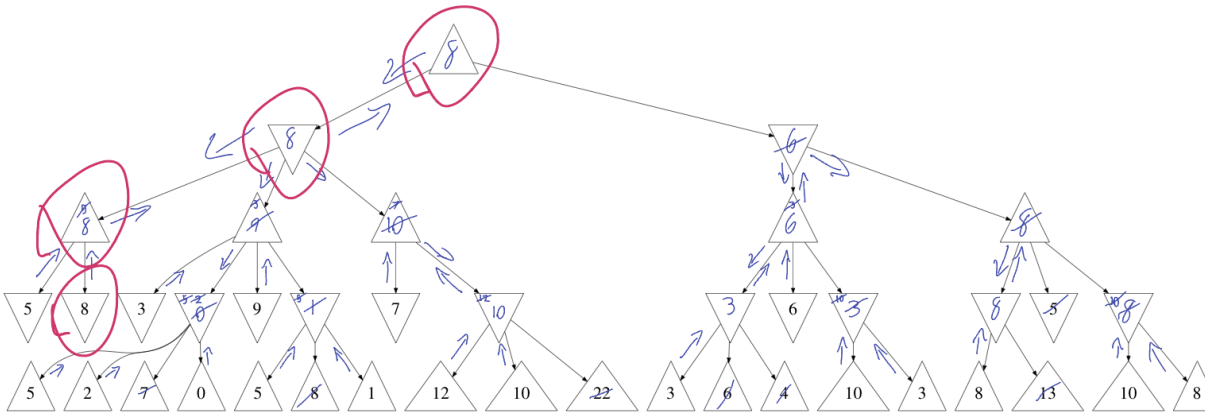
1 Minmax Search

Problem 1 (written: 30 pts): Using the following figure 1, use minmax search to assign utility values for each internal node (i.e., non-leaf node) and indicate which path is the optimal solution for the MAX node at the root of the tree. Assume you explore the successors from left to right.

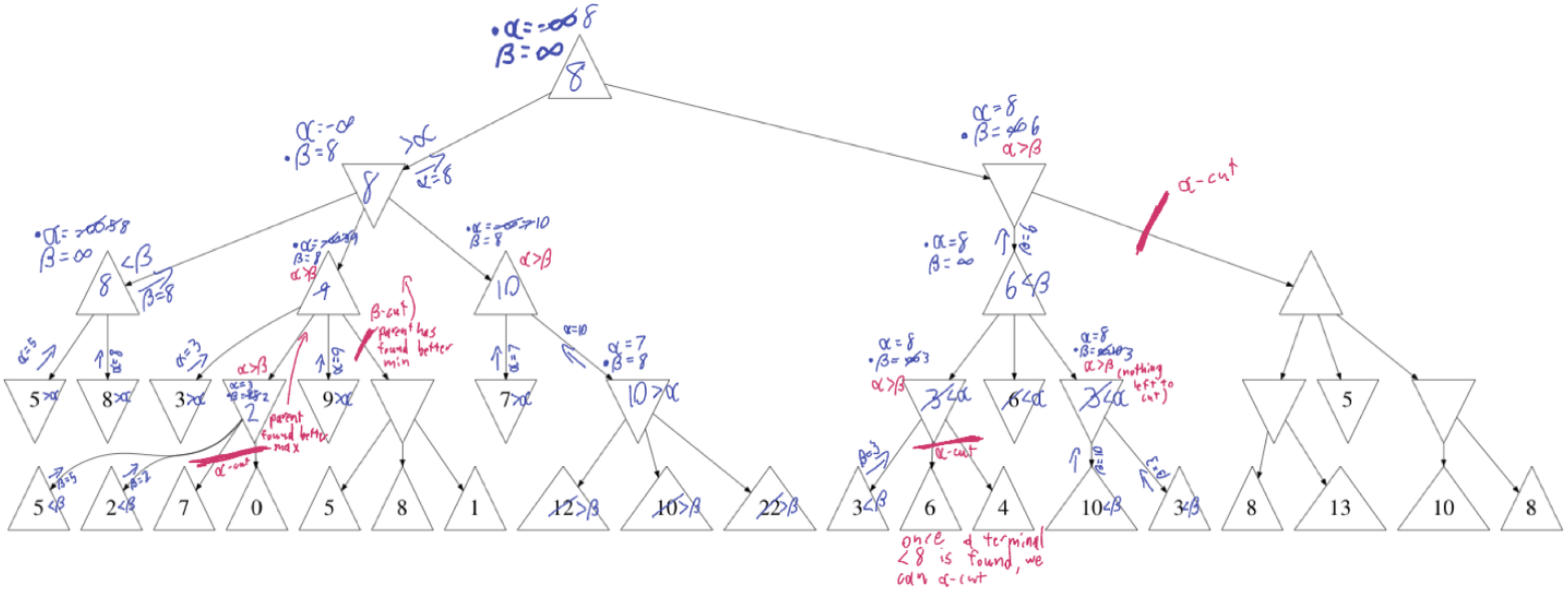


[0, 0, 1]
left,
min will choose left,
right,
where min leaf is 8

2 $\alpha - \beta$ pruning

Problem 2 (written: 30 pts): Using the following figure 2, use $\alpha - \beta$ pruning to (1) assign utility values for each internal node (i.e., non-leaf node) and indicate which path is the optimal solution for the MAX node at the root of the tree. (2) For each node, indicate the final α and β values. (Note that initial values at the root are $\alpha = -\infty, \beta = \infty$.) (3) For each cut that happens, draw a line to cross out that subtree.

Hint: There are 4 places that need to be cut.



Problem 4 (written: 10 pts): In Minmax search, we used a depth-first exploration through the use of recursion. We know that Minmax gives an optimal solution, however, we also know that depth-first search is suboptimal. Explain why Minmax gives an optimal solution even when it is using a depth-first exploration.

DFS is suboptimal when the cost is depth-based. "using" DFS just means node visitation is done with a FIFO queue. MinMax does a complete search and returns the optimal path for the root node to obtain the highest reward possible when the lowest reward choice is chosen on alternating depths. (Like two players competing on back and forth turns)

Even though α-β pruning means no longer doing a complete search, it uses logic to prove that cut nodes are suboptimal paths.

function Max-Value (state, game, α, β) **return** utility value α: best MAX on path to *state* ; β: best MIN on path to *state* **if** Cutoff(state) **then return** Utility(state) $v \leftarrow -\infty$ **for each** s in Successor(state) **do** · $v \leftarrow$ Max(v, Min-Value(s,game,α,β)) · **if** $v \geq \beta$ **then return** v /* β **CUT!!** */ · $\alpha \leftarrow$Max(α, v) **end** **return** v	**function** Min-Value (state, game, α, β) **return** utility value α: best MAX on path to *state* ; β: best MIN on path to *state* **if** Cutoff(state) **then return** Utility (state) $v \leftarrow \infty$ **for each** s in Successor(state) **do** · $v \leftarrow$ Min(v, Max-Value(s,game,α,β)) · **if** $v \leq \alpha$ **then return** v /* α **CUT!!** */ · $\beta \leftarrow$Min(β,v) **end** **return** v