COMP2050 : Artificial Intelligence (Spring 2024)

Assignment 01

2024-03-11

Instructions

- Submit your assignment via Canvas.
- Your submission should include the zip file only (code + written part)
- You may use LaTex or Word to create the written part of your assignment, but you must submit it in PDF format.

1 Problem 1 (5 points)

Professor Khoa has a VinFast VF8. He wants to drive from Hanoi to Ho Chi Minh City next summer. He needs to drive through some plateau, uphill, and downhill road segments. Each segment will cost you some energy. For an electric car, when you go downhill, it will regenerate some energy[1], thus negative energy. He wants to minimize the number of times he must stop and charge his car. So he created a graph on different routes from Hanoi to Ho Chi Minh City with each node representing the beginning and the end of each segment and the cost between 2 nodes as the energy the car consumes. Then, Professor Khoa tries to use UCS for that graph.

1. Could he find the optimal route using UCS? Why or Why not?
2. Since he creates the graph, he knows that the energy consumed between 2 places on the graph is bounded below by a number α. So he modifies the graph by adding $\alpha + 1$ to all the edges. Can he find the optimal route using UCS and why?
3. Does your answer change if α is exactly equal to the minimum cost edge?

2 Problem 2 (5 points)

Construct a search problem where using Depth-First Search (DFS) is better than Breadth-First Search (BFS). Explain why is it better in your case. In general, what kind of problem that usually better to use DFS than BFS.

3 Problem 3 (5 points)

Given the following problem, describe its state representation, successor functions (with actions and plausible cost), initial state, and goal test.

The water-jar problem[2]: Given a water source and 3 jars of possibly different capacities (e.g 5,8 and 12 liters). Collect a given amount of water (e.g., 9 liters) in one of the jugs as fast as possible. The available operations are:

[1] https://en.wikipedia.org/wiki/Regenerative_braking
[2] https://en.wikipedia.org/wiki/Einstellung_effect

1. Fill a jar completely using the water source.
2. Pour from one jar to the other until the other one is full or the pouring jar is empty.
3. Empty the jar completely.

4 Coding Problem (10 points)

In this problem, you will try to implement Depth First Search, Breadth First Search, and A* Search to solve the N-puzzle problem and the N-Queens problem in Python.

The code is provided and you only need to fill in the missing part. The information about code layout is as follows:

- **node.py**: This file contain abstract implementation of Node class. **DO NOT** modify this file
- **problems.py**: The incomplete implementation of the NPuzzleNode and NqueensNode classes ***(each correct implementation will give you 2 points)***
- **search.py**: The file that you need to implement the DFS, BFS, and Astar algorithm. The input of these functions is the root node of the search graph. ***(each correct implementation will give you 2 points)***
- **test.py**: A test helper that runs some tests for your function. ***Note*** that this file only contains simple test cases, we will have some hidden test cases to test your code.

You need to complete DFS, BFS, and Astar in your **search.py** file and the function **is_goal** and **generate_children evaluate_heuristic** in file **problems.py**.

Then you can run the following command to test your code

python -m unittest test.py

4.1 N-Puzzle

The first problem that you will solve is the N-puzzle (in your class or many materials that you might find, they usually use a specific version of it which is 8-Puzzle). Here we will use a bigger version which is fifteen puzzle[3]. You can see the following example

1	2	3	4
5	6	7	8
9	10	11	12
13		14	15

Our action is we can move (or swap the blank spot to any of the adjunction numbers) to get the goal state. In this example, to get the goal state, we will need to move the blank spot 2 steps to the right

[3] `https://en.wikipedia.org/wiki/15_Puzzle`

1	2	3	4
5	6	7	8
9	10	11	12
13	14		15

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

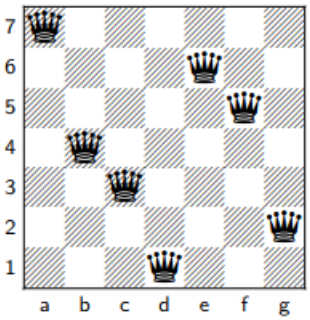
Every move has a cost of 1 and of course, when you use A* you will find the optimal solution if your heuristic is correct. In principle, there might be more than one optimal solution, you only need to find one.

4.2 N-Queens Puzzle

The next problem we will try to solve is the N-Queens Puzzle which many of you may know as 8-Queens Puzzle[4] in your lecture or any material you could find. Given $N \times N$ chess board, we will need to place N queens inside it such that:

- On each row, each column, there will always be at most one queen. (**Note**: This is a hard constraint)
- The number of conflicting queens is minimal (either diagonally or with a knight's move)

For example, the following is an optimal solution for 7-Queens board: there are 3 total attacks (2 diagonal and one a knight's move)



[4] https://en.wikipedia.org/wiki/Eight_queens_puzzle