# COMP3308/COMP3608 Introduction to Artificial Intelligence

## Week 4 Tutorial exercises
## Game Playing

***Exercise 1*. Adversarial search**

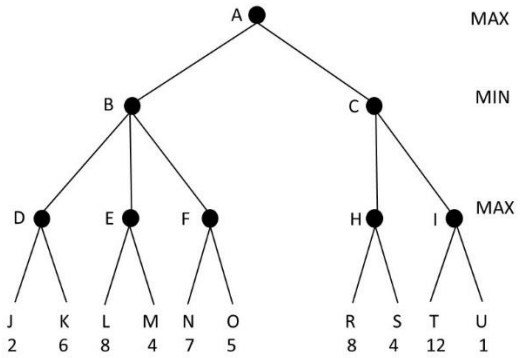
Why does search in game playing programs always proceed forward from the current position rather than backward from the goal?

***Answer:***

1) If we proceed backwards, we would not know where to start from as typically there are many goal states, e.g. there are many checkmate positions in chess.
2) If the goal state is too far from the current state for the search to terminate, the backward search would not give us any useful information, whereas a forward incomplete search will give us a good approximation.

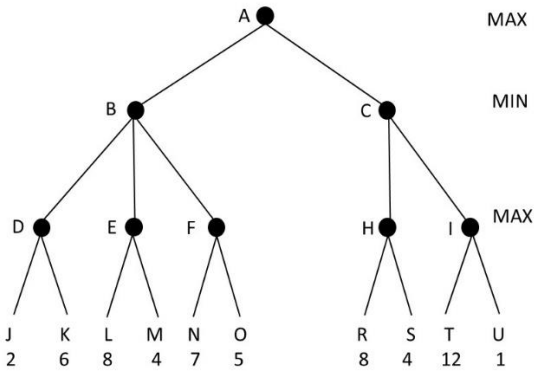
***Exercise 2 (Homework)*. Minimax and alpha-beta**

Consider the following game tree. The evaluation function values for player MAX are shown at the leaf nodes. The opponent MIN wants the same evaluation function to be minimized. The first player is MAX.




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

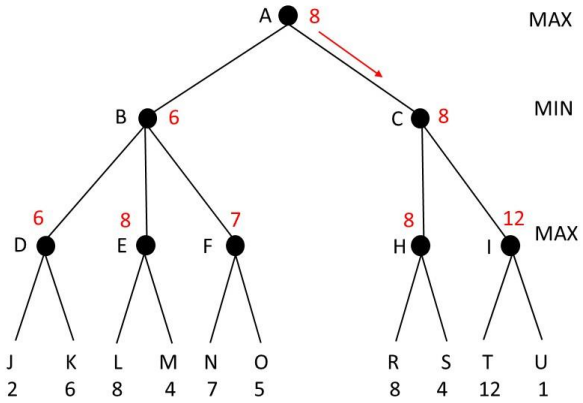
b) We now use the **alpha-beta algorithm**. In the figure below, show all intermediate bounding values at each node as they get updated and cross all the branches that would be pruned, e.g. AB etc. Assume that children are visited left-to-right (as usual). Show with an arrow the move that MAX should choose.



***Solution:***

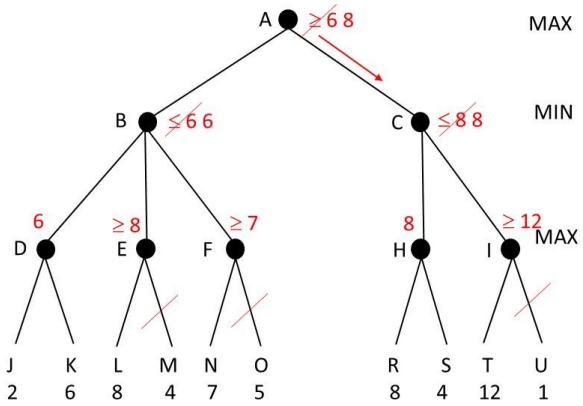
**a) Minimax**

The backed-up values are shown in red. Max should choose C.




**b) Alpha-beta**

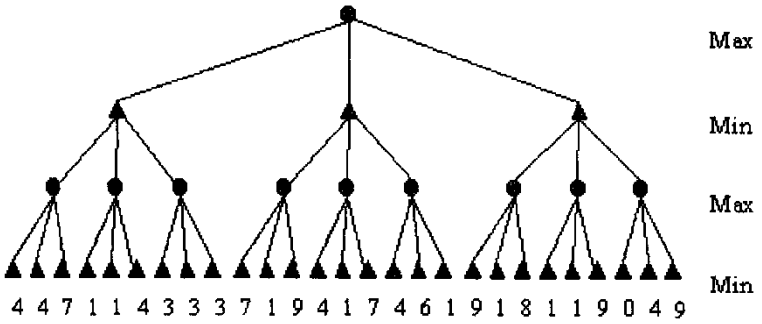
Three branches are pruned as shown. Max should choose C – the same move as minimax.



