CS188 Spring 2010 Section 3: Game Trees

1 Warm-Up: Column-Row

You have a 3x3 matrix of values like the one below. In a somewhat boring game, player A first selects a row, and then player B selects a column. The game is over after these two moves and the outcome of the game is the value in the square that lies in the intersection of the chosen row and column.

1	8	6
7	3	4
5	9	2

For the following questions, assume that player A wants to maximize the final number that is selected. For each question state which action the player takes and justify your decision in one sentence.

(a) What is player A's move if player B is trying to minimize the final number? Draw out the corresponding game tree.

(b) What is player A's move if player B is moving randomly? Draw out the corresponding game tree, labeling non-leaf nodes with their expected value assuming B moves randomly and A maximizes expected value.

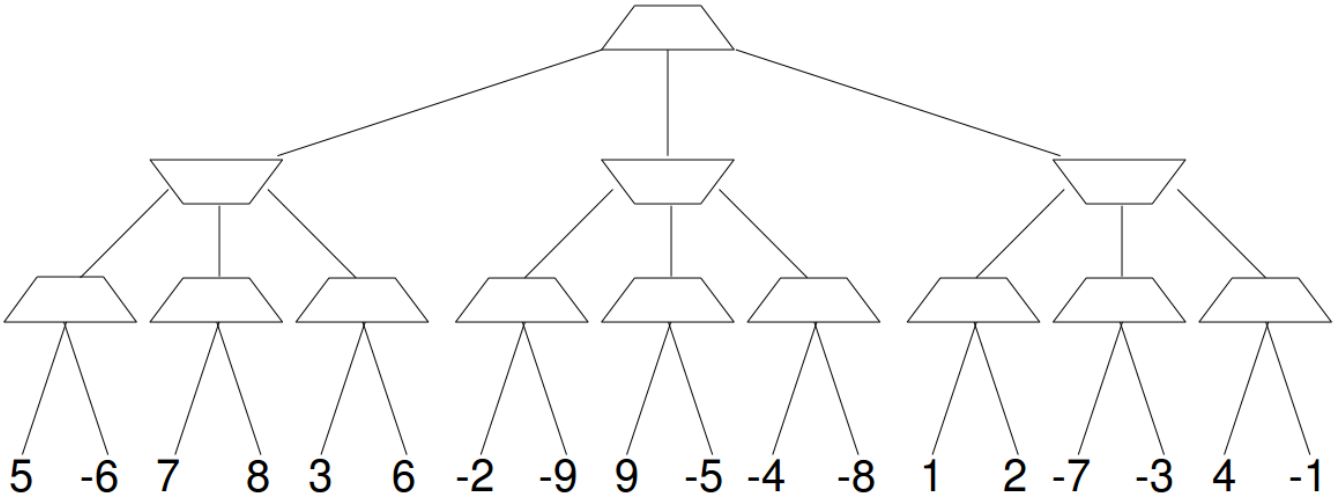
(c) What is player A's move if player B shares A's value function (i.e. wants to maximize the final value)? Draw out the corresponding game tree.

2 Min-Max Search

In this problem, we will explore adversarial search.

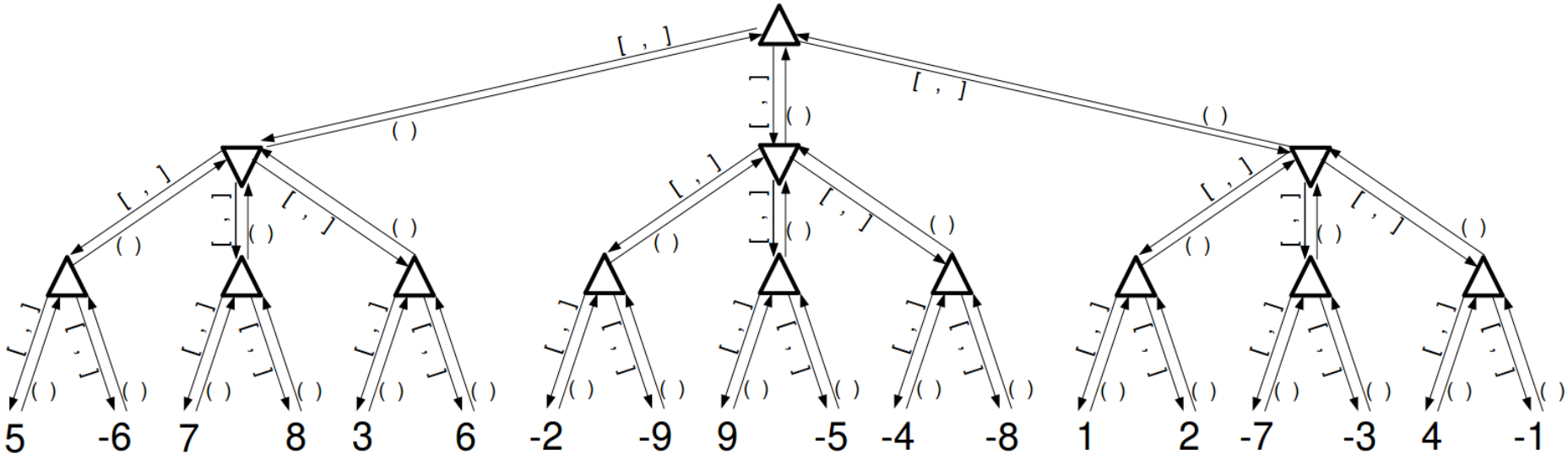
Consider the zero-sum game tree shown below. Trapezoids that point up, such as at the root, represent choices for the player seeking to maximize; trapezoids that point down represent choices for the minimizer. Outcome values for the maximizing player are listed for each leaf node. It is your move, and you seek to maximize the expected value of the game.

