ASSIGNMENT #1 – COMP 3106 INTRODUCTION TO ARTIFICIAL INTELLIGENCE

The assignment is an opportunity to demonstrate your knowledge on informed graph search and practice applying it to a problem.

The assignment may be completed individually, or it may be completed in small groups of two or three students. The expectations will not depend on group size (i.e. same expectations for all group sizes).

Assignment due date: 2024-10-01

Assignments are to be submitted electronically through Brightspace. It is your responsibility to ensure that your assignment is submitted properly. Copying of assignments is NOT allowed. Discussion of assignment work with others is acceptable but each individual or small group are expected to do the work themselves.

Components

The assignment should contain two components: an implementation and answers to the questions below.

Implementation

Programming language: Python 3

You may use the Python Standard Library (https://docs.python.org/3/library/). You may also use the NumPy, Pandas, and SciPy packages (and any packages they directly depend on). Use of any additional packages requires approval of the instructor.

You must implement your code yourself. Do not copy-and-paste code from other sources, but you may use any pseudo-code we wrote in class as a basis for your implementation. Your implementation must follow the outlined specifications. Implementations which do not follow the specifications may receive a grade of zero. Please make sure your code is readable, as it will also be assessed for correctness. You do not need to prove correctness of your implementation.

You may be provided with a set of examples to test your implementation. Note that the provided examples do not necessarily represent a complete set of test cases. Your implementation may be evaluated on a different set of test cases.

The implementation will be graded both on content and use of good programming practices.

Submit the implementation as a single PY file.

Questions

You must answer all questions posed below. Ensure you answers are clear and concise.

If the assignment was completed in a small group of students, you must also include a statement of contributions. This statement should identify: (1) whether each group member made significant contribution, (2) whether each group member made an approximately equal contribution, and (3) exactly which aspects of the assignment each group member contributed to.

Submit your answers to the questions as a single PDF file.

Implementation

Consider the problem of finding a path from a start position to an end position in a grid-like environment. Implement an algorithm using A* search to find the optimal path to goal square.

In this problem, assume we have an agent that starts at a start state on an m x n rectangular grid. The agent tries to find a path from the start state to the goal state by moving to adjacent horizontal or vertical squares on the grid (cannot move diagonally). Assume the environment is fully observable; assume there always exists a path from the start state to the goal state.

In this environment, there are doors and keys. Your agent can only traverse a square with a door on it if your agent has a key. Your agent can pick up a key by traversing a square with a key on it. A key can be used to upon any door, but each key can only be used to open a door once. There is no limit on the number of keys your agent can pick up. Once a door is opened, that square can be traversed again without using another key.

Assume our m x n grid can have the following types of squares:

1. The start square (indicated by the letter “S”).
2. The goal square (indicated by the letter “G”).
3. A square with a key (indicated by the letter “K”).
4. A door (indicated by the letter “D”).
5. A regular square (indicated by the letter “O”).

In all cases, the cost of moving to an adjacent square is 1.

Your implementation must contain a single file named “assignment1.py” with a function named “pathfinding”, which takes one input argument (you may have other variables/functions/classes in your file). The input argument should be the full file path to a comma separated value (CSV) file that contains the input grid.

The CSV file will contain a grid describing the environment. The CSV will contain a rectangular grid with one S (indicating the start square), one G (indicating the goal square), zero or more K (indicating keys), zero or more D (indicating doors), and zero or more O (indicating regular squares).

Your function should return three values.

The first returned value should be a list of tuples indicating the coordinates of the squares occupied along the optimal path, in order. Assume the first index indicates the row and the second index indicates the column. Both row and column indices are integers and start at zero.

The second returned value should be the cost of the optimal path.

The third returned value should be the number of states explored during A* search.

Attached are example CSV files and corresponding example text files containing the optimal path, cost of the optimal path, and number of explored nodes. Note that your function should not write anything to file. These examples are provided in separate files for convenience. Also attached is skeleton code indicating the format your implementation should take.

Note that for some examples there may be multiple correct solutions. The number of explored nodes may also be different depending on the heuristic you choose. All correct solutions will be accepted.

Example CSV file:
S,K,O
D,D,O

G,O,O

Example first output (i.e. optimal path):
[(0, 0), (0, 1), (1, 1), (2, 1), (2, 0)]

Example second output (i.e. cost of optimal path):
4

Example third output (i.e. number of states explored):
7

Grading

The implementation will be worth 70 marks.

50 marks will be allocated to correctness on a series of test cases, with consideration to each of the three outputs (i.e. optimal path, optimal path cost, and number of states explored). These test cases will be run automatically by calling your implementation from another Python script. To facilitate this, your implementation must adhere exactly to the specifications.

20 marks will be allocated to human-based review of code. This human-based review will consider both correctness and use of good programming practices.

Questions

Please answer the following questions. For all questions, explain why your answers are correct.

1. What type of agent have you implemented (simple reflex agent, model-based reflex agent, goal-based agent, or utility-based agent)? [3 marks]
2. Is the task environment: [7 marks]
    a. Fully or partially observable?
    b. Single or multiple agent?
    c. Deterministic or stochastic?
    d. Episodic or sequential?
    e. Static or dynamic?
    f. Discrete or continuous?
    g. Known or unknown?
3. What heuristic did you use for A* search for this environment? Show that your heuristic is consistent. [8 marks]
4. Suggest a particular instance of this problem (i.e. grid) where A* search using your heuristic would find the optimal solution faster than uniform cost search. [4 marks]
5. Suggest a particular instance of this problem (i.e. grid) where a greedy heuristic search using your heuristic would not find the optimal solution. [4 marks]
6. Consider a modification of this problem where one key can be used to open an unlimited number of doors. Determine whether your heuristic is still consistent. [4 marks]

Grading

The technical document will be worth 30 marks, allocated as described above.