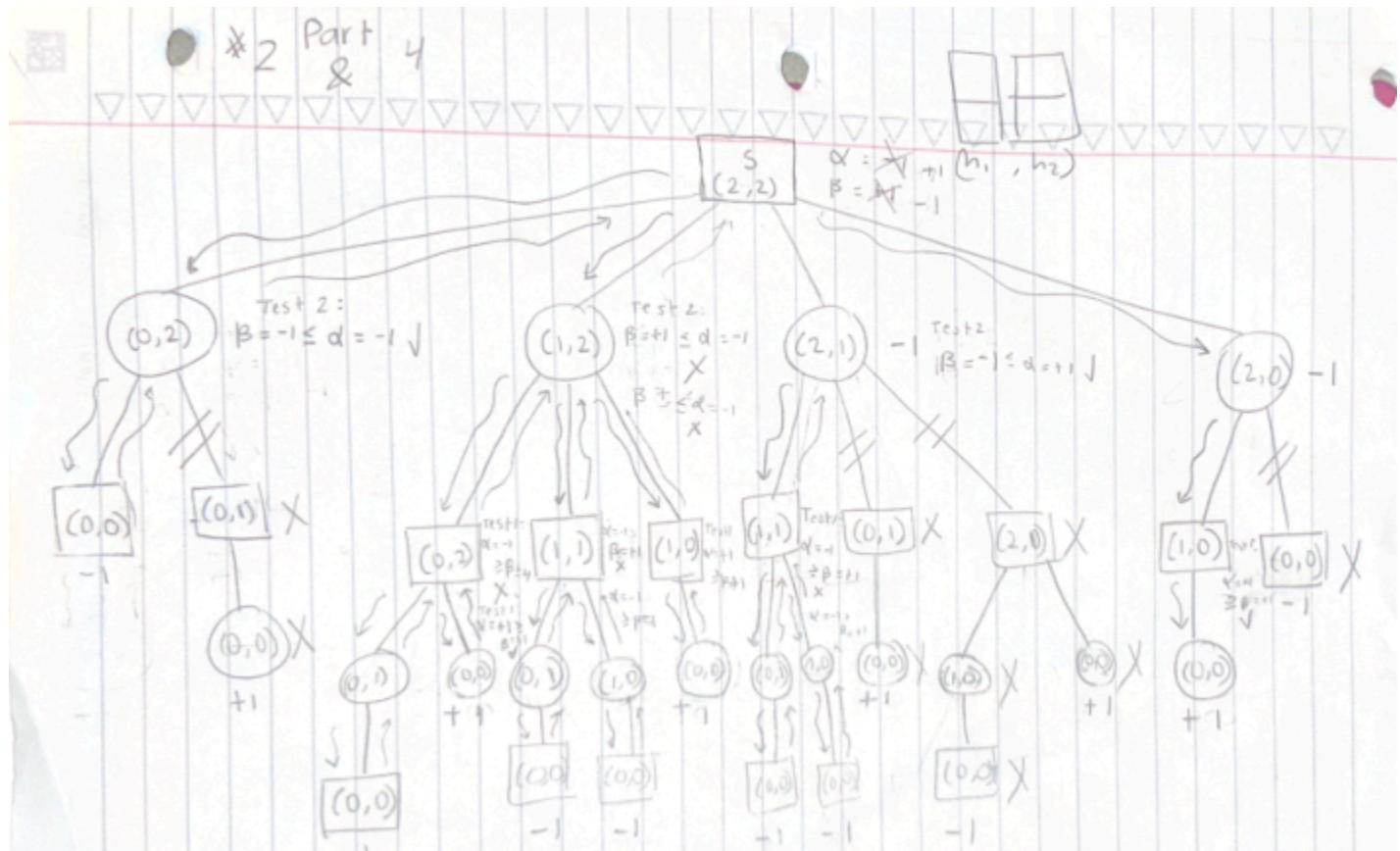


### Assignment 4

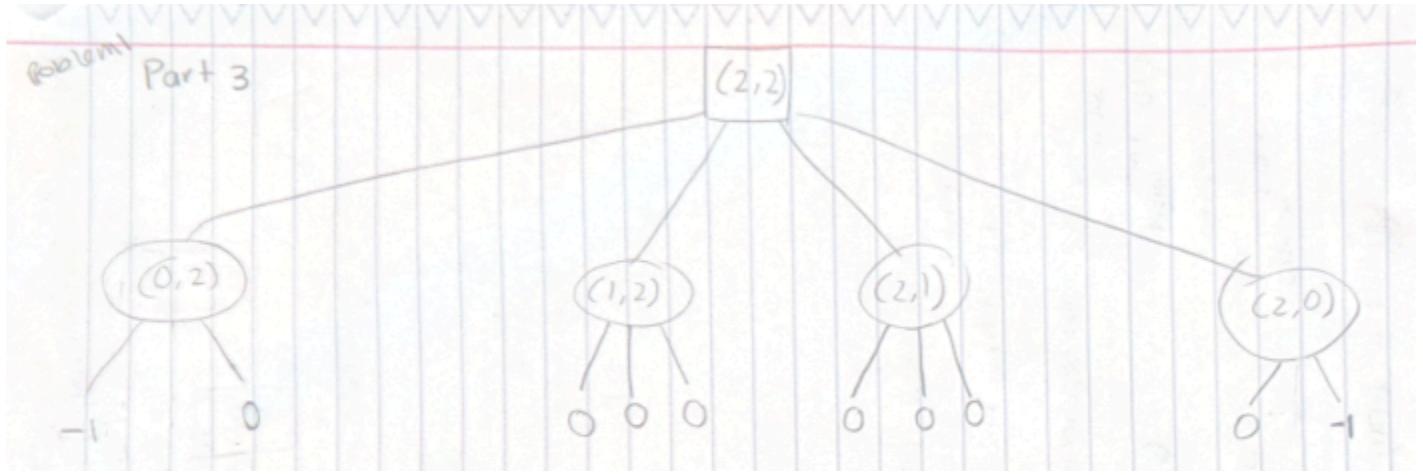
#### Problem 1

- Game Description:** The game is played with 2-players. Nim begins with a distinct number of heaps. In the example provided and most commonly played, there are three starting heaps. The quantity within each heap also can vary. Traditionally, there are 3, 5, and 7 objects in each heap according to a quick Google search. Ultimately, those heaps can contain any number of objects. The two players alternate removing objects from the heaps and cannot skip their turns. The player to remove the final object wins the game.

