Module 3

Problem Solving using Search- (Two agent)

Lesson 8

Two agent games : alpha beta pruning

Version 1 CSE IIT, Kharagpur

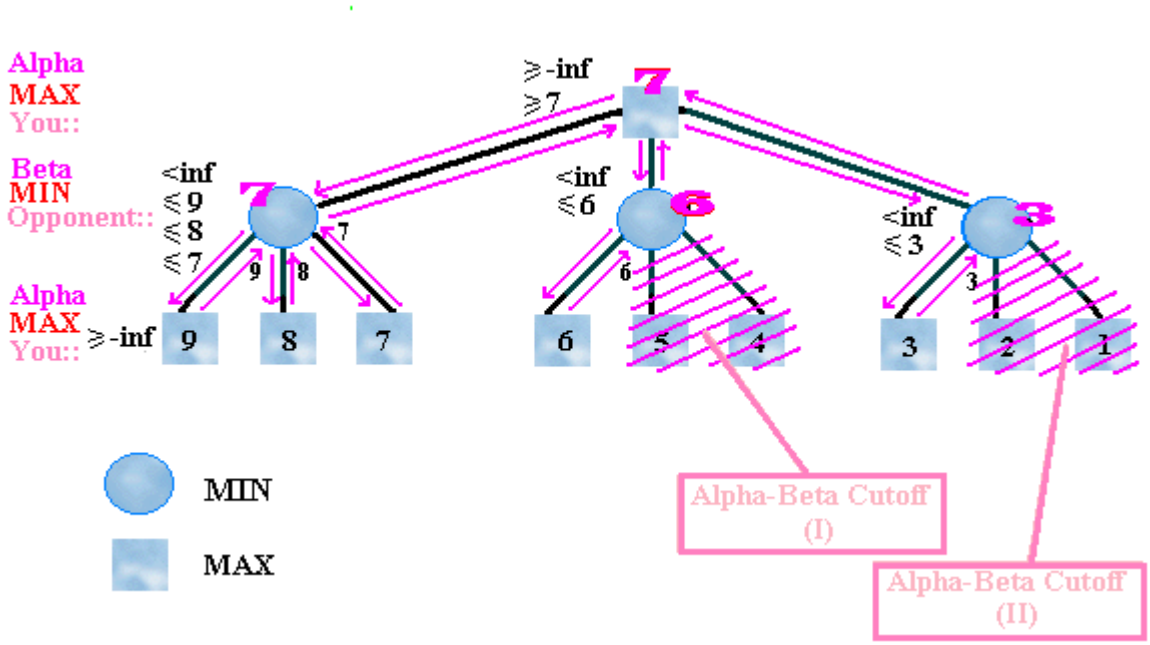
3.5 Alpha-Beta Pruning

ALPHA-BETA pruning is a method that reduces the number of nodes explored in Minimax strategy. It reduces the time required for the search and it must be restricted so that no time is to be wasted searching moves that are obviously bad for the current player. The exact implementation of alpha-beta keeps track of the best move for each side as it moves throughout the tree.

We proceed in the same (preorder) way as for the minimax algorithm. For the **MIN** nodes, the score computed starts with **+infinity** and decreases with time. For **MAX** nodes, scores computed starts with **-infinity** and increase with time.

The efficiency of the *Alpha-Beta* procedure depends on the order in which successors of a node are examined. If we were lucky, at a MIN node we would always consider the nodes in order from low to high score and at a MAX node the nodes in order from high to low score. In general it can be shown that in the most favorable circumstances the alpha-beta search opens as many leaves as minimax on a game tree with double its depth.

Here is an example of Alpha-Beta search:



3.5.1 Alpha-Beta algorithm:

The algorithm maintains two values, alpha and beta, which represent the minimum score that the maximizing player is assured of and the maximum score that the minimizing player is assured of respectively. Initially alpha is negative infinity and beta is positive infinity. As the recursion progresses the "window" becomes smaller.

