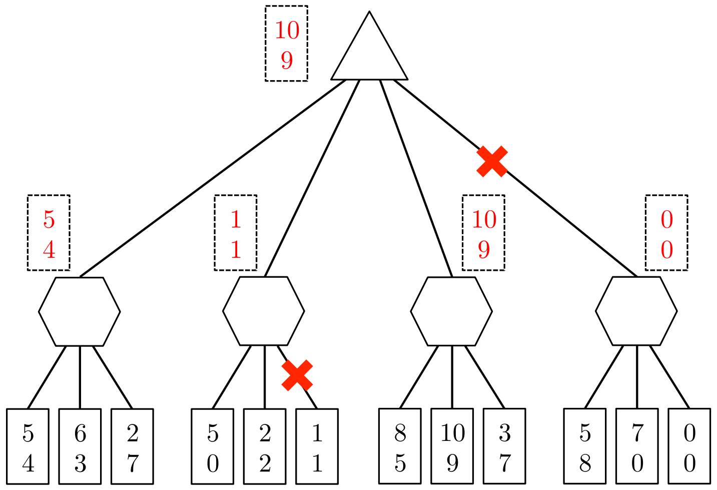
Q1. Game Trees and Pruning

You and one of the 188 robots are playing a game where you both have your own score.

- The maximum possible score for either player is 10.
- You are trying to maximize your score, and you do not care what score the robot gets.
- The robot is trying to minimize the absolute difference between the two scores. In the case of a tie, the robot prefers a lower score. For example, the robot prefers (5,3) to (6,3); it prefers (5,3) to (0,3); and it prefers (3,3) to (5,5).

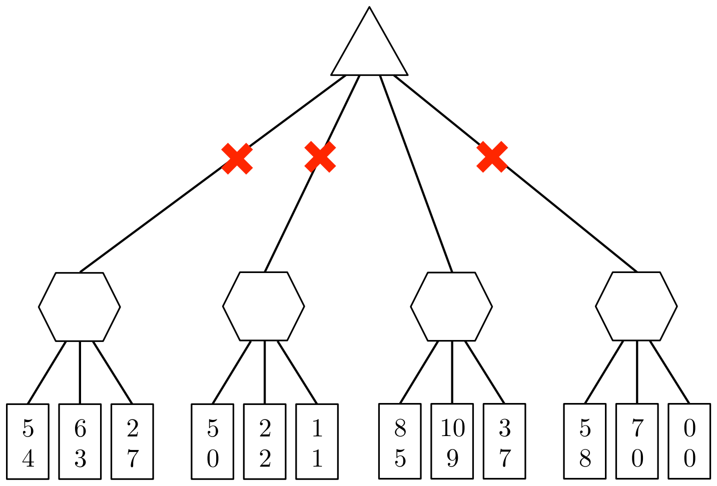
The figure below shows the game tree of your max node followed by the robots nodes for your four different actions. The scores are shown at the leaf nodes with your score always on top and the robots score on the bottom.

(a) Fill in the dashed rectangles with the pair of scores preferred by each node of the game tree.



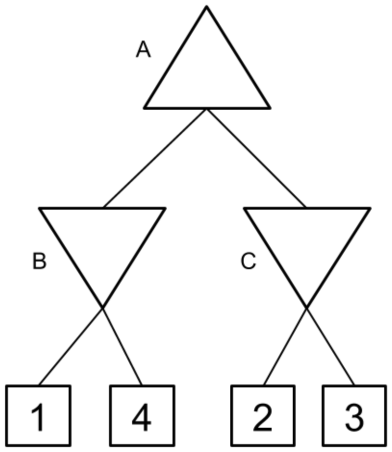
(b) You can save computation time by using pruning in your game tree search. On the game tree above, put an 'X' on line of branches that do not need to be explored. Assume that branches are explored from left to right.

(c) You now have access to an oracle that tells you the order of branches to explore that maximizes pruning. On the copy of the game tree below, put an 'X' on line of branches that do not need to be explored given this new information from the oracle.



