Emily Foreacre

Purpose: Analyzing the game, Nim, by using game trees to demonstrate minimax, alpha-beta pruning, and optimization techniques.

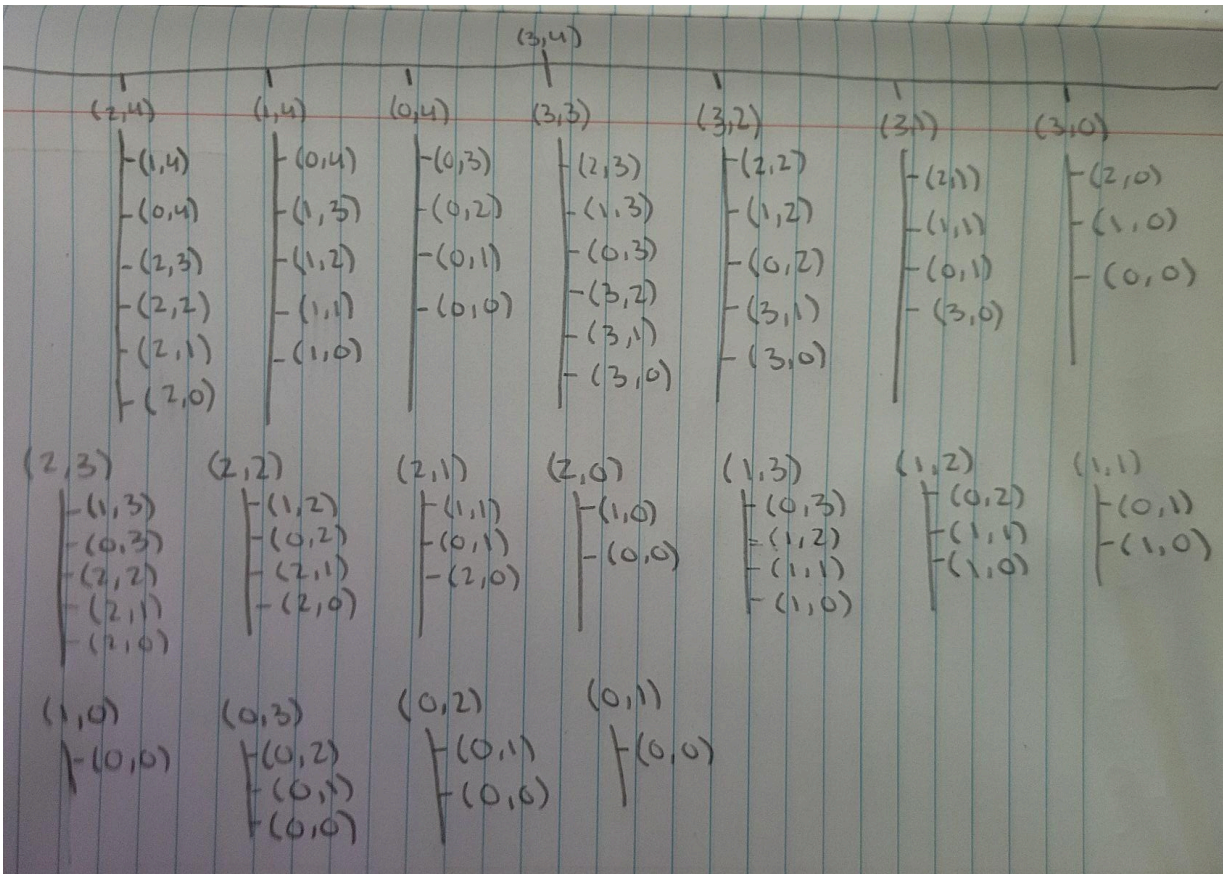
Problem 1

Here, you will analyze the game of Nim using the minimax algorithm and alpha-beta pruning. Instead of implementing the game in code, you will create a written report that includes the following:

1. Game Description: Explain the rules of the game of Nim, including how many heaps there are, how many objects are in each heap at the start of the game, and how players take turns removing objects from the heaps.

