Artificial Intelligence
Machine Problem 1 – A* for Solving a Maze

IMPORTANT NOTICE: You do not have permission to share the description of this assignment or discuss it with anyone outside the class. Also, you cannot share your code (or solution in general) with anyone, nor can you ask anyone for the code or solution. Failure to adhere to this policy will result in a zero grade for the assignment and may result in a dismissal from Lewis University.

Introduction

For this assignment, you will implement the A* algorithm to find an exit out of a maze. Your goal is to return the instructions for solving the maze and show the configuration after each move. The maze is a rectangular grid containing passageways, walls, and a single exit point that the agent must reach. The agent can move in four directions, but cannot move to a location containing a wall. The agent's position can wrap-around, meaning that if the agent is in the leftmost location and the rightmost location of the same row is not a wall, then the agent can still move left.

Requirements

For this assignment, you are given base Python 3 code that executes a uniform cost search for a similar maze problem, but without the possibility of wrap-around moves. So download this code first, and then do the following:

1. Modify that code so it solves the maze problem using the A* algorithm and allowing for wrap-around moves.

2. Make sure your heuristic takes into account the wrap-around possibility. Remember, heuristic needs to satisfy the admissibility and consistency properties.

3. Modify the print statements from the base code to include your info in the printed heading:
```
Artificial Intelligence
MP1: A* for Sliding Puzzle
SEMESTER: [put semester and year here]
NAME: [your name here]
```

Additional Requirements

1. The name of your source code file should be `mp1.py`. All your code should be within a single file.
2. You can only import numpy and queue packages.
3. Your code should follow good coding practices, including good use of whitespace and use of both inline and block comments.
4. You need to use meaningful identifier names that conform to standard naming conventions.
5. At the top of each file, you need to put in a block comment with the following information: your name, date, course name, semester, and assignment name.
6. The output should **exactly** match the sample output shown on the last page. Note that for a different input state, the output may be different. I will be testing on a different input than shown in the sample.

What to Turn In

You will turn in the single mp1.py file using BlackBoard.

Grading Rubric

Category	**Unsatisfactory (0-1 points)**	**Satisfactory (2-3 point)**	**Distinguished (4-5 points)**
Program Correctness	• Program does not execute due to errors • Incorrect results for most or all input	• Program works and completes most tasks appropriately • Program fails to work for special cases	• Program runs and completes all required tasks • Handles any required special cases • Executes without errors
Programming Style	• No name, date, or assignment title included • Poor use of white space • Disorganized and messy • No or few comments in the source code • Poor use of variables (improper scope/visibility, ambiguous naming).	• Includes name, date, and assignment title. • White space makes program fairly easy to read. • Well organized code. • Some comments missing in the source code or too many comments • Good use of variables (few issues with scope/visibility or unambiguous naming).	• Includes name, date, and assignment title. • Excellent use of white space. • Perfectly organized code. • Source code is commented throughout when needed • Excellent use of variables (no issues with scope/visibility or unambiguous naming).
Following Specifications	• Incorrect filenames • Incorrect specified identifier names • Source code organization different from requirements • Additional requirements not satisfied	• Correct filenames and class names • Few issues with other specified identifier names • Source code organization close to requirements • Some additional requirements not satisfied	• Correct filenames and specified identifier names • Source code organization satisfies all requirements • All additional requirements satisfied

Sample Program Output 1

```
Artificial Intelligence
MP1: A* search algorithm implementation for a maze
SEMESTER: Spring 2021
NAME: [your name]

START
[[0 1 1 0 1]
 [0 4 0 0 1]
 [0 0 1 1 1]
 [1 0 0 0 0]
 [1 1 0 1 0]
 [1 0 0 1 0]
 [1 0 0 1 0]
 [1 0 1 1 0]
 [1 0 0 0 0]
 [0 2 1 1 1]]
Move 1 ACTION: left
[[0 1 1 0 1]
 [4 0 0 0 1]
 [0 0 1 1 1]
 [1 0 0 0 0]
 [1 1 0 1 0]
 [1 0 0 1 0]
 [1 0 0 1 0]
 [1 0 1 1 0]
 [1 0 0 0 0]
 [0 2 1 1 1]]
Move 2 ACTION: up
[[4 1 1 0 1]
 [0 0 0 0 1]
 [0 0 1 1 1]
 [1 0 0 0 0]
 [1 1 0 1 0]
 [1 0 0 1 0]
 [1 0 0 1 0]
 [1 0 1 1 0]
 [1 0 0 0 0]
 [0 2 1 1 1]]
Move 3 ACTION: up
[[0 1 1 0 1]
 [0 0 0 0 1]
 [0 0 1 1 1]
 [1 0 0 0 0]
 [1 1 0 1 0]
 [1 0 0 1 0]
 [1 0 0 1 0]
 [1 0 1 1 0]
 [1 0 0 0 0]
 [4 2 1 1 1]]
Move 4 ACTION: right
[[0 1 1 0 1]
 [0 0 0 0 1]
 [0 0 1 1 1]
 [1 0 0 0 0]
 [1 1 0 1 0]
 [1 0 0 1 0]
 [1 0 0 1 0]
 [1 0 1 1 0]
 [1 0 0 0 0]
 [0 4 1 1 1]]

Number of states visited = 7
```