function ALPHA-BETA-SEARCH($state$) **returns** an action
  $v \leftarrow$ MAX-VALUE($state, -\infty, +\infty$)
  return the $action$ in ACTIONS($state$) with value v

---

function MAX-VALUE($state, \alpha, \beta$) **returns** a $utility$ $value$
  if TERMINAL-TEST($state$) **then return** UTILITY($state$)
  $v \leftarrow -\infty$
  for each a **in** ACTIONS($state$) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(RESULT(s,a), α, β))
    if $v \geq \beta$ **then return** v
    $\alpha \leftarrow$ MAX(α, v)
  return v

---

function MIN-VALUE($state, \alpha, \beta$) **returns** a $utility$ $value$
  if TERMINAL-TEST($state$) **then return** UTILITY($state$)
  $v \leftarrow +\infty$
  for each a **in** ACTIONS($state$) **do**
    $v \leftarrow$ MIN(v, MAX-VALUE(RESULT(s,a) , α, β))
    if $v \leq \alpha$ **then return** v
    $\beta \leftarrow$ MIN(β, v)
  return v

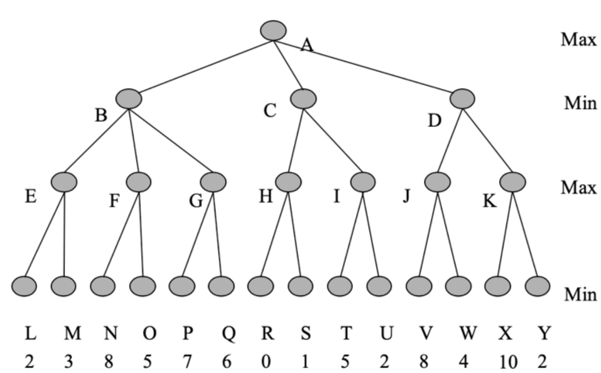
Exercise: Game Playing

Consider the following game tree in which the evaluation function values are shown below each leaf node. Assume that the root node corresponds to the maximizing player. Assume the search always visits children left-to-right.

(a) Compute the backed-up values computed by the minimax algorithm. Show your answer by writing values at the appropriate nodes in the above tree.

(b) Compute the backed-up values computed by the alpha-beta algorithm. What nodes will not be examined by the alpha-beta pruning algorithm?

(c) What move should Max choose once the values have been backed-up all the way?



Intuition (using inequalities):

(a) backed up values: Remember that root node is max (see text of the question). So value(A) is max(value(B), value(C), value(D)), and so on. We get:
E=3, F=8, G=7, H=1, I=5, J=8, K=10, then
B=3, C=1, D=8, and finally
A=8

(b) the backed-up values are the same, except that some will not be computed (they will be pruned). Here, E is computed, with E=3. Then, moving to F, we evaluate N=8 and then realize F >= 8, so it will not be selected by Min at B, who will instead select E=3. Hence we do not need to evaluate O (O is pruned). Likewise with G: we evaluate P=7 and realize that G >= 7 will not be selected by Min at B, since E=3 will be instead. Hence Q is pruned. Then we also compute H=1. This tells us C <= 1 since C is a min node. Because B=3, A >= 3 since A is a max node. Since C <= 1, it will not be chosen since we can make 3 by choosing B; hence, we do not need to compute I (so you would cross out I in your answer, and also possibly T and U if you want). We then compute J=8 and thus we know D <= 8. This is potentially still better than 3, so we do need to look at K. We evaluate X=10 hence K >= 10, so we can safely prune Y since Min at D will not select K >= 10 given that we also know that J=8. We end up with D=8, and finally A=8.

So, nodes O, Q, I (and children T, U), and Y were pruned in this example.

(c) At the root, max will choose the move that goes to state D since this guarantees 8 (or more if the opponent does not deploy perfect play), which is higher than if we chose B (worth 3) or C (worth 1 or less).

Full alpha-beta run:

Below we do a detailed run of alpha-beta on this example.

Call AlphaBetaSearch(A)

Starts with $\alpha = -\infty$ and $\beta = +\infty$

MaxValue(A, $\alpha = -\infty$, $\beta = +\infty$)

v = $-\infty$

Loop over B, C, D:

Start with B:

MinValue(B, $\alpha = -\infty$, $\beta = +\infty$)

v = $+\infty$ **Note: v is a local variable (not same here as the other v above)**

Loop over E, F, G:

Start with E:

MaxValue(E, $\alpha = -\infty$, $\beta = +\infty$)

v = $-\infty$

Loop over L, M:

Start with L:

MinValue(L, $\alpha = -\infty$, $\beta = +\infty$)

L is terminal; return 2

Done with L.

v = max(v=$-\infty$, 2 from L) = 2 **Value so far is 2**

v $\geq \beta$ fails **No pruning**

α = max(α=$-\infty$, v) = 2 **Update α=2 best so far**

Start with M:

MinValue(M, $\alpha = 2$, $\beta = +\infty$)

M is terminal; return 3

Done with M.

