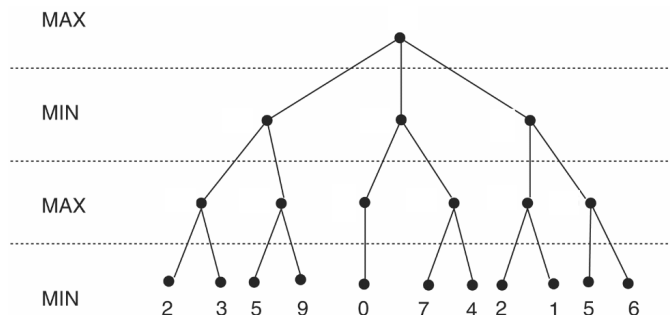


COMP 6721 Applied Artificial Intelligence (Fall 2021)

Worksheet #2: Adversarial Search

Game of Nim. Play a game of Nim against your team mate, starting with 7 tokens (write down the number of tokens at each move).

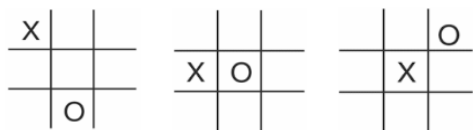
MiniMax. Let's apply the MiniMax algorithm discussed in the lecture on an example (fixed ply depth of 3):



Leaf nodes show the actual heuristic value $e(n)$

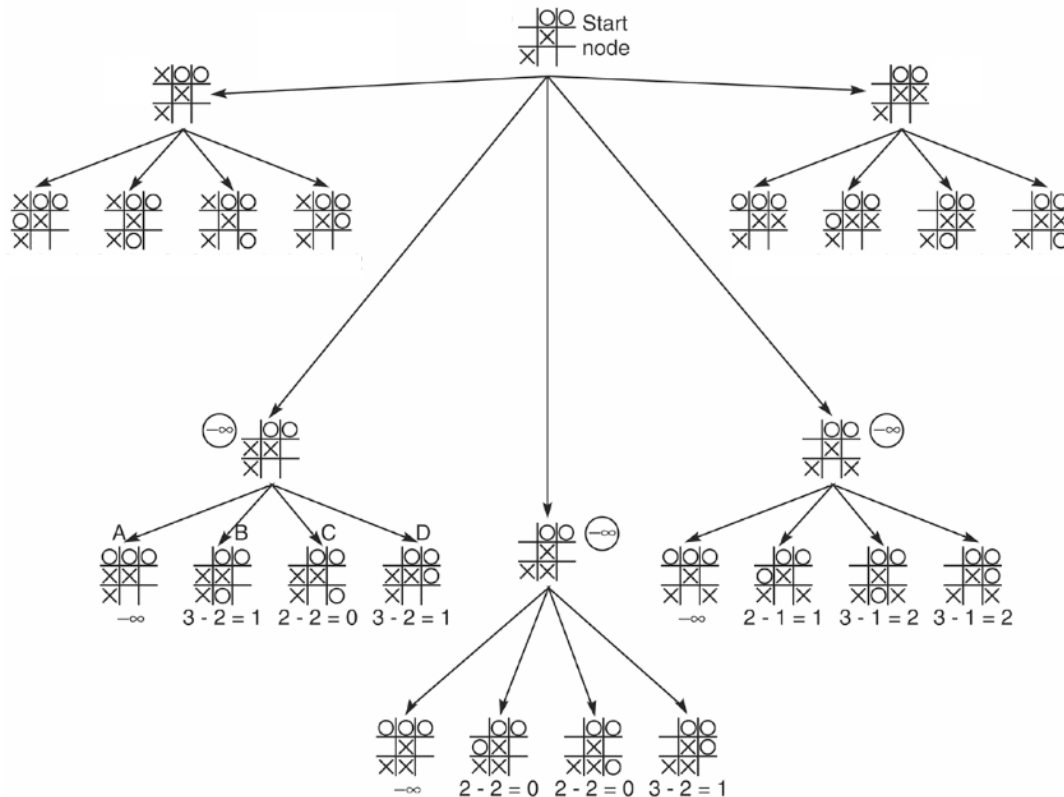
- Find the *back-up* heuristic values of all non-leaf (internal) nodes using the MiniMax procedure.
- Highlight the *best next move* for MAX (starting from the root node).

MiniMax Heuristic for Tic-Tac-Toe. Using the heuristic shown below, compute the values of $e(n)$ for the three game states (MAX plays X):



$$e(n) = \begin{cases} \text{number of rows, columns, and diagonals open for MAX} \\ \quad - \text{number of rows, columns, and diagonals open for MIN} \\ +\infty, \text{ if } n \text{ is a forced win for MAX} \\ -\infty, \text{ if } n \text{ is a forced win for MIN} \end{cases}$$

Two-ply MiniMax. Compute the missing values using MiniMax in the game tree shown below (same heuristic as above, start node is MAX). What will be MAX's next move?



