists of a Node class and SolvingAgent class:

  Node class which holds the puzzle object, its parent, the cost to reach it, and its swapped number.

  SolvingAgent class which solves the 8-puzzle using BFS and DFS algorithms on user demand and keep track of states (node objects) through our journey to the goal state.

- A_Star_SolvingAgent file :

  Consists of a Node class and SolvingAgent class:

  Node class which holds the puzzle object, its parent, the cost to reach it, and its swapped number.

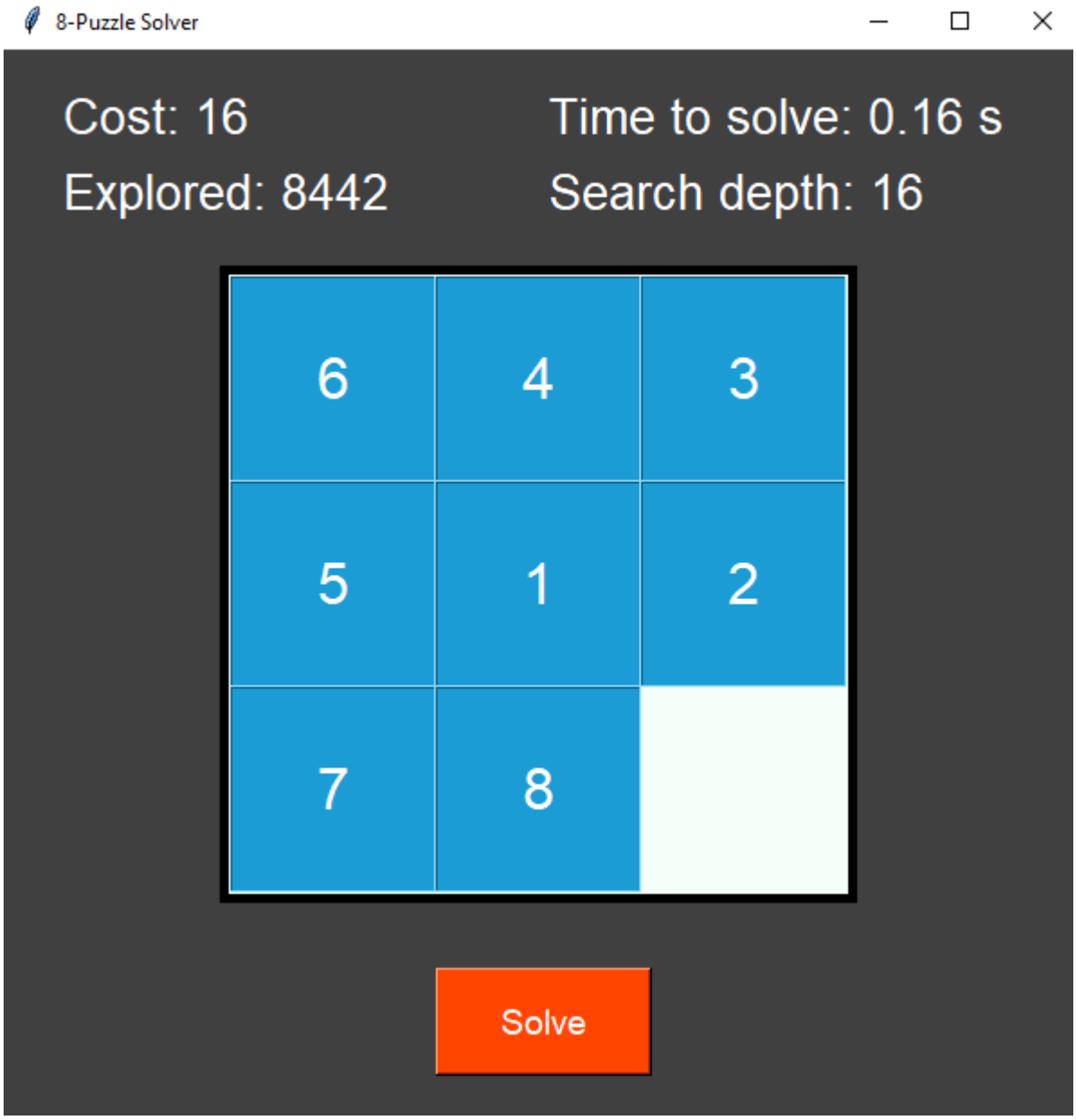
  SolvingAgent class which solves the 8-puzzle using the A* algorithm and chooses the heuristic on user demand and keep track of states (node objects) through our journey to the goal state.

Our A* agent uses one of four different heuristics based on user choice.