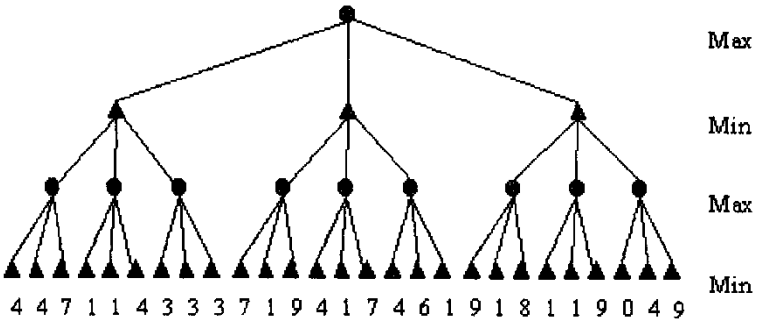

***Exercise 3*. Alpha-beta algorithm**

Consider the following game tree where MAX is the first player. The evaluation values for MAX are shown at the leaf nodes.

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. Cross the pruned branches. Show with an arrow the move that MAX should choose.

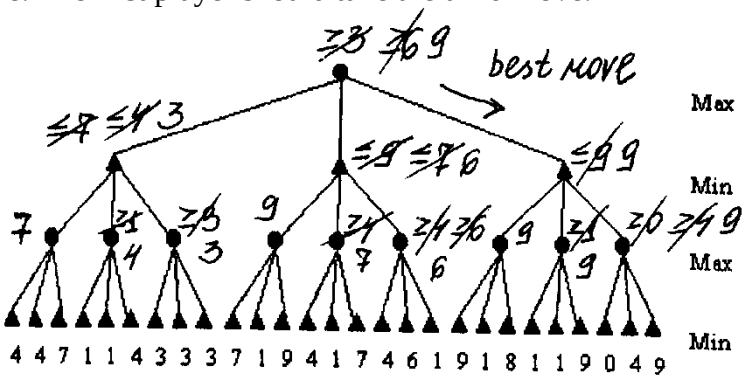



b) The same question as a) but now assume that the nodes are examined in right-to-left order.

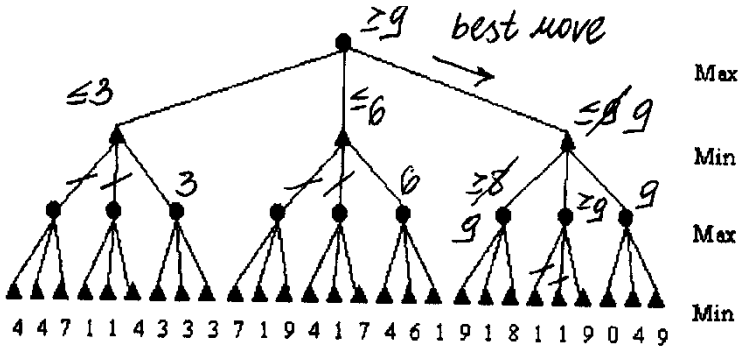



***Solution:***

a) No pruning is possible. The first player should take the third move.



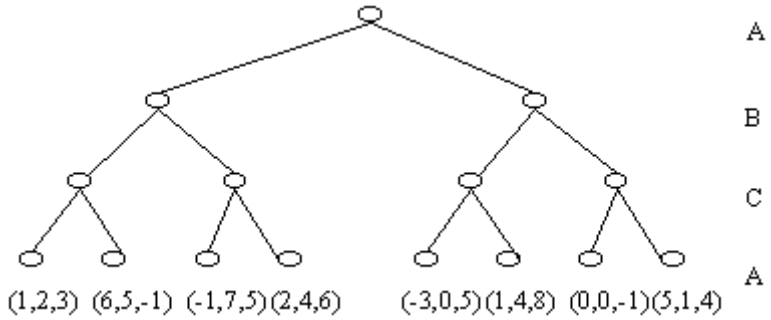

b) Pruning is possible. Again, the third move is the best one.



