COMP3308/3608 Artificial Intelligence

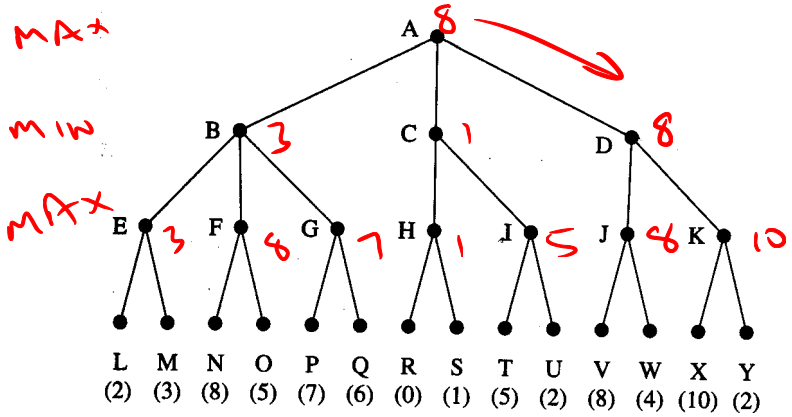
Week 4 Tutorial exercises
Game Playing

***Exercise 1*. Adversarial search**

Why does search in game playing programs always proceed forward from the current position rather than backward from the goal?

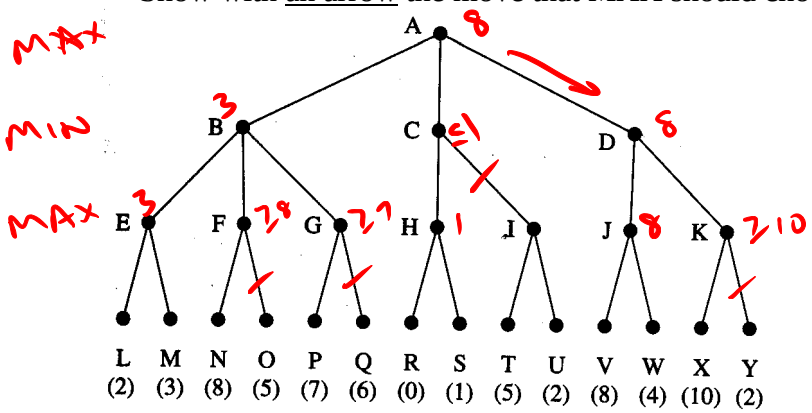
***Exercise 2 (Homework)*. Minimax and alpha-beta**

Consider the following game in which the evaluation function values for the MAX player are shown at the leaf nodes. MAX wants to maximize the evaluation function, while its opponent MIN wants to minimize the same evaluation function. The first player is MAX, and hence the root node corresponds to MAX.



a) In the figure above, fill in the backed-up values that are computed by the minimax algorithm for all the nodes. Show with an arrow the move that MAX should choose.

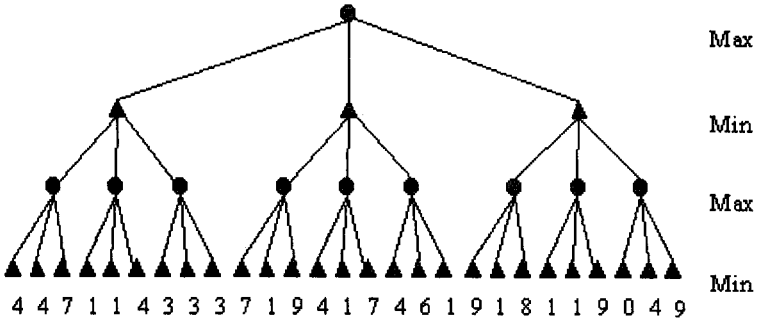
b) Assume that we now use the alpha-beta algorithm and that the children are visited left-to-right (i.e. in normal order). In the following figure, show all intermediate bounding values at each node as they get updated and cross all the branches that would be pruned. (Make sure you cross out branches, not nodes!) Show with an arrow the move that MAX should choose.



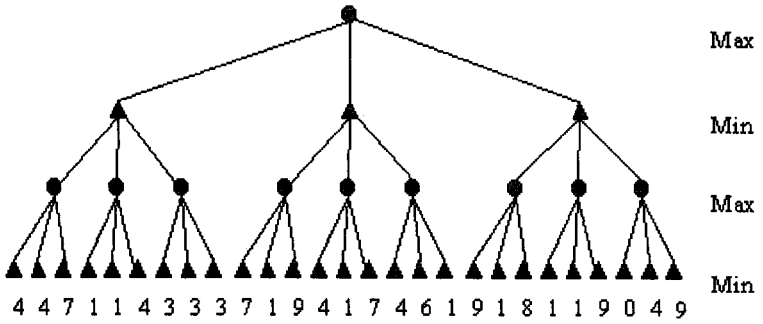
***Exercise 3*. Alpha-beta algorithm**

Consider the following game tree where MAX is the first player. The evaluation values from the first player's point of view are shown below the leaves.

a) Assume that the nodes are examined in normal order, i.e. left-to-right. Compute the final backed-up values using the alpha-beta algorithm and show your answer by writing the intermediate and final values at the nodes in the tree. Show with an arrow the move the first player should choose.



