CS 470 Homework
Adversarial Search
80 Points

---

1. **Minimax and Alpha-Beta [30 points]** Consider the zero-sum game tree shown in Figure 1, where upward pointing triangles indicate MAX nodes, downward pointing triangles indicate MIN nodes, and squares (each labeled with a letter) indicate terminal nodes. In this problem you will assign utility values to the terminal nodes. Remember that the values you assign indicate utility from the MAX player's perspective.

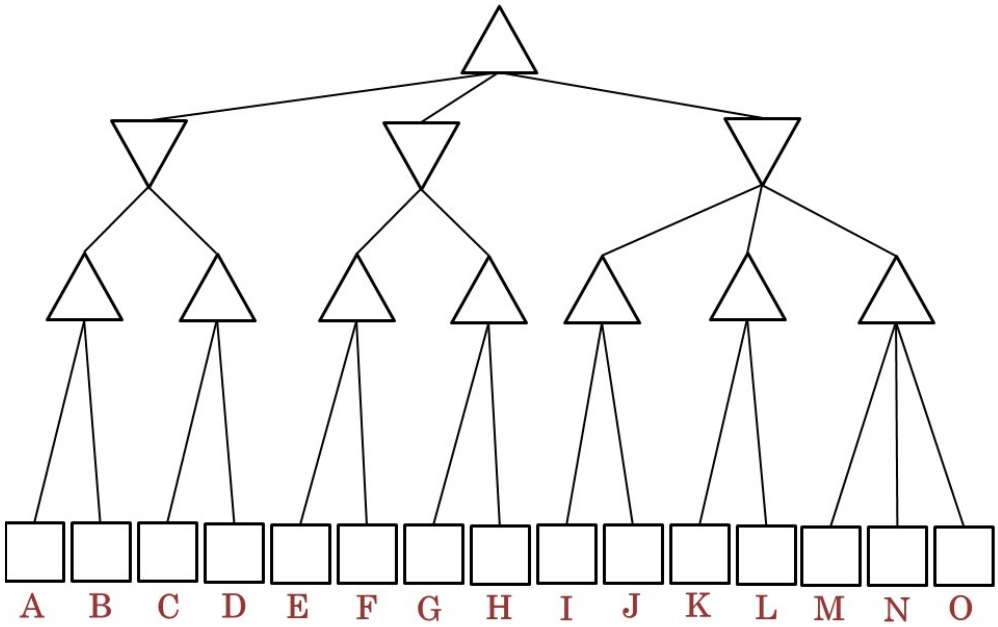

Figure 1: Game Tree for Question 1

   (a) **[7 points]** Assign the utility values 1 through 15 to the terminal nodes, using each once, so that the minimax value at the root MAX node is as high as possible.
      i. **[1 points]** What utility value did you assign to each terminal node?
      ii. **[5 points]** What is the minimax value of the root node in your game?
      iii. **[1 points]** Which action should the MAX player select at the root node in your game?

   (b) **[7 points]** Assign the utility values 1 through 15 to the terminal nodes, using each once, so that the minimax value at the root MAX node is as low as possible.

i. **[1 points]** What utility value did you assign to each terminal node?

ii. **[5 points]** What is the minimax value of the root node in your game?

iii. **[1 points]** Which action should the MAX player select at the root node in your game?

(c) **[8 points]** Assign the utility values 1 through 15 to the terminal nodes, using each once, so that the alpha-beta algorithm prunes as many nodes as possible. (If a Min/Max node is pruned, count it and its children in your count)

i. **[1 points]** What utility value did you assign to each terminal node?

ii. **[1 points]** What is the minimax value of the root node in your game?

iii. **[5 points]** Which nodes are pruned in your arrangement?

iv. **[1 points]** Which action should the MAX player select at the root node in your game?

(d) **[8 points]** Now, assign the utility values 1 through 15 to the terminal nodes, using each once, so that alpha-beta algorithm prunes exactly 6 nodes. (If a Min/Max node is pruned, count it and its children in your count)

i. **[1 points]** What utility value did you assign to each terminal node?

ii. **[1 points]** What is the minimax value of the root node in your game?

iii. **[5 points]** Which nodes are pruned in your arrangement?

iv. **[1 points]** Which action should the MAX player select at the root node in your game?