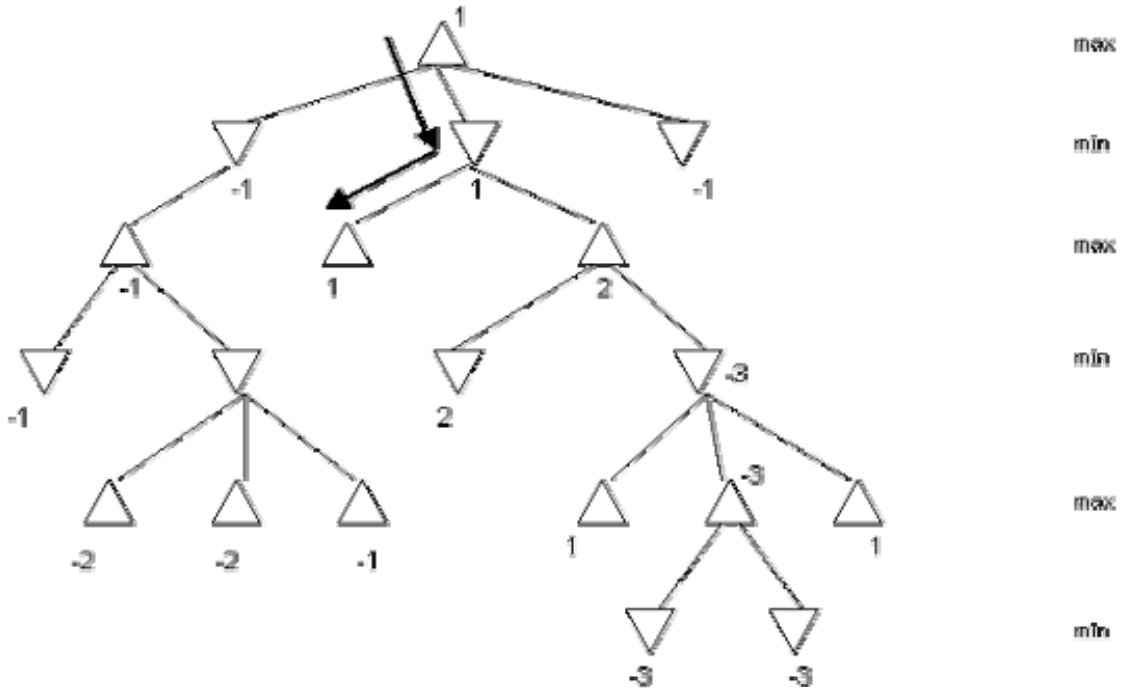
When beta becomes less than alpha, it means that the current position cannot be the result of best play by both players and hence need not be explored further.

Pseudocode for the alpha-beta algorithm is given below.

```
evaluate (node, alpha, beta)
    if node is a leaf
        return the heuristic value of node
    if node is a minimizing node
        for each child of node
            beta = min (beta, evaluate (child, alpha, beta))
            if beta <= alpha
                return beta
        return beta
    if node is a maximizing node
        for each child of node
            alpha = max (alpha, evaluate (child, alpha, beta))
            if beta <= alpha
                return alpha
        return alpha
```

Questions

1. Suppose you and a friend of yours are playing a board game. It is your turn to move, and the tree below represents your situation. The values of the evaluation function at the leaf nodes are shown in the tree. Note that in this tree not all leaf nodes are at the same level. Use the minimax procedure to select your next move. Show your work on the tree below.



2. In the above tree use the minimax procedure along with alpha-beta pruning to select the next move. Mark the nodes that don't need to be evaluated.

3. Consider the game of tic-tac-toe. Assume that X is the MAX player. Let the utility of a win for X be 10, a loss for X be -10, and a draw be 0.

a) Given the game board **board1** below where it is X's turn to play next, show the entire game tree. Mark the utilities of each terminal state and use the minimax algorithm to calculate the optimal move.

b) Given the game board **board2** below where it is X's turn to play next, show the game tree with a cut-off depth of two ply (i.e., stop after each player makes one move). Use the following evaluation function on all leaf nodes:

$$Eval(s) = 10X_3(s) + 3X_2(s) + X_1(s) - (10O_3(s) + 3O_2(s) + O_1(s))$$

where we define $X_n(s)$ as the number of rows, columns, or diagonals in state *s* with exactly *n* X's and no O's, and similarly define $O_n(s)$ as the number of rows, columns, or diagonals in state *s* with exactly *n* O's and no X's. Use the minimax algorithm to determine X's best move.

