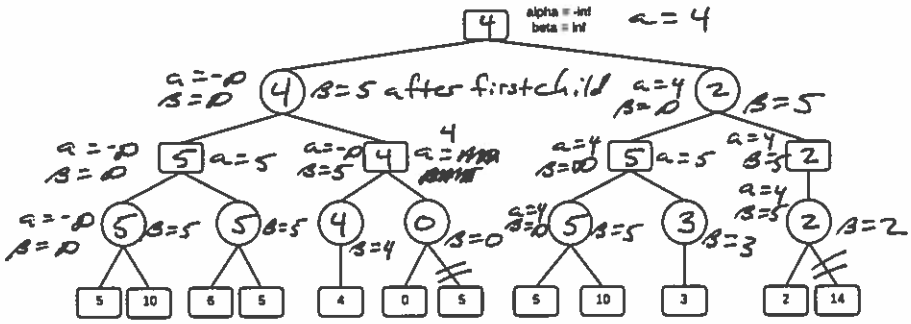
CSCI 3202 - Introduction to Artificial Intelligence
Instructor: Hoenigman
Assignment 4
Due Friday, September 23 by 4pm.
Submit written answers at the beginning of class

Problems

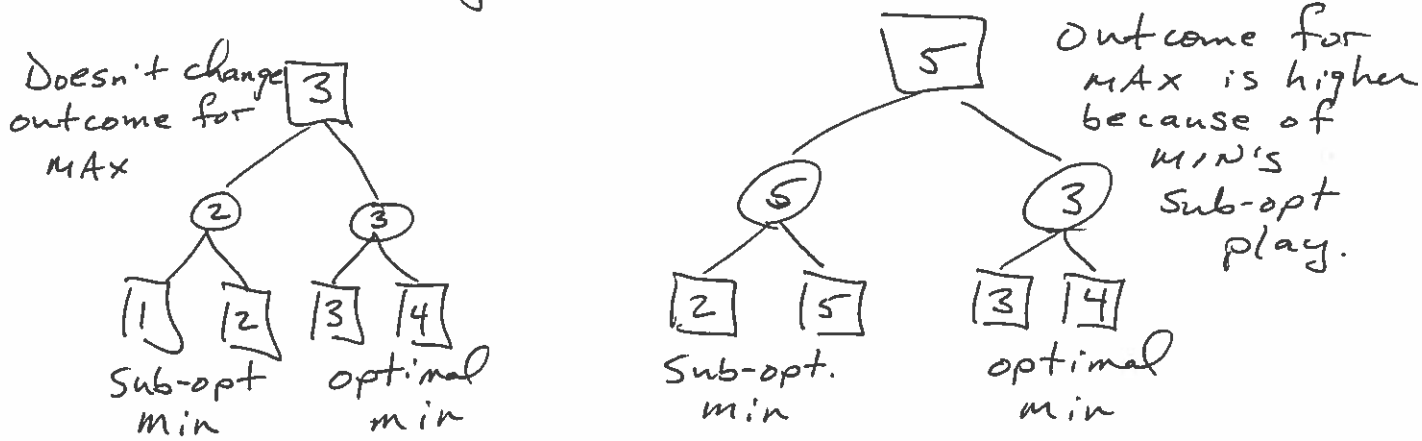
1. Use the alpha-beta algorithm to show the minimax value to MAX in the following game tree. Show the branches that are pruned during the search, and the values for MAX and MIN at each level in the tree.



2. Show that the following assertion is true using an example: For every game tree, the utility obtained by MAX using minimax decisions against a suboptimal MIN will never be lower than the utility obtained playing against an optimal MIN. Your example should also include an explanation of what an optimal utility is for MAX and MIN and what it would mean for either MAX or MIN to play a sub-optimal game.

   Answer: A suboptimal MIN means that MIN doesn't select the minimum value from its children and a suboptimal MAX means that MAX doesn't select the maximum value from its children. If MIN selects suboptimal value, then the choice for MAX, as the parent of MIN, will be at least as big as if MIN were playing optimally. The same is true for MIN as the parent of MAX, where if MAX selects a suboptimal value, then MIN will have a choice that is less than or equal to an optimal value.

Example showing sub-optimal MIN.

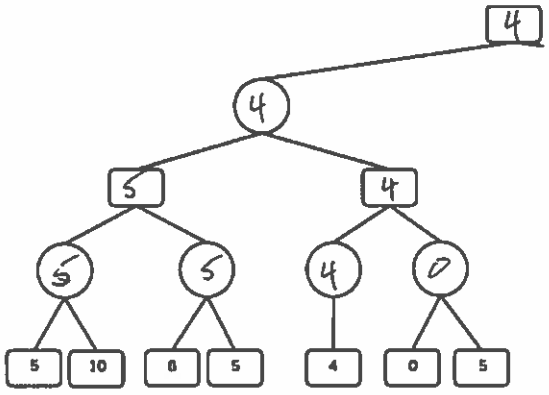


3. The following pseudo-code for the minimax algorithm is similar to the code included in the lecture notes. This code shows additional print commands that are not in the lecture notes code.

```
minimax(node, depth, maxPlayer)
    if depth == 0 or terminal(node) //terminal test is true
        return f(node)  //evaluation of the node
    if maxPlayer //Player(s) = MAX
        bestValue = -MAX_INT //system property, maximum negative integer
        for each child in node.adjacent
            eval = minimax(child, depth - 1, FALSE)
            print eval
            bestValue = max(bestValue, eval)
        return bestValue
    else //Player(s) = MIN
        bestValue = MAX_INT
        for each child in node.adjacent
            eval = minimax(child, depth - 1, TRUE)
            print eval
            bestValue = min(bestValue, eval)
        return bestValue

minimax(origin, depth, TRUE) //call from root for MAX player
```

For the following tree, what numbers are displayed when the algorithm executes with an initial depth = 4. You can assume that the root of the tree only has one child.



5, 10, 5, 6, 5, 5, 5, 4, 4, 0, 5, 0,

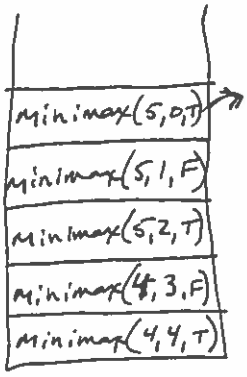
4, 4

Doesn't print the root.

Initial call to minimax passes in the root.

Call minimax recursively until bottom of tree reached.

Call stack until first popped node



this node pops first and prints, Then push right child and then pop.

push root