Alpha-Beta Pruning. Apply the Alpha-Beta Pruning algorithm:

```

01 function alphabeta(node, depth,  $\alpha$ ,  $\beta$ , maximizingPlayer)
02   if depth = 0 or node is a terminal node
03     return the heuristic value of node
04   if maximizingPlayer
05     v := - $\infty$ 
06     for each child of node
07       v := max(v, alphabeta(child, depth - 1,  $\alpha$ ,  $\beta$ , FALSE))
08      $\alpha$  := max( $\alpha$ , v)
09     if  $\beta \leq \alpha$ 
10       break (*  $\beta$  cut-off *)
11   return v
12   else
13     v :=  $\infty$ 
14     for each child of node
15       v := min(v, alphabeta(child, depth - 1,  $\alpha$ ,  $\beta$ , TRUE))
16        $\beta$  := min( $\beta$ , v)
17       if  $\beta \leq \alpha$ 
18         break (*  $\alpha$  cut-off *)
19   return v

```

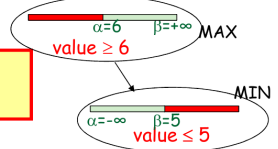
Initial call:
 alphabeta(origin, depth, - ∞ , + ∞ , TRUE)

- α : lower bound on the final backed-up value.
- β : upper bound on the final backed-up value.

■ Alpha pruning:

- eg. if MAX node's $\alpha = 6$, then the search can prune branches from a MIN descendant that has a $\beta \leq 6$.
- if child $\beta \leq$ ancestor $\alpha \rightarrow$ prune

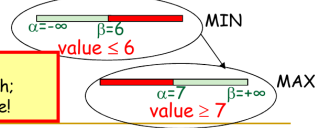
incompatible...
 so stop searching the right branch;
 the value cannot come from there!



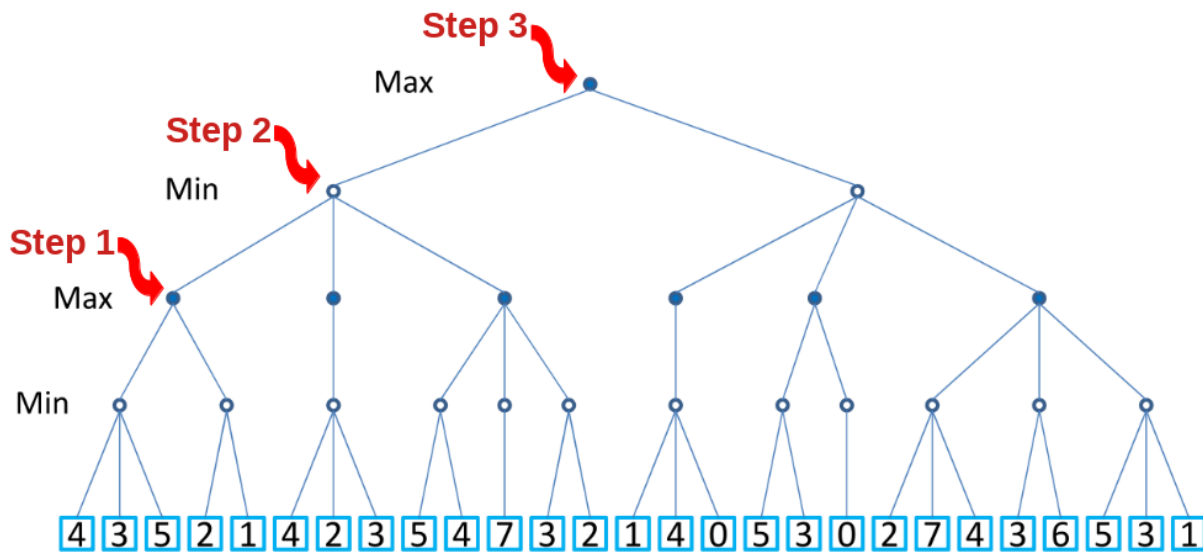
■ Beta pruning:

- eg. if a MIN node's $\beta = 6$, then the search can prune branches from a MAX descendant that has an $\alpha \geq 6$.
- if ancestor $\beta \leq$ child $\alpha \rightarrow$ prune

incompatible...
 so stop searching the right branch;
 the value cannot come from there!



on the following search tree:



Step 1: Perform the Alpha-Beta procedure (left-to-right) until you reached the node marked with “Step 1”. Circle each node that you explored and show which subtrees are cut off by the algorithm (if any).

Step 2: Now continue with the algorithm until you reached the node marked “Step 2”, marking explored nodes and cut subtrees as before.

Step 3: Complete the algorithm until you calculated the value for the root node in the same fashion.

How many nodes did the algorithm explore (out of 27 possible): ?