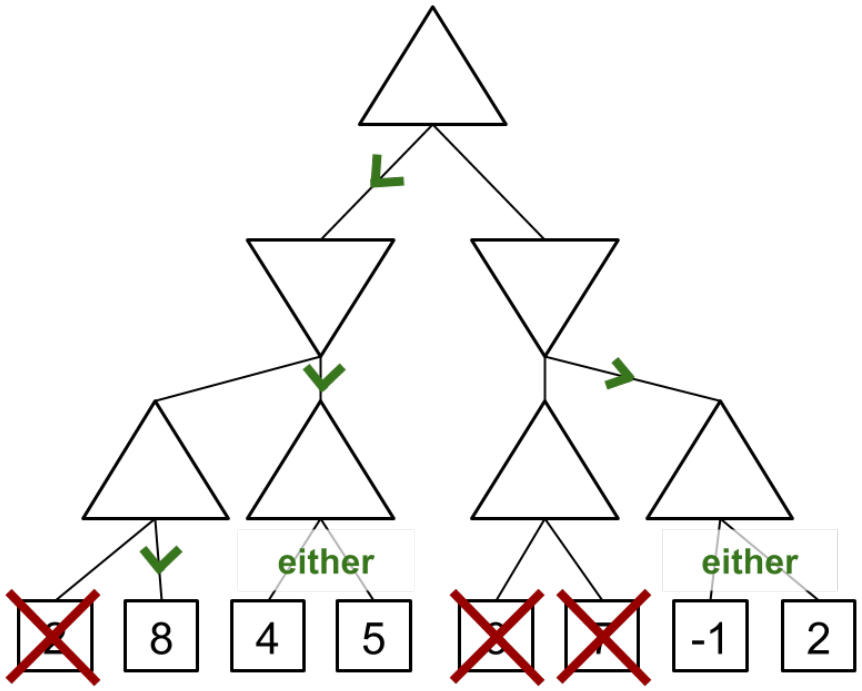
Q2. Alpha-Beta Pruning

The number of nodes pruned using alpha-beta pruning depends on the order in which the nodes are expanded. For example, consider the following minimax tree.



In this tree, if the children of each node are expanded from left to right for each of the three nodes then no pruning is possible. However, if the expansion ordering were to be first Right then Left for node A, first Right then Left for node C, and first Left then Right for node B, then the leaf containing the value 4 can be pruned. (Similarly for first Right then Left for node A, first Left then Right for node C, and first Left then Right for node B.)

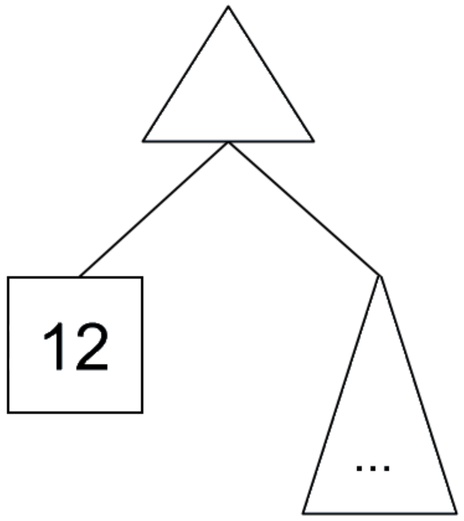
For the following tree, give an ordering of expansion for each of the nodes that will *maximize the number of leaf nodes* that are never visited due the search (thanks to pruning). ***For each node, draw an arrow indicating which child will be visited first. Cross out every leaf node that never gets visited.***
Hint: Your solution should have three leaf nodes crossed out and indicate a child ordering for six of the seven internal nodes.



They key to this problem is understanding how alpha-beta pruning works conceptually. A node is pruned from under a max node if it "knows" that the min node above it has a better – smaller – value to pick than the value that the max node just found. Similarly, a node is pruned from under a min node if it knows that the max node above it has a better – larger – value to pick than the value that the min node just found.

Q3. Utilities

Pacman is in a dilemma. He is trying to maximize his overall utility in a game, which is modeled as the following game tree.



The left subtree contains a utility of 12. The right subtree contains an unknown utility value. An oracle has told you that the value of the right subtree is one of -3, -9, or 21. You know that each value is equally likely, but without exploring the subtree you do not know which one it is.

Now Pacman has 3 options:

1. Choose left;
2. Choose right;
3. Pay a cost of c = 1 to explore the right subtree, determine the exact utility it contains, and then make a decision.

(a) What is the expected utility for option 3? $12 * \frac{2}{3} + 21 * \frac{1}{3} - 1 = 15 - 1 = 14$

(b) For what values of c (for example, $c > 5$ or $-2 < c < 2$) should Pacman choose option 3? If option 3 is never optimal regardless of the value for c, write None.
The utility of option 3 is $15 - c$ and the utility of option 1 is 12. As a result, option 3 will produce an optimum utility if $15 - c \geq 12$.
Therefore, $c \leq 3$.