# IN3130 Exercise set 11b

## Exercise 1

Study figures 23.13 and 23.14 (pages 735 and 736 in the textbook) where -1 and +1 is used to indicate win and loss, respectively. Look at all nodes and make sure you understand how values for the internal nodes are calculated with the min/max-algorithm. Always keep in mind that values indicate the situation for the player with the opening move – player A. For B smaller values are better. (Note also that in exercise 3 below we negate the values on every level so that we can always maximize!).

## Exercise 2

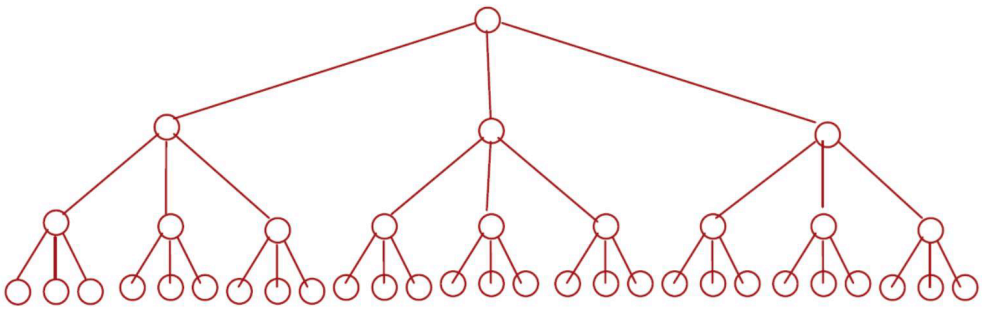
Study figures 23.16 and 23.17 (pages 738 and 740 in the textbook) and check that your understanding of alphabeta-pruning is correct; then solve exercise 23.22 in the textbook.

## Exercise 3

Go through the program on page 741 and discuss the solution chosen there, where values are negated on every level. Note that there are some misprints in the program. First, in both lines following an "else" the name of the called procedure should be "ABNodeValue". Second, a right parenthesis is missing at the end of the last of these lines. Finally, it should be "-∞" in the outermost call. See slides!

## Exercise 4

If we, in each situation in an alpha-beta-search, are lucky enough to always look at the best move (for the player to make the move) first we will get good pruning. One can even prove that if we go down to depth *d*, with branching factor of *b*, the search time with alpha/beta-pruning will be $O\left(\underbrace{b \cdot 1 \cdot b \cdot 1 \cdot b \ \ldots}_{d \text{ factors}}\right) = b^{\frac{d}{2}}$ instead of $O\left(\underbrace{b \cdot b \cdot b \ \ldots}_{d \text{ factors}}\right) = b^{d}$. This means that if you without pruning could reach to depth n in a certain time period, you can in the same period reach to about depth 2**n* using alpha/beta pruning. We shall not attempt to prove this, but instead look at a concrete example. We let *d* = 3, and *b* = 3, and get the tree below. Assume that the best move is always the one drawn to the left in the figure below and that we look at the subtrees from left to right for each node. Mark the branches you will have to evaluate (and thereby the ones you can avoid). The tree has 39 edges, how many do you avoid looking at?



**EXTRA question:** Assume you are less lucky than above, and always look at the best move last. Will you then get any pruning at all?

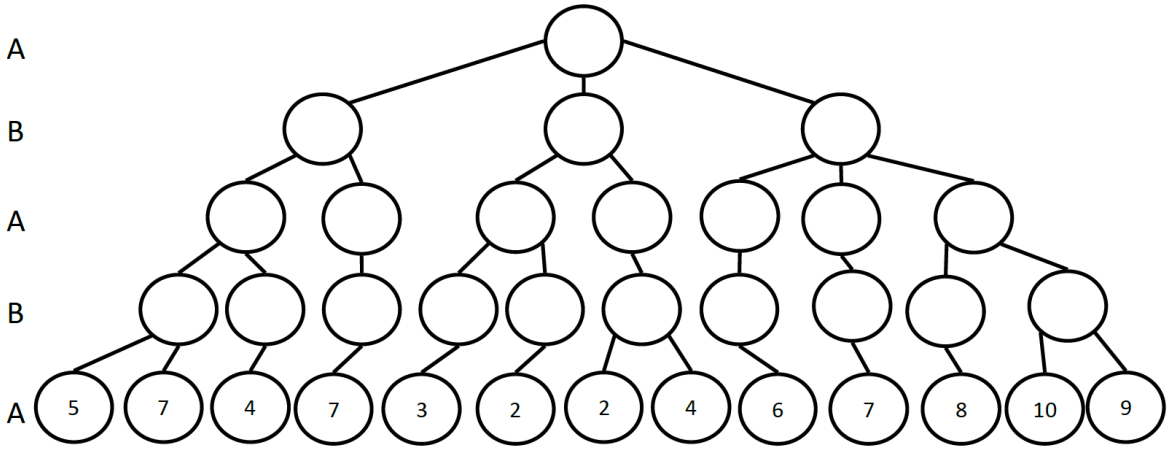
## Exercise 5

(Can be skipped, as it is not central to the course) Assume you are playing the game of NIM, with two piles, and that it is your move, that no pile is empty, and that the piles are of different sizes (number of sticks or pebbles, or whatever). Try to come up with a strategy that guarantees victory.

## Exercise 2 (From Exam 2010)

We shall look at game trees, and we assume that the root node of the tree in the figure below is representing the current situation of a game (that we do not describe further), and that it is player A's turn to move. The other player is B, and A and B alternately make moves. Player A wants to maximize the values of the nodes while B wants to minimize them.

Player A shall make considerations for deciding which move to make from the root situation, and the tree in the figure below shows all situations it is possible to reach with at most four moves from the current situation. A has a heuristic function (that is, a function that for a given situation gives an integer) that he uses to evaluate how god the situation is for him. A uses this function for situations where he terminates the search towards deeper nodes. For each terminal node in the tree below this function is evaluated and the value appears in the nodes.




### 6.a

Using the heuristic values in the terminal nodes, indicate how good the situation is for A in each of the other nodes. What is the best value player A can achieve regardless of how well B plays. Don't use any alpha-beta pruning.

### 6.b

We now assume that we are back to the start-situation, no nodes have values. We start the algorithm again, and A will then make a depth-first search in the tree from the root node, down to the depth of the tree above. In each terminal node A computes the value of the heuristic function (and thereby gets the value given in the corresponding terminal node in the figure above). The search is done from left to right in the figure above. Indicate which alpha- and beta-cutoffs you will get during this search. In the drawing you should simply write an 'X' at the root nodes of the sub trees that will not visit because of alpha- or beta-cutoffs. Give a short but explicit explanation for each cutoff (and for this you may suitably give names to some of the nodes in the figure).

[end]