Assignment 1:
Solving Color-Maze Puzzle using A* Search

Due Sunday, 17 March, 11:30pm

Color-Maze puzzle is a single-agent grid-game played on a rectangular board that includes a maze. Initially, the agent is located on a single maze cell. The agent can move in four cardinal directions: up, down, right or left. Once a direction is chosen, the agent moves in that direction until it reaches a wall at once, and colors all the cells it travels over. Once a cell is colored, its color does not change. The goal is to color all the cells of the maze by moving the agent over them, while minimizing the total distance traveled by the agent. This game is available at:

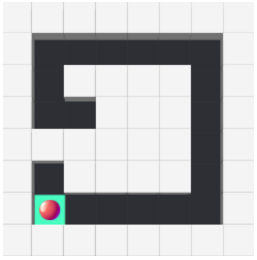
`https://www.mathplayground.com/logic_color_maze`.

Please implement in Python an A* search algorithm to solve the following version of the Color-Maze puzzle.

Input is a rectangular game board with a single agent, represented as a grid where each grid cell is uniquely marked with 0, X or S:

- 0 denotes the cells that are empty and that need to be colored,
- X denotes the cells occupied by the walls, and
- S denotes the cell occupied by the agent.

For instance, in Figure 1 (right) describes the game board depicted in Figure 1 (left).



0	0	0	0	0	0
0	X	X	X	X	0
0	0	X	X	X	0
X	X	X	X	X	0
0	X	X	X	X	0
S	0	0	0	0	0

Figure 1: An example puzzle (left) and its representation in the given format (right).

Output is an alternating sequence of states and moves, that illustrates how the agent colors all the cells such that the total distance traveled by the agent is minimized.

For instance, Figure 2 (resp. Figure 3) shows step by step how the agent can color the cells of a given maze, where the total distance traveled by the agent is 6 (resp. 5). In both solutions, the agent takes the same number of moves but the total distance traveled by the agent is smaller in Figure 3. Therefore, the sequence of states and actions illustrated in Figure 3 should be returned as the output.

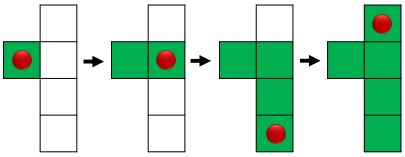


Figure 2: The agent moves right, down, and then up, and travels 6 units of distance in total.

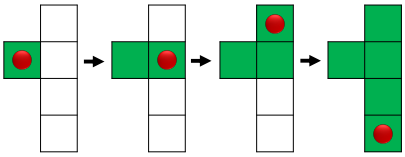


Figure 3: The agent moves right, up, and then down, and travels 5 units of distance in total.

What to do The assignment consists of four parts:

1. (20 points) Model the Color-Maze puzzle as a search problem: Specify the states, successor state function, initial state, goal test, and step cost function.[1]
2. (20 points, provided that part 1 is completed) Extend your search model for Color-Maze to apply A* search:[1]
   (a) Find an inadmissible heuristic function h_1 and illustrate with an example that h_1 is not admissible.
   (b) Find an admissible heuristic function h_2 and prove that h_2 is admissible.
   (c) Discuss whether h_2 is monotone or not.
3. (20 points, provided that part 2 is completed) Implement in Python the A* search algorithm studied in class,[2]to solve the Color-Maze puzzle. Make sure that your implementation displays the solution (i.e., an alternating sequence of states and actions) as well as the total distance travelled.
4. (40 points, provided that part 3 is completed)
   (a) Define 3 difficulty levels (e.g., easy, normal, difficult) for a Color-Maze puzzle instance, and prepare a benchmark set of at least 15 Color-Maze puzzle instances of 3 difficulty levels (i.e., 5 easy, 5 normal, 5 difficult instances), on a game board of size 12×12.[3]
   (b) Evaluate your A* implementations experimentally over these benchmark instances, and summarize the results of your experiments in a table that shows, for each puzzle instance,
      – the number of cells in the maze, the difficulty level,
      – the total distance traveled by the agent, with heuristic h_1 vs. h_2,
      – the total number of expanded nodes, with heuristic h_1 vs. h_2,
      – the CPU time and the memory consumption, with heuristic h_1 vs. h_2.
   (c) Discuss the results presented in the table:
      – What do you observe about the scalability of the A* search algorithm? How does the time and memory consumption increase as the input size increases? Are these observations surprising or expected, considering the asymptotic time and space complexity of the algorithm? Please explain.
      – How does the A* search algorithm explore the search space, with h_1 vs. with h_2? Are these observations surprising or expected, considering the admissibility/monotonicity features of the functions? Please explain.

[1]Reminder: The step cost function and the heuristic function return positive numbers $\geq \epsilon > 0$, at every non-goal state.

[2]Attention: Some A* implementations available online are incorrect or incomplete. Make sure that your own implementations match the algorithms taught in class.

[3]Attention: In the demos, you are expected to show one example for each difficulty level.

Submit

- A pdf copy of a description of[4]
  - your formulations of the Color-Maze puzzle (i.e., search model and heuristic functions), and
  - experimental evaluation of your A* search implementation on the Color-Maze instances (i.e., table and discussion).
- A zip file containing the following:
  - Your Python code for the algorithm, with comments describing your solution.
  - Several test boards you created for the 4'th part of the assignment, and the corresponding solutions found by your implementations.

In each one of the deliverables above, please include your name and student id.

Demos If your submitted implementation runs correctly and you can demonstrate it with your test instances successfully (without any bugs), then you will be invited to make a demo of your implementations. The 3'rd and the 4'th parts of the assignment will be graded at the demos.[5] The demos are planned for the week following the deadline and will be scheduled later on.

Collaboration You are allowed to work with another classmate. In that case, each team should submit one report in pdf, and one zip file at SUCourse+. Both team members should be present at the demos, if the team would like to make a demo of their implementation.

[4]Suggestion: You can use Overleaf for editing in LaTeX: `https://www.overleaf.com/`.

[5]Attention: If your implementation does not run correctly with any of your instances, you will not be invited to the demos and thus no credit will be given to the 3'rd and the 4'th parts.