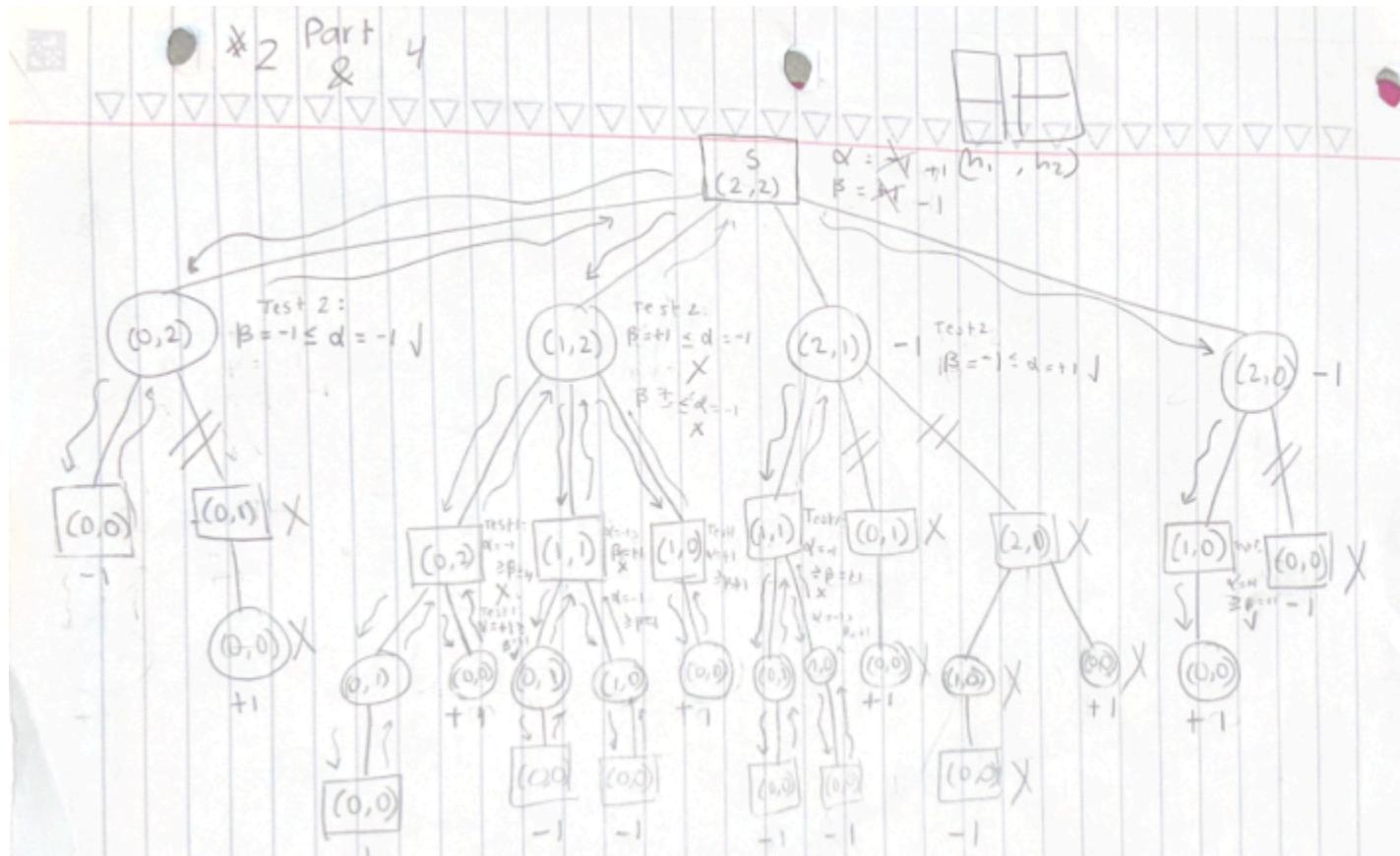
- Game Tree:**



### 3. Minimax Algorithm:



### 4. Alpha-Beta Pruning:



**5. Analysis:** The minimax algorithm was effective for this game (2 heaps, 2 objects per heap) given that there were not many branches. However, if a different Nim game was set up with significantly more heaps and these heaps were larger (higher complexity), the decision tree would get very big. If the tree were this big, it would take much more time and space to explore. Thus, it would become inefficient after a point. I pruned at four points which saved time and space in solving the game of Nim. I have 33 nodes total and by pruning at these four points I did not explore 9 nodes (4 terminal nodes). I had 14 terminal nodes total. Therefore, my undeveloped percentage was about 28.6%. This is fairly good. If my child nodes were in a different order, and I still explored left to right, this percentage has potential to increase or decrease slightly.

## Problem 2

**1. Optimization Techniques:** There are optimization techniques for the minimax algorithm. The iterative deepening algorithm works similarly to a breadth-first search but has the time and space complexity of a depth-first search. The DFS space complexity is more desirable than BFS's. The interactive deepening search also maintains the optimality and completeness of BFS. This would work very well with Nim which, generally, does not have a large branching factor. Nim, in general, is a shorter game; it does not require many turns to reach a terminal node.

In the case of the 2-ply game, the algorithm first explores the child nodes from the starting position before making its initial move, effectively practicing a depth-fist search approach. After examining the first-level child nodes, the algorithm increases the search depth by one to evaluate the next level of child nodes, continuing until it reaches a terminal node. In this instance, the terminal node corresponds to a winning position for the minimizing player.

**2. Heuristic Evaluation:**  $f' = \frac{\# \text{ heaps with odd size}}{\# \text{ nonzero heaps}}$  This heuristic only considers piles that are active (nonzero) measures how many have odd counts. The idea of having more odd counts favors the current player since in the game of Nim heap sizes affect winning potential. Ignoring empty heaps can keep the game moving. The new states in the two ply

game have the calculated  $f'$  values of 0, 0.5, and 0.5 (see diagram). This heuristic is computationally faster than the standard minimax algorithm as demonstrated in problem one. It also has a gradient measure of advantage. I hypothesize that if the number of heaps increased that this heuristic would still be helpful.

