Extra credit part

```
def future_eval(r, b, turn):
    x = abs(r - b)
    return -2 if turn and x % 2 == 0 else 2 if not turn and x % 2 == 0 else -2 if x % 2 == 0 else 2
```

When the given depth is lower than the tree depth this part activates.

I found a loophole in this game playing using this “future_eval” it will function. I can make a game-playing agent like a constant time. O(1).

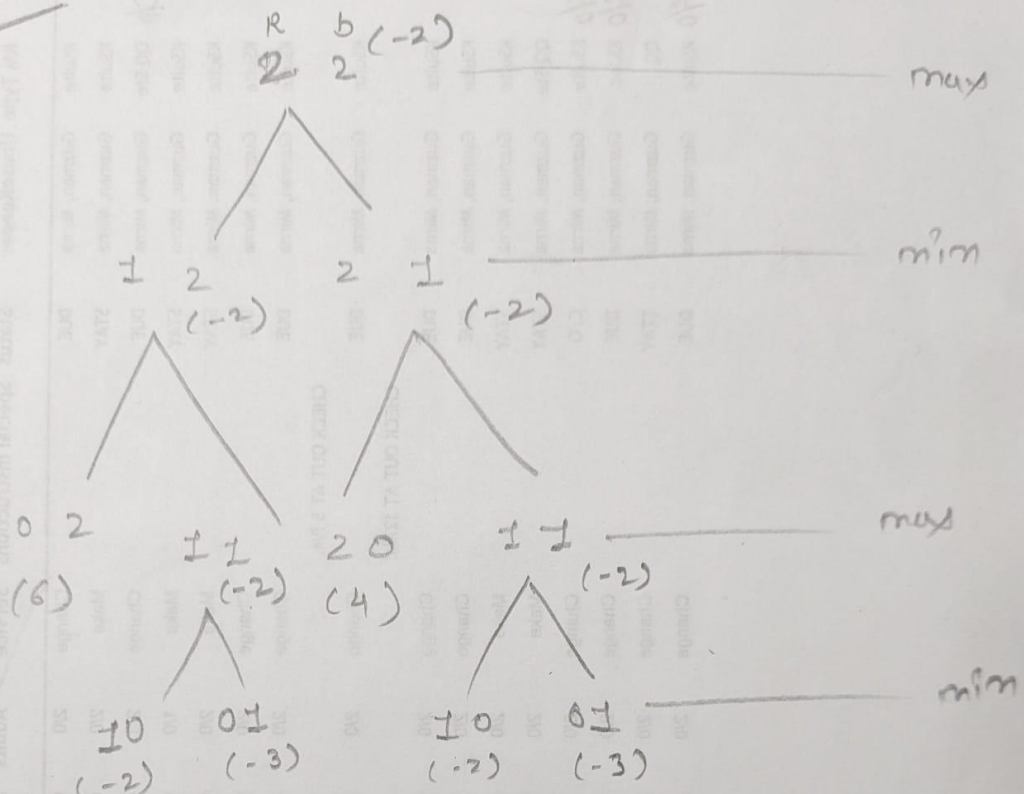
“future_eval(r, b, turn)”

This function takes three arguments, the number of red tokens, the number of blue tokens, and a boolean turn that indicates which player is making the move. The function returns a score for the given state based on the difference between the number of red and blue tokens. If it is the turn of the computer, the function returns 2 if the difference is even and -2 if it is odd. If it is the turn of the human, the function returns 2 if the difference is even and -2 if it is odd.

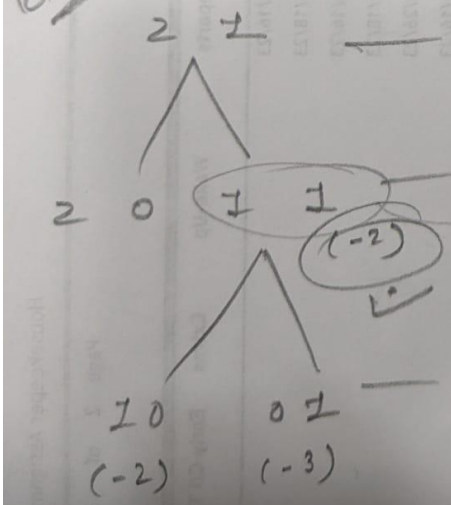
Logic:

For better understanding, I attach a tree picture take any tree or any node, and apply this function on it you will get optimal output always, note: only against an optimal player.

Example-1




Example-2




Take any true any node value and put into fuction.

(1, 1, True)

↳ (1 - 1) = 0.

↳ if True:

↳ 0.1.2 == 0:

↳ return -2

The future_eval function takes three parameters as input: the current number of red and blue stones on the board, and a boolean flag turn indicating which player's turn it is. The function calculates a score for the current state of the game based on the difference between the number of red and blue stones remaining, and whether it is the maximizing player's or the minimizing player's turn.

If turn is True, it means that the maximizing player (i.e., the computer) is playing, and the function returns a score of 2 if the difference between the number of red and blue stones is even, and -2 if it is odd.

On the other hand, if turn is False, it means that the minimizing player (i.e., the human) is playing, and the function returns a score of 2 if the difference between the number of red and blue stones is odd, and -2 if it is even.

The function is designed to provide a heuristic evaluation of the current state of the game, which can be used by the minimax algorithm to determine the best move to make. The idea behind the heuristic is that if the number of stones remaining on the board is relatively balanced, then it is advantageous for the maximizing player to take the next turn, while if there is a significant imbalance, it is better for the minimizing player to have the next turn.