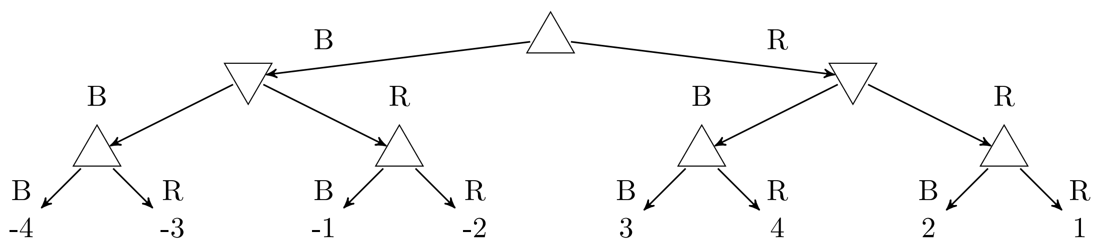
Collab Worksheet 9

CS440/ECE448, Spring 2021

Week of 4/12 - 4/17, 2021

Question 1

The following **minimax tree** shows all possible outcomes of the RED-BLUE game. In this game, Max plays first, then Min, then Max. Each player, when it's their turn, chooses either a blue stone (B) or a red stone (R); after three turns, Max wins the number of points shown (negative scores indicate a win for Min).



(a) Suppose that the tree above is searched using alpha-beta search. Of the eight leaf nodes, how many can be pruned?

> **Solution:** One: the node with value -2 never needs to be evaluated, because -1 is higher than $\beta = -3$.

(b) Recall that an $\alpha - \beta$ search prunes the largest possible number of moves if there is extra information available to the players that permits them to evaluate the moves in the best possible order. IN GENERAL (not just for this game tree),

- In what order should the moves available to MAX be evaluated, in order to prune as many moves as possible?
- In what order should the moves available to MIN be evaluated, in order to prune as many moves as possible?

> **Solution:**
>
> - Moves available to MAX should be evaluated in order of descending value, starting with the highest-value move.
> - Moves available to MIN should be evaluated in order of ascending value, starting with the lowest-value move.

(c) Re-draw the minimax tree for the RED-BLUE game so that, if moves are always evaluated from left to right, the $\alpha - \beta$ search only needs to evaluate 5 of the 8 terminal states.

> **Solution:** The first two leaves should be (1,2) or (2,1). The next two leaves should be (3,4) or (4,3). The last four leaves should be (-1,-2,-3,-4), in any order.

Question 2

What additional difficulties does dice throwing or other sources of uncertainty introduce into a game?

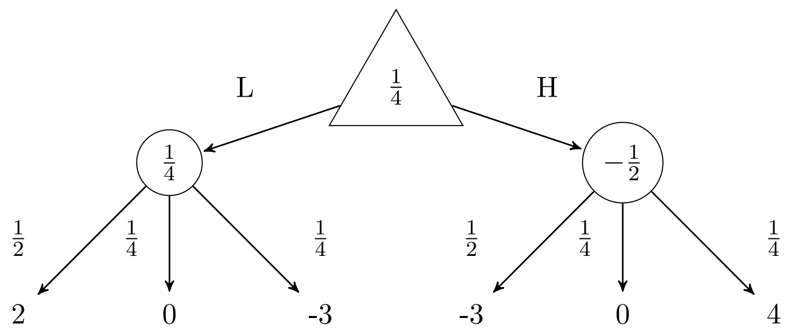
> **Solution:** Uncertainties introduce probabilities into the game. Expectiminimax is used to find solutions for these type of games. Expectiminimax doubles the number of levels as compared to minimax, because there is randomness after every play. Expectiminimax has nasty branching factor and often times defining evaluation functions and pruning algorithms are difficult.

Question 3

Consider the following game, called "High/Low." There is an infinite deck of cards, half of which are 2s, one quarter are 3s, and one quarter are 4s. The game starts with a 3 showing. After each card, you say "High" or "Low," and a new card is flipped. If you are correct (e.g., you say "High" and then the next card is higher than the one showing), you win the points shown on the new card. If there is a tie (the next card equals the one showing), you get zero points. If you are wrong (e.g., you say "High" and then the next card is lower than the one showing), then you lose the amount of the card that was already showing.

Draw the expectiminimax tree for the first round of this game and write down the expected utility of every node. What is the optimal policy assuming the game only lasts one round?

Solution:



Optimal policy: L