2. **Alpha-Beta [20 points]**

   A zero-sum game tree is shown in Figure 2. The triangles pointing up are MAX nodes, the triangles pointing down are MIN nodes, and the squares are terminal nodes, with the utility for MAX player indicated in the center of each square.

   Your task is to run alpha-beta pruning on this game tree to compute the optimal decision for the max player at the root node.

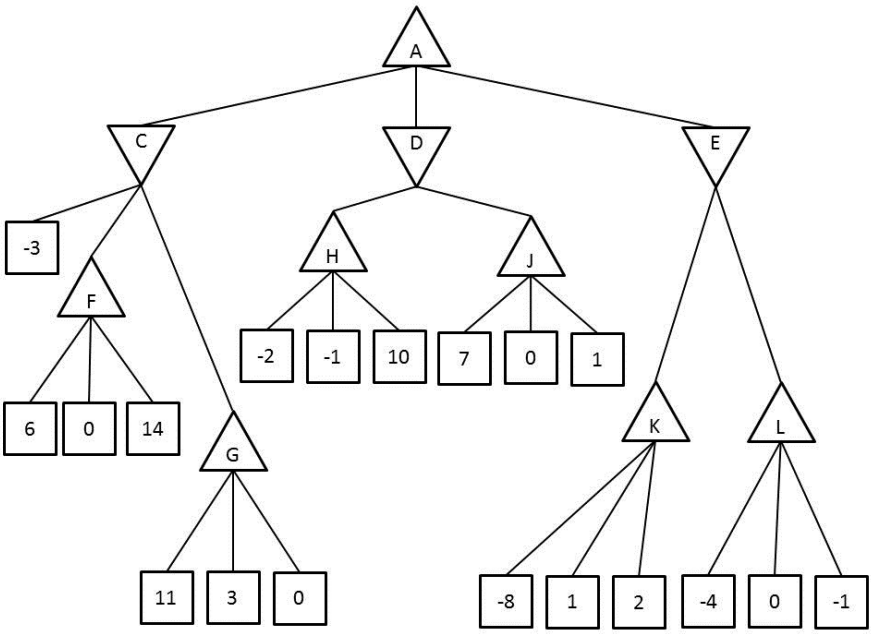

Figure 2: Game Tree for Question 2

(a) **[16 points]** List (or mark with an "X" on a copy of the figure) all of the nodes that were not visited by the search due to pruning. Assume the search expands successors in left-to-right order. If you list the pruned nodes, refer to the utility nodes by their parent node and then value (i.e. the left-most -3 utility node would be C-3).

(b) **[2 points]** What is the minimax value of the root node?

(c) **[2 points]** Which move should the MAX player select at the root node (action leading to C, D, or E)?

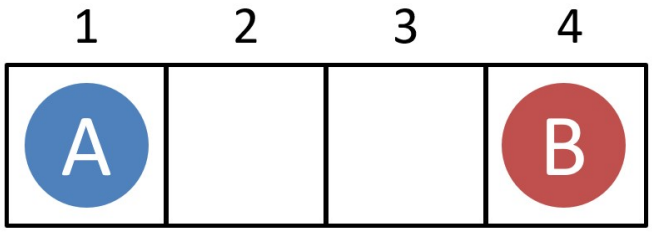

Figure 3: The starting position of a simple game. Player A moves first. The two players take turns moving, and each player must move his token to an open adjacent space in either direction. If the opponent occupies an adjacent space, then a player may jump over the opponent into the next open space, if any. (For example, if A is on 3 and B is on 2, then A may move back to 1.) The game ends when one player reaches the opposite end of the board. If player A reaches space 4 first, then the value of the game to A is +1; if player B reaches space 1 first, then the value of the game to A is -1.

3. **Minimax-Values** (5.8 from R&N) **[30 points]**

   Consider the two-player game shown in Figure 3

   (a) **[10 points]** Draw the complete game tree, using the following conventions:
   - Write each state as (s_A, s_B), where s_A and s_B denote the token locations
   - Put each terminal state in a square box and write its game value in a circle
   - Put *loop states* (states that already appear on the path to the root) in double square boxes. Since their value is unclear, annotate each with a "?" in a circle.

   (b) **[10 points]** Now mark each node with its backed-up minimax value (also in a circle). Explain how you handled the "?" values and why.

   (c) **[10 points]** Explain why the standard minimax algorithm would fail on this game tree and briefly sketch how you might fix it, drawing on your answer to the previous question. Does your modified algorithm give optimal decisions for all games with loops?