Written Assignment 3

Due: Friday 02/21/2025 @ 11:59pm EST

Disclaimer

I encourage you to work together, I am a firm believer that we are at our best (and learn better) when we communicate with our peers. Perspective is incredibly important when it comes to solving problems, and sometimes it takes talking to other humans (or rubber ducks in the case of programmers) to gain a perspective we normally would not be able to achieve on our own. The only thing I ask is that you report who you work with: this is **not** to punish anyone, but instead will help me figure out what topics I need to spend extra time on/who to help. When you turn in your solution (please use some form of typesetting: do **NOT** turn in handwritten solutions), please note who you worked with.

Question 1: Minimax Execution (25 points)

Consider the tree expansion:
Run the vanilla minimax algorithm on this tree (the root node is the `MAX` player and the two players `MAX` and `MIN` alternate turns) to determine what move the `MAX` player should make at the root. If you need a heuristic value, that the heuristic function is defined as follows:

$$h(n) := n.level + n.idx$$

where n is a node in the tree, $n.level$ is the 1-indexed level of that node in the tree (i.e. the root has level 1, the next layer has level 2, etc.), and $n.idx$ is the 1-indexed index of that node within its level (i.e. the left most node in a level has index 1, the node immediately to its right has index 2, etc.). Show all of your steps.

Question 1 Solution: The `MAX` player should make the move which takes it from state a to c.

Steps:

1. DFS traverses from a to b.
2. b is a terminal state with value 4.
3. DFS backtracks from b to a.
4. DFS traverses from a to c.
5. c is not a terminal state so explore its neighbors.
6. DFS traverses from c to h.
7. h is a terminal state with value 7.
8. DFS backtracks from h to c.
9. DFS traverses from c to i.
10. i is a terminal state with value 11.
11. DFS backtracks from i to c.
12. DFS traverses from c to j.
13. j is a terminal state with value 8.
14. DFS backtracks from j to c.
15. c has explored all neighbors. Since it is `MIN` turn, c is given utility value $\min(7, 11, 8) = 7$.
16. DFS backtracks from c to a.
17. DFS traverses from a to d.
18. d is not a terminal state so explore its neighbors.
19. DFS traverses from d to k.
20. k is a terminal state with value 9.
21. DFS backtracks from k to d.
22. DFS traverses from d to l.
23. l is a terminal state with value 2.
24. DFS backtracks from l to d.
25. d has explored all neighbors. Since it is `MIN` turn, d is given utility value $\min(9, 2) = 2$.
26. DFS backtracks from d to a.
27. DFS traverses from a to e.
28. e is not a terminal state so explore its neighbors.
29. DFS traverses from e to f.

30. f is a terminal state with value 11.
31. DFS backtracks from f to e.
32. DFS traverses from e to g.
33. g is a terminal state with value 3.
34. DFS backtracks from g to e.
35. e has explored all neighbors. Since it is `MIN` turn, e is given utility value $\min(11, 3) = 3$.
36. DFS backtracks from e to a.
37. a has explored all neighbors. Since it is `MAX` turn, a is given utility value $\max(4, 7, 2, 3) = 7$.
38. So `MAX` player chooses to go to state c to maximize the utility value.

Question 2: Iterative Deepening Execution (25 points)

Consider the tree expansion:
Run the Iterative Deepening algorithm on this tree (the root node is the `MAX` player and the two players `MAX` and `MIN` alternate turns) to determine what move the `MAX` player should make at the root. If you need a heuristic value, that the heuristic function is defined as follows:

$$h(n) := n.level + n.idx$$

where n is a node in the tree, $n.level$ is the 1-indexed level of that node in the tree (i.e. the root has level 1, the next layer has level 2, etc.), and $n.idx$ is the 1-indexed index of that node within its level (i.e. the left most node in a level has index 1, the node immediately to its right has index 2, etc.). Show all of your steps.

Question 2 Solution: The `MAX` player should make the move which takes it from state a to c.

Steps:

1. BFS begins at root node a.
2. BFS explores the first neighborhood of a.
3. BFS explores b.
4. b is a terminal state with value 4.
5. BFS explores c.
6. c is not a terminal state so it is given utility value $h(c) = 2 + 2 = 4$.
7. BFS explores d.
8. d is not a terminal state so it is given utility value $h(d) = 2 + 3 = 5$.
9. BFS explores e.
10. e is not a terminal state so it is given utility value $h(e) = 2 + 4 = 6$.
11. Utility value of a is updated. Since a is `MAX` turn, a is given utility value $\max(4, 4, 5, 6) = 6$.
12. BFS explores the second neighborhood of a.
13. BFS explores h.
14. h is a terminal state with value 7.
15. BFS explores i.
16. i is a terminal state with value 11.
17. BFS explores j.
18. j is a terminal state with value 8.
19. Utility value of c is updated. Since c is `MIN` turn, c is given utility value $\min(7, 11, 8) = 7$.
20. BFS explores k.
21. k is a terminal state with value 9.
22. BFS explores l.
23. l is a terminal state with value 2.
24. Utility value of d is updated. Since d is `MIN` turn, d is given utility value $\min(9, 2) = 2$.
25. BFS explores f.
26. f is a terminal state with value 11.
27. BFS explores g.
28. g is a terminal state with value 3.
29. Utility value of e is updated. Since e is `MIN` turn, e is given utility value $\min(11, 3) = 3$.
30. Utility value of a is updated. Since a is `MAX` turn, a is given utility value $\max(4, 7, 2, 3) = 7$.
31. So `MAX` player chooses to go to state c to maximize the utility value.