ASSIGNMENT #1 – COMP 3106 ARTIFICIAL INTELLIGENCE

The assignment is an opportunity to demonstrate and solidify your knowledge on informed graph search.

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).

Components

The assignment should contain two components: an implementation and a technical document.

Implementation

The implementation should provide a complete solution in Python 3 to the problem outlined below.

Technical Document

Your technical document should answer any questions posed below. Use these questions to elaborate on your implementation. After reading the technical document, the reader should understand how your implementation works. The technical document will be graded both on content and on presentation quality.

If the assignment was completed in a small group of students, the technical document must 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.

Logistics

Assignment due date: Monday, October 18, 2021

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

Implementation

Programming language: Python 3

You may use any standard Python libraries. You may also use the NumPy and SciPy packages. 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.

Technical Document

Your technical document must answer all questions posed below. Ensure you answers are clear and concise. Submit the technical document as a single PDF file.

Implementation

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

In this problem, assume we have an agent that starts at a start state on an m x n 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). The cost of moving to an adjacent square is 1. There may also be hazards on the grid which cannot be traversed nor can any square horizontally or vertically adjacent to it be traversed. Assume the environment is fully observable; assume there always exists a path from the start state to the goal state.

Suppose we have an m x n grid where each square in the grid can be one of the following:
The start state (indicated by the letter "S")
The goal state (indicated by the letter "G")
A wall/obstacle that cannot be traversed (indicated by the letter "X")
A hazard that cannot be traversed nor can any square adjacent to it be traversed (indicated by the letter "H")
A regular square which can be visited provided it is not adjacent to a hazard (indicated by the letter "O")

Your implementation must contain a single file named "assignment1.py" with a function named "pathfinding", which takes one input argument. 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 state), one G (indicating the goal states), zero or more X (indicating walls/obstacles), zero or more H (indicating hazards), and zero or more O (indicating regular squares).

Your function should return three outputs.

The first output should be a list of tuples along the optimal path from the start state to a goal state (including the start state and the goal state). Assume the first index indicates the row and the second index indicates the column. Both row and column indices start at zero.

The second output should be list of tuples that were explored during the A* search, in order (including the start state and the goal state). Assume the first index indicates the row and the second index indicates the column. Both row and column indices start at zero.

The third output should be the cost of the optimal path from the start state to the goal state.

Attached to this file are an example CSV file and the corresponding example text files containing the optimal path and list of explored nodes. Note that for some examples their may be multiple correct solutions. All correct solutions will be accepted. Also attached is skeleton code indicating the format your implementation should take.

Example CSV file:
S,O,O
O,X,O
O,X,O
H,O,G

Example first output (i.e. optimal path):
[(0, 0), (0, 1), (0 ,2), (1, 2), (2, 2), (3, 2)]

Example second output (i.e. list of states explored):
[(0, 0), (1, 0), (0, 1), (0 ,2), (1, 2), (2, 2), (3, 2)]

Example third output (i.e. optimal path cost):
5

Grading

The implementation will be worth 60 marks.

40 marks will be allocated to correctness on a series of test cases, with consideration to each of the three outputs (i.e. optimal path, list of explored states, and optimal path cost).

20 marks will be allocated to human-based review of code.

Technical Document

Please answer the following questions in the technical document. Explain why your answers are correct.

1. Briefly describe how your implementation works. Include information on any important algorithms, design decisions, data structures, etc. used in your implementation. [10 marks]
2. What type of agent have you implemented (simple reflex agent, model-based reflex agent, goal-based agent, or utility-based agent)? [3 marks]
3. What heuristic should be used for A* search for this environment? Show that this heuristic is consistent. Show that this heuristic is more informed than some alternative heuristic. [8 marks]
4. Suggest a particular instance of this problem where A* search would find the optimal solution faster than uniform cost search. [4 marks]
5. Suggest a particular instance of this problem where a greedy heuristic search would not find the optimal solution. [4 marks]
6. What strategy should be used to break ties when two nodes on the frontier have equal priority function? [3 marks]
7. Consider a variant of this problem where the agent can move to adjacent diagonal squares. What would be the best heuristic to use in A* search if the agent could also make such diagonal moves? [8 marks]

Grading

The technical document will be worth 40 marks, allocated as described above.