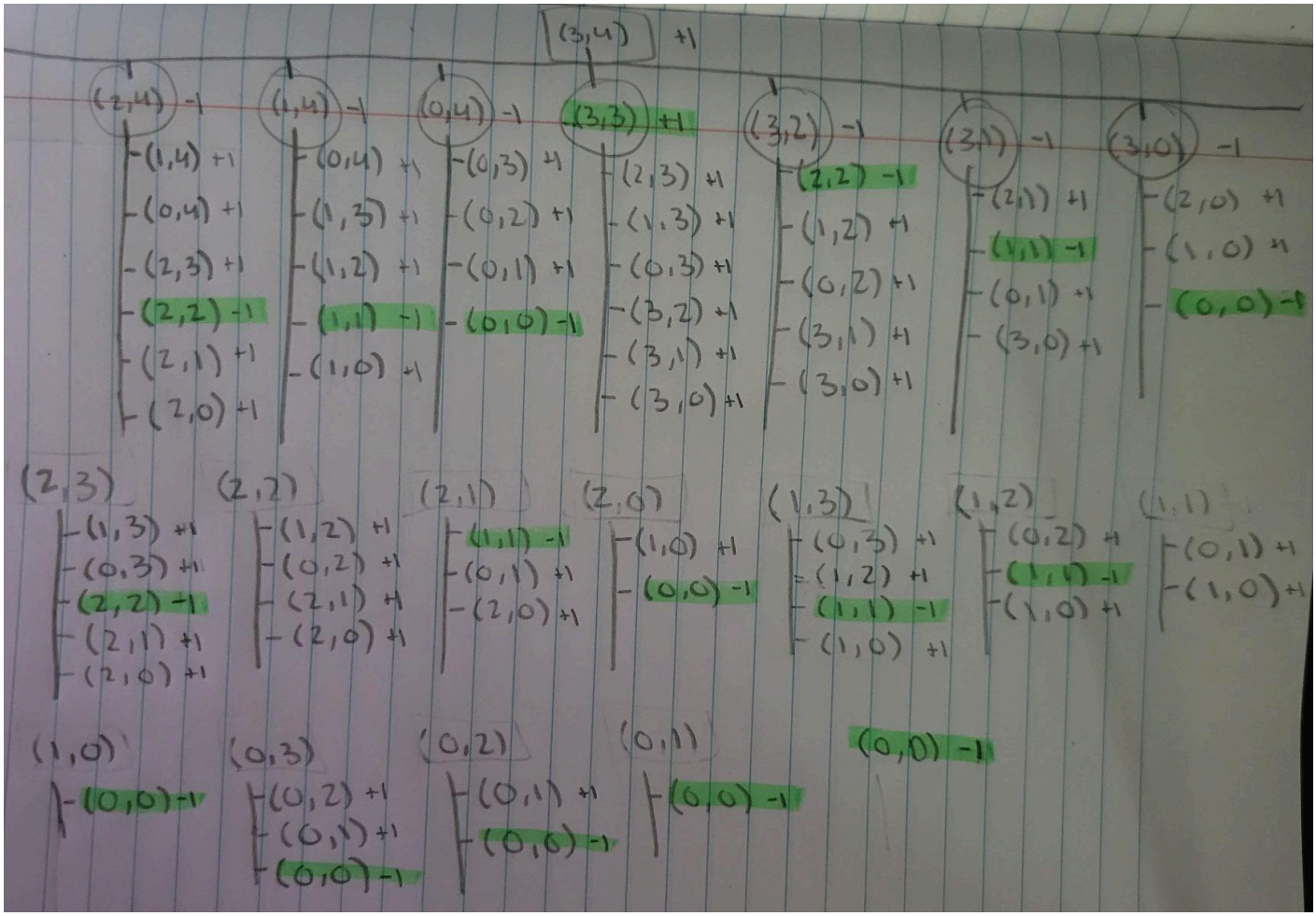
Nim is a two player game where you have two or more piles, or heaps. Each of the piles contains a number of stones. In the example, it's 3 stones in one pile and 4 stones in the other, but it can be any number of stones, [2,4], [5,5], [9,4], etc. Each player takes turns taking as many stones as they want from one of the piles. The player to take the last stone standing wins.

