

CS 6364-002 Homework 5

October 17, 2022

Requirements:

- Deadline for the first submission: **Oct-19-2022**.
- All assignments **MUST** have your name, student ID, course name/number at the beginning of your documents.
- Some hints are provided in the following google doc:
<https://docs.google.com/document/d/12p2ccUvfkzR6Aq33DABpnx1b2J3A-k1bxHhH6kuErTI/edit?usp=sharing>
- The following youtube video shows how to implement Tic Tac Toe AI with the minimax algorithm. I posted this link here in case you find it useful.
<https://www.youtube.com/watch?v=trKjYdBASyQ>
- Please write all the codes in one Jupyter notebook and run the codes to display the results in the notebook before you save the notebook (ipynb file). Pls zip the notebook and related files (if there are) into one file and submitted the zipped file.

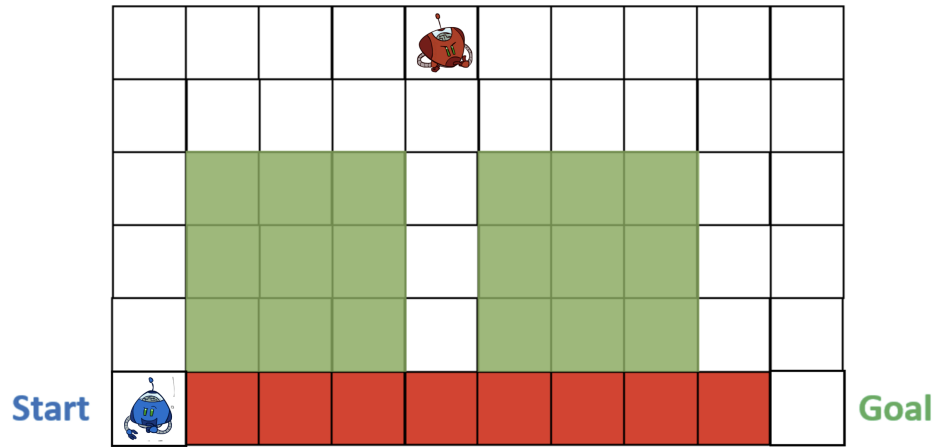
If you have any questions, please contact me.

In this question, we aim to solve the “cliff walking” zero-sum game using the **minimax search method with alpha-beta pruning with the depth limited to d layers (d=2 for part (a) and d =6 for part (b))** as introduced in Lecture 4-C. There are two players (the maximizing player and the minimizing player) who will control the movements of two agents respectively. There are one **blue agent** controlled by the maximizing player and **red agent** controlled by the minimizing player. The depth limit “d=2” means that each of the two agents can make one-step movement.

The size of the grid is 6×10 (see below). The **blue agent** starts at the leftmost cell in the bottom, that is, (6,1). The goal cell is the rightmost cell in the bottom (blue), that is, (6,10). All the cells with the green color refer to the regions with water. For the **blue agent**, the cost of each movement (action) in the white-colored cells (non-water region) is 1, and the cost of each movement in the green-colored cells (water region) has the cost 5. All the cells between (6,2) and (6,9) is the cliff (red). If the **blue agent** enters the cliff, which means the agent falls into the cliff, then the agent will die. The **red agent** starts at the cell (1, 5). The **red agent** does not consider costs for its movements. Its task is to block the direction of the blue agent, such that the blue agent will select a path with the minimal cost to the goal cell. **The cost of the path is the sum of the costs of the movements within the path.** The **red agent** will not enter the cliff and will be always alive. **Here, we define the utility of a path as “ $-1 \times$ the cost of this path,” such that the maximization of the utility is the same as the minimization of the path cost.**

The **blue agent** will move first. Both **blue** and **red** agents can move only one cell at a time to the neighboring cell, that is, up, down, right and left, unless the agent touches the border or the other agent. When the agent touches the border or the other agent, the action that makes the agent cross the border is not performed but it must remain stopped at the point waiting until the next action. For example, if the agent is at (1, 3) and the action is to up, then agent remains at that point, and if the next action is to right, then it moves to (1, 4), or when the next action is down then agent moves to (2, 3).

A suggested evaluation function of a state is “ $-1 \times$ the Manhattan distance of the blue agent to the goal state (6, 10)” + “the vertical distance between the red agent and the blue agent”, where the cliff cells should be avoided when you calculate the Mahanttan distrance. For example, the Manhattan distance of the blue agent in its initial position (6,1) to the goal position (6, 10) is 11. The vertical distance between the position of the blue agent (6,1)



and the position of the red agent (1,5) is 5. Therefore, the returned value of the evaluation function for the initial state is: $-1 \times 11 + 5 = 6$. Note that you can choose your own evaluation function as long as your own evaluation performs better than the preceding suggested one, meaning that the agents can make more rational actions based on your own evaluation function.

In this question, please finish the following two parts:

- [50%] Implement the **minimax search method (without alpha-beta pruning) with the depth limited to two layers (d=2)** and print out the returned paths of blue and red agents and the cost of the path for the blue agent.
- [25%] Implement the **minimax search method (without alpha-beta pruning) with the depth limited to six layers (d=6)** and print out the returned paths of blue and red agents and the cost of the path for the blue agent.
- [25%] Extend your implementation in part (b) and add alpha-beta pruning. Print out the returned paths of blue and red agents and the cost of the path for the blue agent.

[Note that, for each of the above three parts, if the blue agent will always be blocked by the red agent and can never reach the goal cell, you just print out the the paths of length 100.]