board1

X	O	X
O		O
	X	

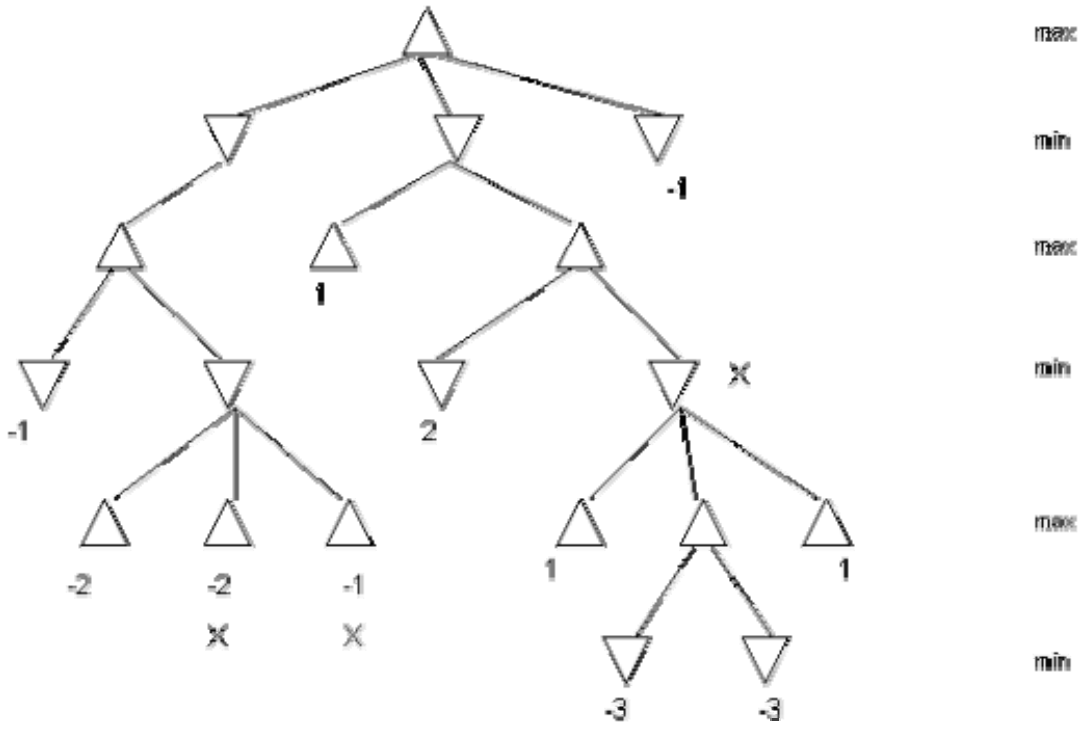
board2

	X	
O		
X		O

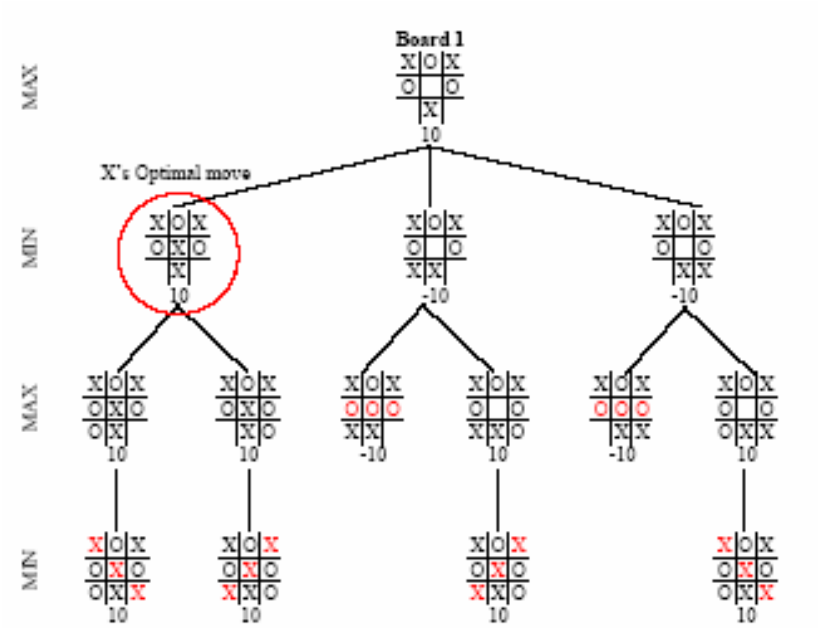
Solution

1. The second child of the root node as your next move since that child produces the highest gain.

2. Nodes marked with an "X" are not evaluated when alpha-beta pruning is used. None of the nodes in the subtree rooted at the node marked with an "X" in the third level of the tree are evaluated.



3.a. The game tree is shown below



3.b. The values are shown below