2. Game Tree: Draw the game tree for a simple case of Nim, such as a game with two heaps containing 3 and 4 objects, respectively. Show all possible moves and their outcomes at each level of the tree.



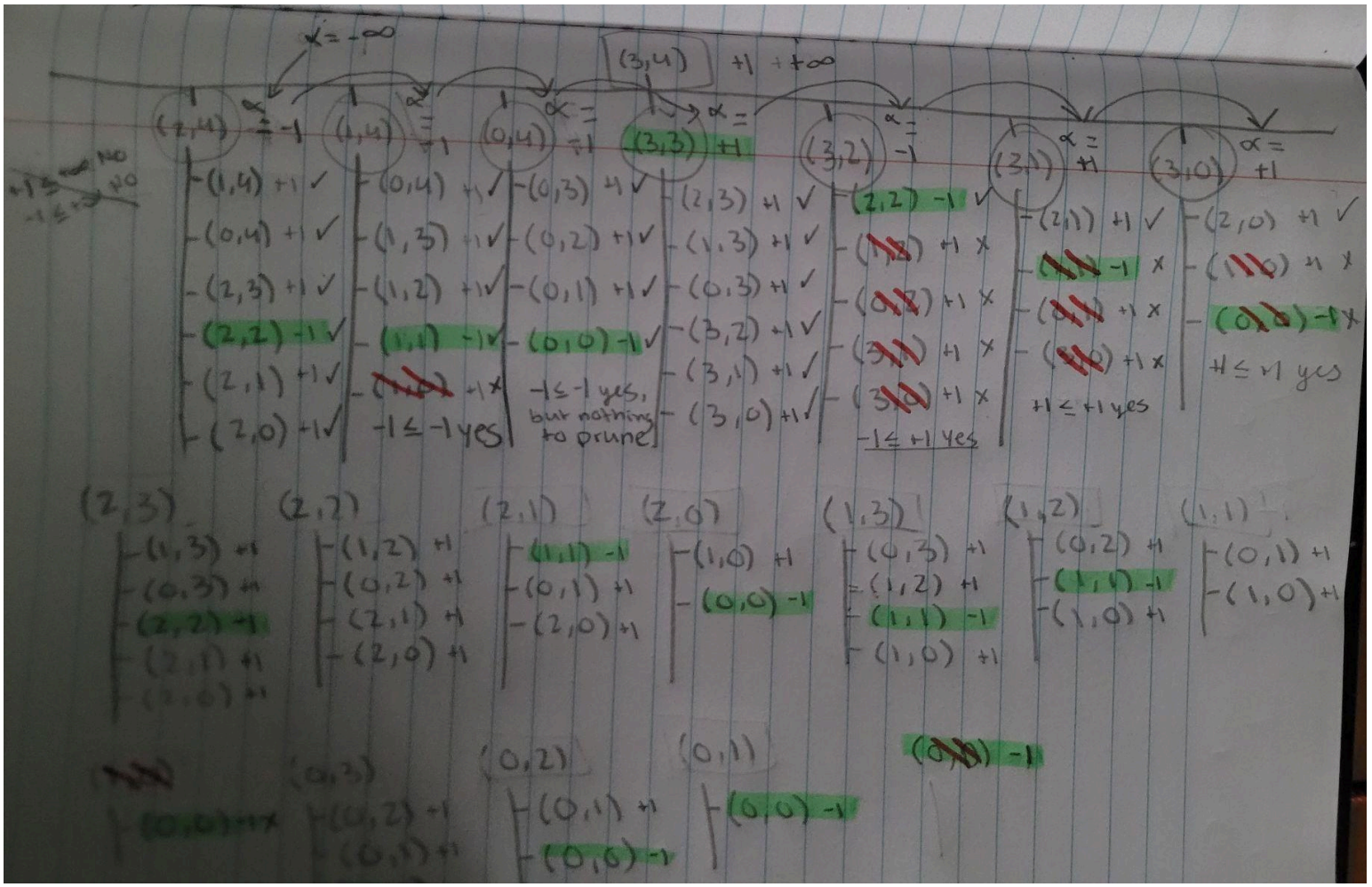
I got a little creative on how the game tree looks so that I would have enough room to add everything.