v = max(v=2, 3 from M) = 3 **Value so far is 3**

v $\geq \beta$ fails **No pruning**

α = max(α=2, v) = 3 **Update α=3 best so far**

Done with loop over L, M.

return v = 3

Done with E.

v = min(v=$+\infty$, 3 from E) = 3

v $\leq \alpha$ fails

β = min(β=$+\infty$, v) = 3

Start with F:

MaxValue(F, α = -∞, β = 3)

v = -∞

Loop over N, O:

Start with N:

MinValue(N, α = -∞, β = 3)

N is terminal; return 8

Done with N.

v = max(v=-∞, 8 from N) = 8 **Value so far is 8**

v $\geq$ β passes! **End loop, prune O**

Done with loop over N, O.

return v = 8

Done with F.

v = min(v=3, 8 from F) = 3

v $\leq$ α fails

β = min(β=3, v) = 3

Start with G:

MaxValue(G, α = -∞, β = 3)

v = -∞

Loop over P, Q:

Start with P:

MinValue(P, α = -∞, β = 3)

P is terminal; return 7

Done with P.

v = max(v=-∞, 7 from P) = 7 **Value so far is 7**

v $\geq$ β passes! **End loop, prune Q**

Done with loop over P, Q.

return v = 7

Done with G.

v = min(v=3, 7 from G) = 3

v $\leq$ α fails

β = min(β=3, v) = 3

Done with loop over E, F, G.

return v = 3

Done with B.

$v = \max(v=-\infty, 3 \text{ from } B) = 3$ **Value so far is 3**
$v \geq \beta$ fails **No pruning**
$\alpha = \max(\alpha=-\infty, v) = 3$ **Update α=3 best so far**

Start with C:
MinValue(C, $\alpha = 3$, $\beta = +\infty$)
    $v = +\infty$
    Loop over H, I:

    Start with H:
    MaxValue(H, $\alpha = 3$, $\beta = +\infty$)
        $v = -\infty$
        Loop over R, S:

        Start with R:
            MinValue(R, $\alpha = 3$, $\beta = +\infty$)
            R is terminal; return 0
        Done with R.
        $v = \max(v=-\infty, 0 \text{ from } R) = 0$ **Value so far is 0**
        $v \geq \beta$ fails **No pruning**
        $\alpha = \max(\alpha=3, v) = 3$ **α=3 still best so far**

        Start with S:
            MinValue(S, $\alpha = 3$, $\beta = +\infty$)
            S is terminal; return 1
        Done with S.
        $v = \max(v=0, 1 \text{ from } S) = 1$ **Value so far is 1**
        $v \geq \beta$ fails **No pruning**
        $\alpha = \max(\alpha=3, v) = 3$ **α=3 still best so far**

        Done with loop over R, S.
        return $v = 1$
    Done with H.

    $v = \min(v=+\infty, 1 \text{ from } H) = 1$
    $v \leq \alpha$ passes! **End loop, prune I (and T, U)**
    Done with loop over H, I.
    return $v = 1$
Done with C.

$v = \max(v=3, 1 \text{ from } C) = 3$ **Value so far is still 3**
$v \geq \beta$ fails **No pruning**
$\alpha = \max(\alpha=3, v) = 3$ **α=3 still best so far**

Start with D:
MinValue(D, α = 3, β = $+\infty$)

v = $+\infty$
Loop over J, K:

Start with J:
MaxValue(J, α = 3, β = $+\infty$)

v = $-\infty$
Loop over V, W:

Start with V:

MinValue(V, α = 3, β = $+\infty$)
V is terminal; return 8

Done with V.

v = max(v=$-\infty$, 8 from V) = 8	**Value so far is 8**
v $\geq$ β fails	**No pruning**
α = max(α=3, v) = 8	**Update α=8 best so far**

Start with W:

MinValue(W, α = 8, β = $+\infty$)
W is terminal; return 4

Done with W.

v = max(v=8, 4 from W) = 8	**Value so far is 8**
v $\geq$ β fails	**No pruning**
α = max(α=8, v) = 8	**α=8 still best so far**

Done with loop over V, W.
return v = 8

Done with J.

v = min(v=$+\infty$, 8 from J) = 8
v $\leq$ α fails
β = min(β=$+\infty$, v) = 8

Start with K:
MaxValue(F, α = 3, β = 8)
v = -∞
Loop over X, Y:

Start with X:
MinValue(X, α = 3, β = 8)
X is terminal; return 10
Done with N.
v = max(v=-∞, 10 from N) = 10 **Value so far is 10**
v ≥ β passes! **End loop, prune Y**

Done with loop over X, Y.
return v = 10
Done with K.

v = min(v=8, 10 from K) = 8
v ≤ α fails
β = min(β=8, v) = 8

Done with loop over J, K.
return v = 8
Done with D.

v = max(v=3, 8 from D) = 8 **Value so far is 8**
v ≥ β fails **No pruning**
α = max(α=3, v) = 8 **Update α=8 best so far**
Done with loop over B, C, D
return v = 8

v=8 (max value over B, C, D)

return an action with value 8 (i.e., select D)