(a) Assuming both opponents act optimally, carry out the min-max search algorithm. Write the value of each node inside the corresponding trapzoid. What move should you make now? How much is the game worth to you?



(b) Now reconsider the same game tree, but use α-β pruning (the tree is printed on the next page). Expand successors from left to right. In the brackets [,], record the $[\alpha, \beta]$ pair that is passed down that edge (through a call to MIN-VALUE or MAX-VALUE). In the parentheses ( ), record the value (v) that is passed up the edge (the value returned by MIN-VALUE or MAX-VALUE). **Circle all leaf nodes that are visited. Put an 'X' through edges that are pruned off.** How much is the game worth according to α-β pruning?

(b)

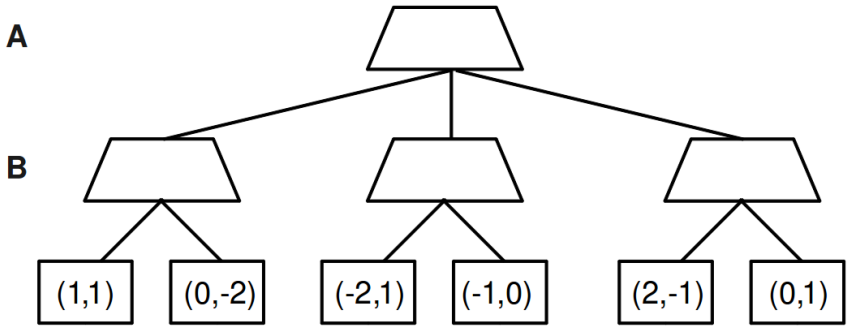


3 Non zero-sum games

The standard Minimax algorithm calculates worst-case values in a *zero-sum* two player game, i.e. a game in which for all terminal states s, the utilities for players A (max) and B (min) obey $U_A(s) + U_B(s) = 0$. In the zero sum case, we know that $U_A(s) = -U_B(s)$, and so we can think of player B as simply minimizing $U_A(s)$.

In this problem, you will consider the *non zero-sum* generalization in which the sum of the two players' utilities are not necessarily zero. Because player A's utility no longer determines player B's utility exactly, the leaf utilities are written as pairs (U_A, U_B), with the first and second component indicating the utility to A and B respecively. In this generalized setting, A seeks to maximize U_A, while B seeks to **maximize** U_B.

(a) Consider the non zero-sum game tree below. Propagate the terminal utility pairs up the tree using the appropriate generalization of the minimax algorithm on this game tree. Fill in the values (as pairs) at each of the internal nodes. Assume that each player maximizes their own utility and that the root node is an A node. In cases of ties, choose the leftmost child.



(b) Briefly explain why no α-β style pruning is possible in the general non zero-sum case. *Hint:* think first about the case where $U_A = U_B$ for all nodes.

(c) For minimax, we know that the value v computed at the root (say for player A = MAX) is a worst-case value, in that, if the opponent MIN doesn't act optimally, the actual outcome v' for MAX can only be better, never worse, than v. In the general non zero-sum setup, can we also say that the value v_A computed at the root is a worst-case value, or can A's outcome be worse than the computed v_A if B plays suboptimally? Briefly justify.

Now consider the *nearly zero sum* case, in which case $|U_A(s) + U_B(s)| \leq \epsilon$ for some ϵ which is known in advance. For example, the game tree from part (a) is nearly zero sum for $\epsilon = 2$.

(d) In the nearly zero sum case, pruning is possible. List the nodes in the game tree above that could be pruned with the appropriate generalization of α-β pruning. Assume that the exploration is done in the standard left to right depth first order, and that the value of ϵ is known to be 2. Make sure you make use of ϵ in your reasoning.

(e) Give a general condition under which a child n of a B node b can be pruned. Your condition should generalize *alpha*-pruning and should be stated in terms of quantities such as the utilities $U_A(s)$ and/or $U_B(s)$ of relevant nodes s in the game tree, the bound ϵ, and so on. Do not worry about ties.

(f) In the nearly zero sum case with bound ϵ, what guarantee, if any, can we make for the actual outcome u' for player A (in terms of the value U_A of the root) in the case where player B might act suboptimally?