3. Minimax Algorithm: Apply the minimax algorithm to the game tree to determine the optimal move for each player at each level. Show the values assigned to each node in the tree by the minimax algorithm in a 2-ply game.



What is highlighted in green is what either min or max choose.

4. Alpha-Beta Pruning: Apply the alpha-beta pruning algorithm to the game tree to reduce the number of nodes that need to be evaluated. Show the alpha and beta values at each level of the tree and indicate which nodes were pruned.



The red slanted lines indicate the pruning that was done.

5. Analysis: Discuss the effectiveness of the minimax algorithm and alpha-beta pruning in solving the game of Nim. Consider factors such as the size of the game tree, number of heaps, number of objects in each heap, the number of nodes pruned by alpha-beta pruning, and the complexity of the game.

Using the minimax algorithm for a game such as Nim is effective, since it explores every possible game state and ensures optimality. However, it depends on the size and complexity of the game tree. In other words, how many heaps and objects there are in the game. In a game of Nim with the size (3,4), there are only a few dozen possible states. In a larger game of Nim, say (12,15), the possible number of game states can reach into the thousands, or even millions. When that happens, the branches and depth increases and the minimax algorithm can become expensive with a time complexity of O(b^d).

Alpha-beta pruning improves the performance of Minimax by pruning branches in the game tree that doesn't affect the final decision. In the (3,4) example above, pruning reduced the possible game states by about 30%, but in larger games of Nim, pruning could reduce the game states by more than 30%. Alpha-beta pruning doesn't change the outcome of minimax, but it does reduce the time of the algorithm itself, which is the best when it comes to larger games of Nim.

Problem 2

1. Optimization Techniques: Explore additional optimization techniques for the minimax algorithm, such as the use of iterative deepening, to improve its performance in analyzing larger game trees. Use a 2-ply game to analyze and explain your strategy.

Iterative deepening is a way to make the minimax algorithm more efficient by running it multiple times with increasing depth limits. Each time, it looks a bit deeper into the game tree, and it keeps track of the best move that can be made. This also helps with move ordering, which can also make alpha-beta pruning more effective. Making alpha-beta pruning more efficient improves the performance of larger game trees.

For a 2-ply Nim game with heaps and objects (3,4), has several child nodes, such as (2,4), (1,4), (0,4), (3,3), (3,2), (3,1), and (3,0). In the first iteration, the algorithm just considers these moves and sees which seem strongest. In the next iteration, it looks at Player 2's possible responses and applies minimax logic: Player 2 tries to minimize Player 1's advantage, and Player 1 tries to maximize it. Using results from the first iteration to order moves, and adding on alpha-beta pruning, helps prune unnecessary branches. This makes the search faster while still finding the best move available.

2. Heuristic Evaluation: Develop a heuristic evaluation function to estimate the value of game states in Nim, and compare the expected performance of heuristic-based minimax with the standard minimax algorithm. Discuss your approach using a 2-ply game.

The heuristic that I came up with estimates the advantage of player 1 by evaluating states based on the largest pile size and the number of stones in the game, larger piles increase the score, and smaller piles decrease it. Positive values indicate states favorable to player 1, and while negative values are game states that favor player 2.

Using a 2-ply game with piles (3,4) as an example, Player 1 has several possible first moves, each leading to game states for Player 2. The heuristic evaluates the resulting nodes. For example, removing 1 stone from pile 2 leads to (3,3), which we saw was the best choice in problem 1, maintains a large pile and scores higher than a choice like (0,4) or (3,0), which gives the opponent an strong position from the start since they can just take 4 from pile 2, or take 3 from pile 1 and win the game. This heuristic works because maintaining a large pile gives player 1 more control and options in later moves, increasing the likelihood of favorable outcomes.

Standard minimax evaluates all possible responses to find the optimal move, but requires many node evaluations. Heuristic-based minimax, in contrast, assigns scores to early nodes, significantly reducing computation while often choosing the same optimal move.

The heuristic helps order moves for alpha-beta pruning, and allows near-optimal choices in larger or more complex games. While it may occasionally select the non-optimal move due to the limited depth or simplified scoring, it provides a practical and efficient method for estimating game state value in Nim.