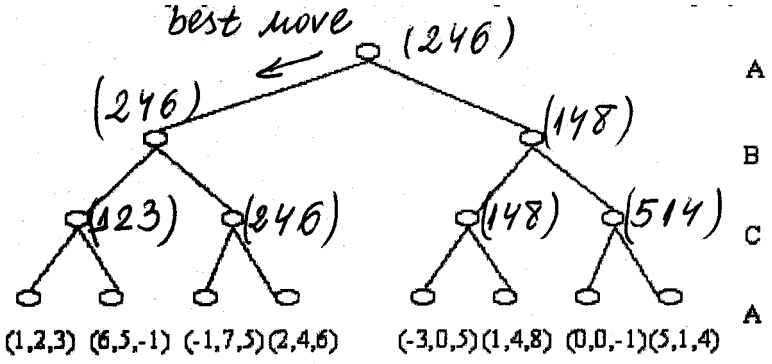
***Exercise 4*. Minimax applied to 3-player games**

Consider a 3-player, perfect information game with 3 players, A, B, C. Based on information specific to each player, we have 3 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* is 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.




***Solution:***

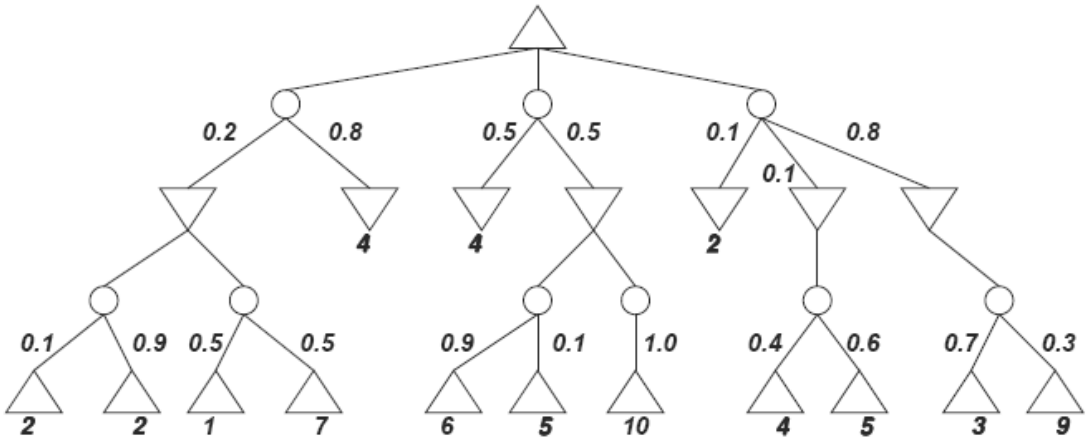



All players are maximizing. We start from level C and work towards the root. At level C, the player C tries to maximize his evaluation function (i.e. the third element of the triplet). At level B, player B tries to maximize his evaluation function (i.e. the second element in the triplet). Similarly, player A tries to maximize the first element of the triplet. The backed-up value of a node *n* is the evaluation function of whichever child has the highest evaluation function for the player choosing at *n*. Note that there is no mixing of values, the whole triplet is backed up.

***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.

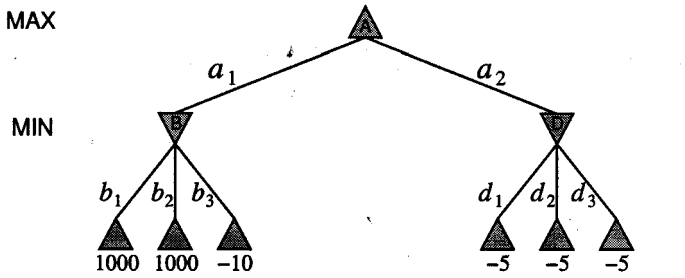
Determine the values of all tree nodes using the expectiminimax algorithm. Which node should MAX choose?




***Solution:***
The value of each chance node is the weighted sum of its children nodes. For instance, the value of the left most bottom node is 0.1*2+0.9*2=2. The value of the other nodes is determined as in the standard minimax: max of children values for MAX nodes or min of children values for MIN nodes.

***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?

***Answer:***
If the suboptimal play by MAX is predictable, then MAX can set traps and do better than using minimax. For example, consider the tree below.




According to minimax MAX should choose a2: by doing this MAX will achieve utility -5 which is better than -10 if MAX follows a1 and MIN plays optimally. However, if MAX knows that MIN falls for certain traps (i.e. we can predict the sub-optimality of MIN), for example we know that in cases like B (the left node in the tree) MIN tends to play b1 or b2, then A can play a1 (which is not the optimal move for him) and set a trap for MIN. This will be a win for MAX if MIN falls in the trap (i.e. plays b1 or b2, utility=1000 for MAX) but may also be a disaster for MAX if MIN did not fall in the trap (i.e. play b3, utility for MAX=-10).

