Adversarial Search Review

Artificial Intelligence

1. What is a zero-sum game?

> **Solution:** A game in which the "score" for one agent is the negative of the other agent's, i.e., they add to zero.

2. What is another,perhaps better, term for zero-sum? Why?

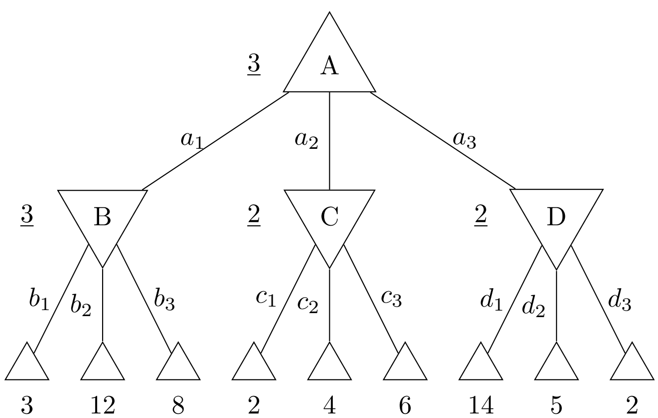
> **Solution:** Equivalently, in a *constant sum* game the scores add to a constant value. For example, in chess each player gets a score of 1 or 0 in the case of a win, or each player gets $\frac{1}{2}$ in the case of a draw. These scores always add to 1.

3. What is a ply?

> **Solution:** One move by one player is called a ply. A ply for MAX plus the response ply for MIN constitute a game move.

4. In the following 2-ply minimax game tree, what are the minimax values of nodes A, B, C, and D, and which move is selected by MAX?

   Move a_1 is selected.



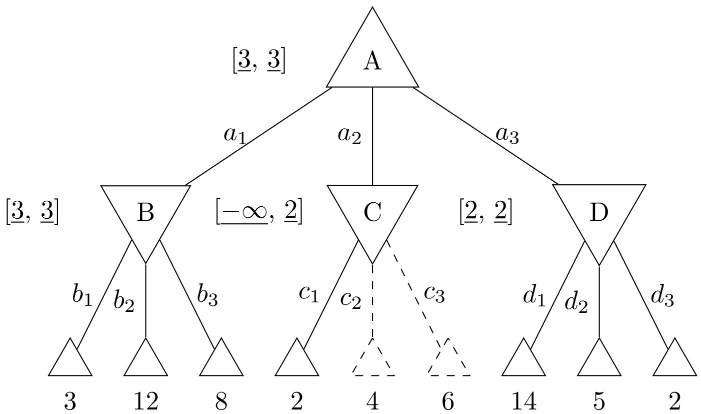
5. In the following game tree, what are the α and β values in the intervals and which branches would be pruned from the tree with Alpha-Beta pruning?

   Remember,

   - α = the value of the best (i.e., highest-value) choice we have found so far at any choice point along the path for MAX. Think: α = "at least."
   - β = the value of the best (i.e., lowest-value) choice we have found so far at any choice point along the path for MIN. Think: β = "at most."

   At a MIN node, once we exapnd a child node that sets β to a value less than the highest α value for a sibling node, we can stop expanding child nodes because they won't change the choice taken at the parent MAX node.

   At a MAX node, once we exapnd a child node that sets α to a value greater than the lowest β value for a sibling node, we can stop expanding child nodes because they won't change the choice taken at the parent MIN node.



6. In basic minimax search, the $Minimax$ value function is defined by:

$$Minimax(s) = \begin{cases} Utility(s, MAX) & \text{if } IsTerminal(s) \\ max_{a \in Actions(s)} Minimax(Result(s,a)) & \text{if } ToMove(s) = MAX \\ min_{a \in Actions(s)} Minimax(Result(s,a)) & \text{if } ToMove(s) = MIN \end{cases}$$

   How does the minimax algorithm and the *minimax* value function change when using a heuristic static evaluation function?

   Solution: We add a depth parameter, d, which limits the number of plies of the game tree generated by the algorithm, and use this updated $Minimax$ function:

$$HMinimax(s,d) = \begin{cases} Eval(s, MAX) & \text{if } IsCutoff(s,d) \\ max_{a \in Actions(s)} HMinimax(Result(s,a), d+1) & \text{if } ToMove(s) = MAX \\ min_{a \in Actions(s)} HMinimax(Result(s,a), d+1) & \text{if } ToMove(s) = MIN \end{cases}$$

7. In terminal states, what is the value of the $Eval(s, MAX)$ function?

   Solution: In terminal states, $Eval(state, player) = Utility(state, player)$, which is defined by the game rules.

8. How does Monte-Carlo Tree Search differ from Heuristic Alph-Beta Search?

Solution: Instead of searching to a given depth and applying a heurstic evaluation function to the resulting positions, we

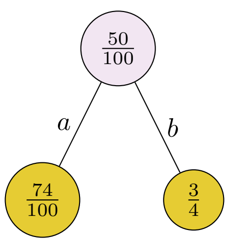
- simulate complete games (from a given position) to terminal positions, and
- back-up the win/loss scores up the tree.

9. Which weaknesses of Heuristic Alph-Beta Search does MCTS seek to overcome?

Solution: Two major weaknesses of heuristic alpha-beta tree search:

- Can't handle high branching factors. Go has a branching factor that starts at 361, which means alpha–beta search would be limited to only 4 or 5 ply.
- Can't always define a good static evaluation function. E.g., in Go material value is not a strong indicator and most positions are in flux until the endgame.

10. Using the $UCB1$ upper confidence bound selection policy with a low C value, which path, a or b, would MCTS expand?



Solution: a