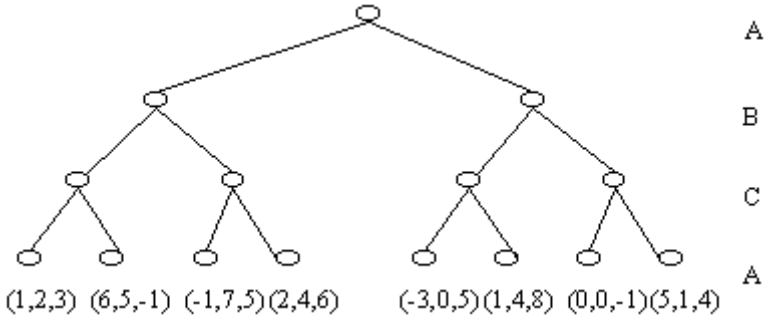
b) The same question as a) but now assume that the nodes are examined in right-to-left order.



***Exercise 4*. Minimax applied to 3-player games**

Consider a 3-player, perfect information game with 3 players, A, B, C. Based on the information specific to each player, we have 3 specific evaluation functions, f_A, f_B, f_C associated with each player respectively. These functions indicate the estimated value of a board position with respect to that player. For example $f_A(n) = -5$ means that position *n* is not good for player A, whereas $f_A(n)=100$ means that position *n* looks very good for player A.

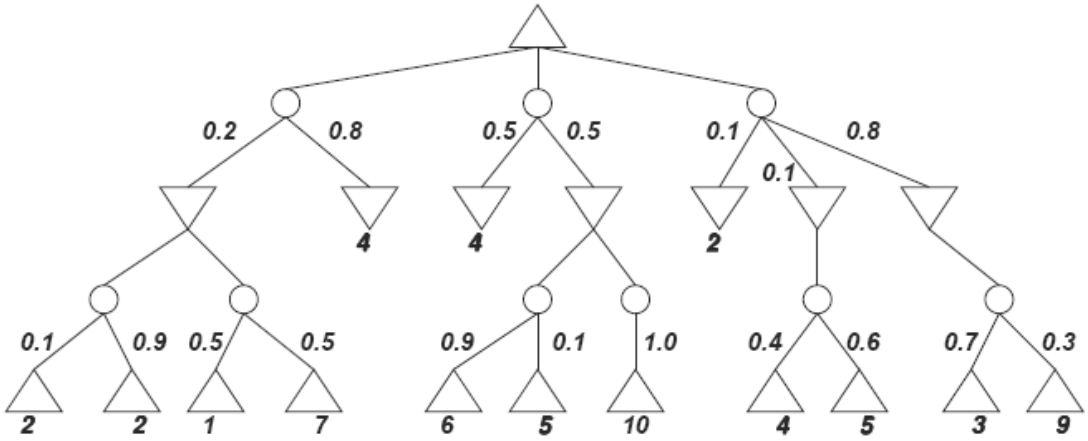
Consider the following game tree, where each triplet at the leaf nodes represents (f_A, f_B, f_C). Backup the values for each player's turn using minimax and show the optimal move for player A.



***Exercise 5*. Expectiminimax for games that include an element of chance**

Given is the following game tree. The utilities of the terminal nodes are indicated below the leaf nodes and the probabilities of chance nodes (the round nodes) are next to the corresponding branch. The root is a max node.

Determine the values of all tree nodes using the expectiminimax algorithm. Which node should MAX choose?



Exercise 6. Minimax (Advanced only)

Suppose that MAX plays against suboptimal MIN. Can you come up with a game tree in which MAX can do better using a suboptimal strategy than using minimax? If so, what more do we need to know about MIN?

Exercise 7. Minimax (Advanced only)

Prove the following: 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.

Additional exercises (to be done at your own time)

Exercise 8. **Evaluation functions, minimax and alpha-beta pruning**

Consider playing tic-tac-toe. We define X_n as the number of rows, columns or diagonals with exactly *n* crosses and no circles. Similarly, O_n is the number of rows, columns, or diagonals with just *n* circles and no crosses. The utility function assigns +1 to any position with $X_3=1$ and -1 to any position with $O_3=1$. All other terminal positions have utility 0. For non-terminal positions we use a linear evaluation function defined as $Eval(s)=3X_2(s)+X_1(s)-(3O_2(s)+O_1(s))$.

a) Show the whole game tree starting from an empty board down to depth 2 (i.e. one X and one O on the board), taking symmetry into account.

b) Mark on your tree the evaluations of all the positions at depth 2.

c) Using the minimax algorithm, mark on your tree the backed-up values for the positions at depths 1 and 0, and use those to choose the best starting move.

d) Circle the nodes at level 2 that would not be evaluated if alpha-beta pruning were applied, assuming the nodes are generated in the optimal order for alpha-beta pruning.