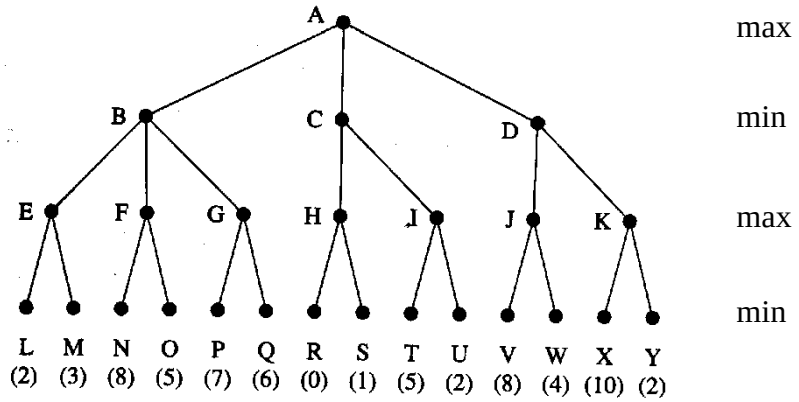
***Exercise 7. Minimax (Advanced only)***
Prove the following: for every game tree, the utility obtained by MAX using minimax decisions against a suboptimal MIN will never be lower than the utility obtained playing against an optimal MIN.

***Answer:*** Consider a MIN node whose children are terminal nodes. If MIN plays sub-optimally, then the value of the node is greater than the value it would have if MIN played optimally. This reasoning can be extended to the root.

## Additional exercises (to be done at your own time)

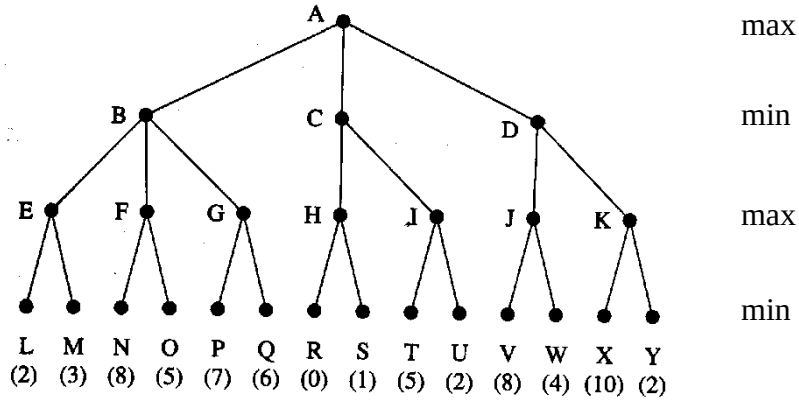
***Exercise 8.*** **Minimax and alpha-beta**

Consider the following game tree. The evaluation function values for player MAX are shown at the leaf nodes. The opponent MIN wants the same evaluation function to be minimized. The first player is MAX.



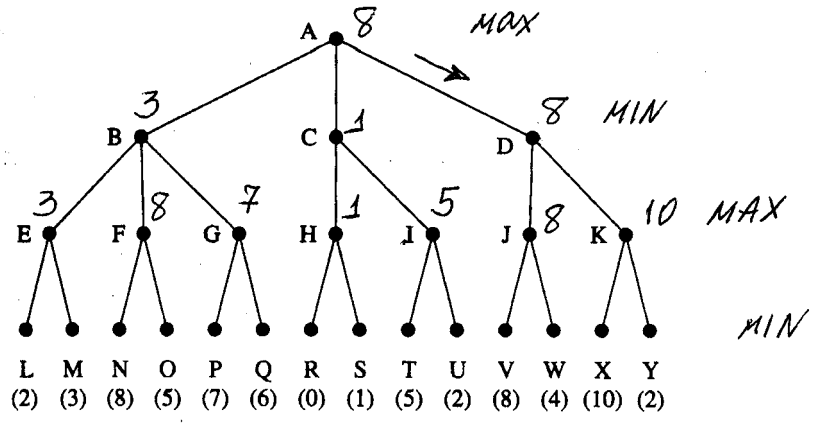

a) In the figure above, write the backed-up values that are computed by the minimax algorithm for all nodes. Show with an arrow the move that MAX should choose.

b) We now use the alpha-beta algorithm. In the figure below, show all intermediate bounding values at each node as they get updated and cross all the branches that would be pruned, e.g. AB etc. Assume that children are visited left-to-right (as usual). Show with an arrow the move that MAX should choose.




***Solution:***

a) Minimax: the backup-values are shown on the figure below. MAX should choose D.




b) Alpha-beta: the backup-values and pruned branches are shown on the figure below. The branches that will be pruned are: OF, QG, IC (and the subtree below) and YK. MAX should select D (as in the previous exercise, of course - minimax and alpha-beta give the same result).