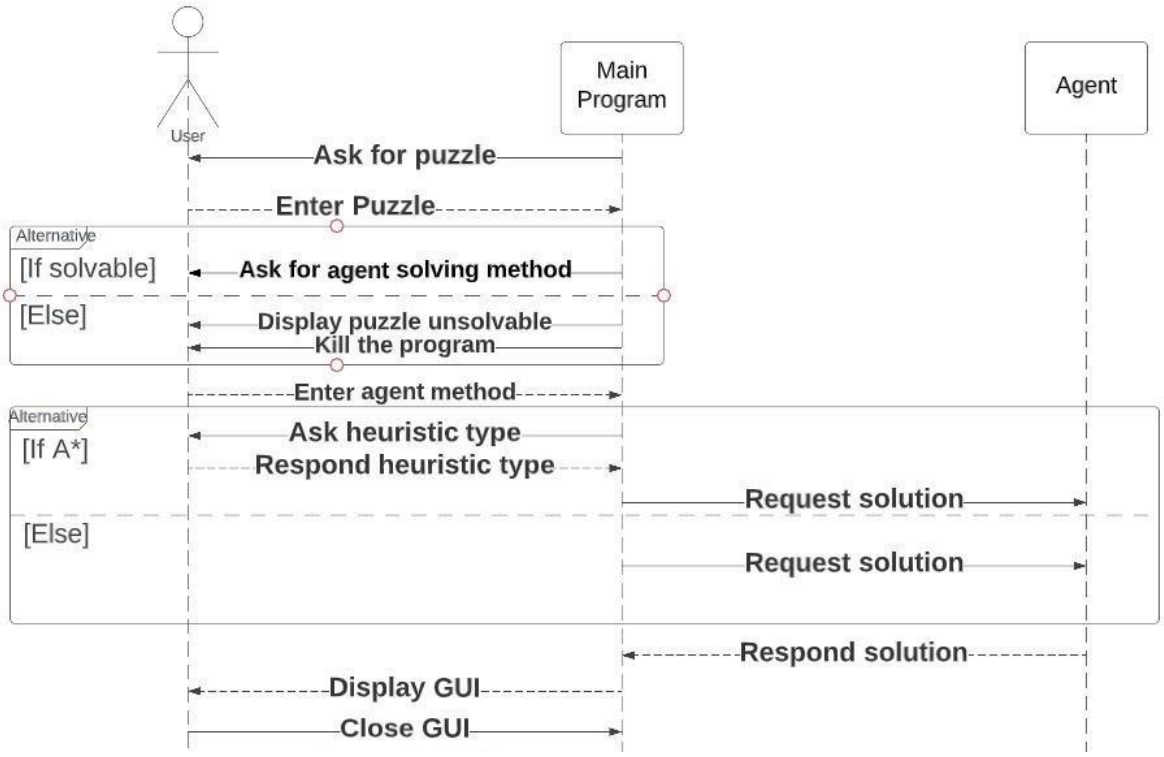
1- The total Euclidean distance of the whole board from this state to the final state.
2- The total mannhaten distance of the whole board from this state to the final state.
3- The Euclidean distance only for the swapped number to its right place (a bad heuristic)
4- The mannhaten distance only for the swapped number to its right place (a bad heuristic)

The third and fourth heuristics are bad heuristics because they don't keep track of the whole board but only a single number, therefore they don't help the agent that much in its journey to reaching the goal, and it visits a lot more nodes and takes a lot more time when compared to it when it uses the previous two heuristics.

- PuzzleGUI class:
  It displays the steps of the path from the initial state to the goal state graphically, number of the explored nodes, the cost of the path and the running time of the chosen algorithm.



- Main file :


User
Main Program
Agent
Ask for puzzle
Enter Puzzle
Alternative
[If solvable]
Ask for agent solving method
[Else]
Display puzzle unsolvable
Kill the program
Enter agent method
Alternative
[If A*]
Ask heuristic type
Respond heuristic type
Request solution
[Else]
Request solution
Respond solution
Display GUI
Close GUI

Data structures used :

- For the puzzle :
    - A matrix using lists to store our puzzle

- For A* agent :
    - Priority queue to store nodes that we are going to explore
    - Dictionary for visited nodes
    - List for path to goal

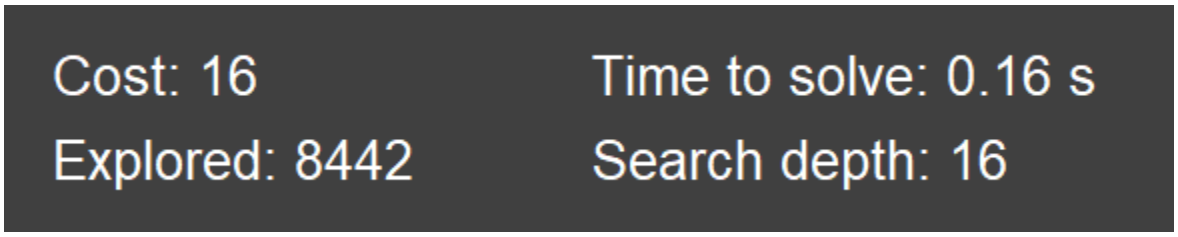
- For BFS & DFS agent :
    - Deque to store nodes that we are going to explore
    - Dictionary for visited nodes
    - List for path to goal

Sample run :

We assumed here that “643512780” is our puzzle

```
Welcome to our humble 8-puzzle intelligent solver.
Our program can solve the puzzle using BFS, DFS and A* algorithm.
It can even make coffee if you want it to. ;)

Please enter the puzzle (ex. 012345678): 643512780
1- BFS
2- DFS
3- A*
Enter the solving method (1, 2, 3): 1
['8', '1', '5', '6', '4', '3', '2', '5', '1', '7', '6', '4', '3', '1', '4', '3']
0.15659499168395996
```



We chose **BFS** to solve that puzzle

Path to goal from the output :
['8', '1', '5', '6', '4', '3', '2', '5', '1', '7', '6', '4', '3', '1', '4', '3']

Solved in run time = 0.15659499168395996 secs

Cost = 16

Explored nodes = 8442

Search depth = 16