ICS 171, Summer 2000: Homework 4 Solutions

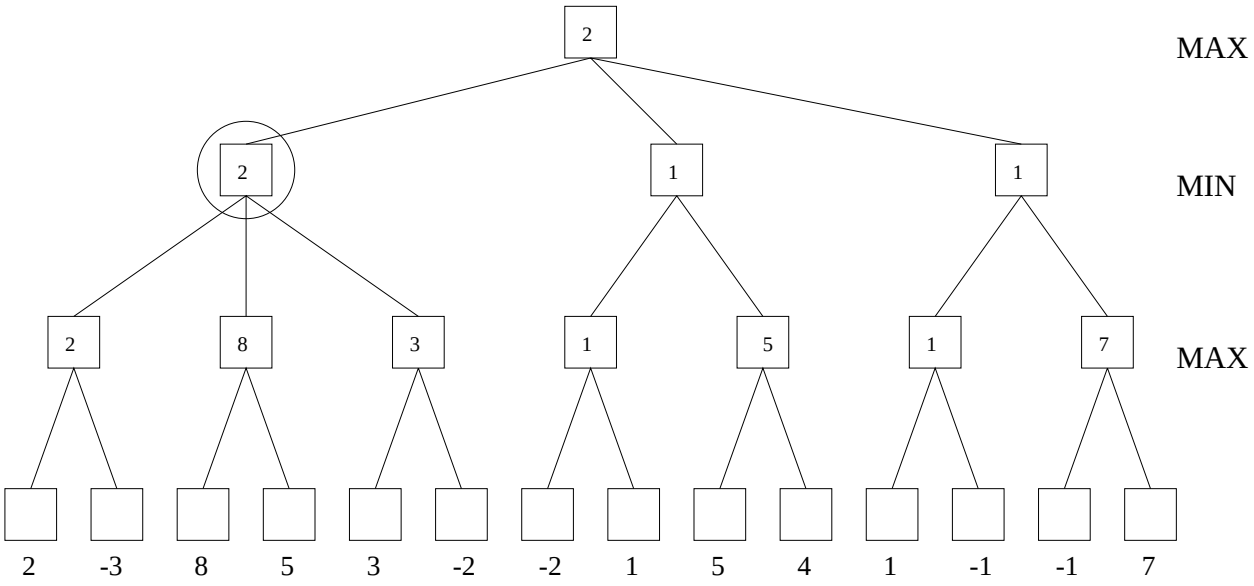
(1) Design evaluation functions for any two of backgammon, chess, checkers, tic-tac-toe, othello, connect-4 or any two board games of your choice.

There are many possible answers here. Your evaluation function needs to do at least two things: (1) It needs to map board states back to a real valued number, and (2) It needs to adjust the score for both the players and opponents configurations (e.g., score = quality of state for player 1 - quality of state for player 2).

See question 5.1 for an example of an evaluation function for tic-tac-toe.

Note that a particularly bad evaluation function for tic-tac-toe is something like: f = number of X's - number of O's. This function clearly fails to distinguish between any board states at the same depth in the search: all boards at the same depth will have identical f values since players alternate moves.

(2) Shown below is a game tree where the root node is a MAX node.

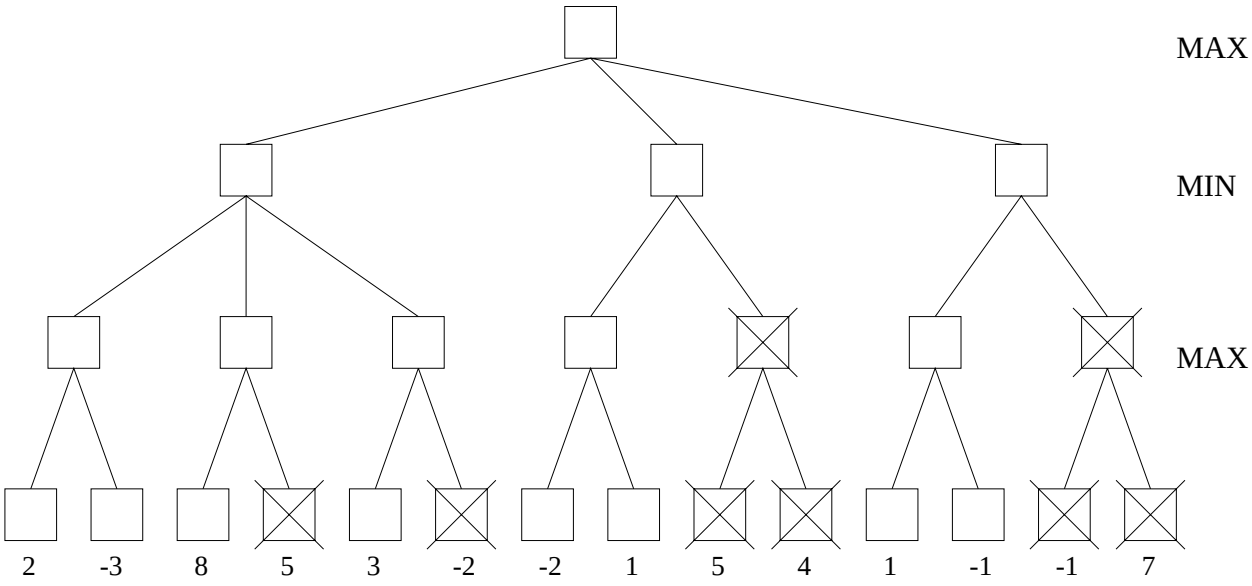


Assume that:

- the tree is explored by minimax (and alpha-beta) in a left to right manner
- the tree is explored to depth 3 and no further
- the numbers beneath the leaves of the tree are the evaluation function values for the corresponding states

Write in the boxes the minimax values for each state. Indicate the move chosen by MAX (the computer) as its first move.

(3) Shown below is the same game tree as in Question 2. Again, the root node is a MAX node. Which states will not be evaluated in minimax search with alpha-beta pruning? Show the nodes that are not evaluated by marking them with an **X**. By "not evaluate" we mean that no minimax values are calculated for that node.



(4) The minimax algorithm returns the best move for MAX under the assuption that MIN plays optimally. What happens when MIN plays suboptimally? (Russell & Norvig, page 148).

The outcome for MAX can only be the same or better if MIN plays suboptimally compared to MIN playing optimally. If a deterministic model is available for MIN's irrationality, then it can be applied to the game tree in the same way as the optimal policy. However, in all real games the opponent is only "reasonable" in some unspecified way and a pure minimax strategy can do far worse than some other schemes. Suppose MAX assumes MIN is rational and minimax says MIN will win. In such cases, all moves are losing and are "equally good", including those that lose immediately! A better algorithm would make moves for which it will be very difficult for MIN to find the winning line. Notice also that minimax never "sets traps." (solution from R & N)