Adam Syed

11/6/25

CSC 362

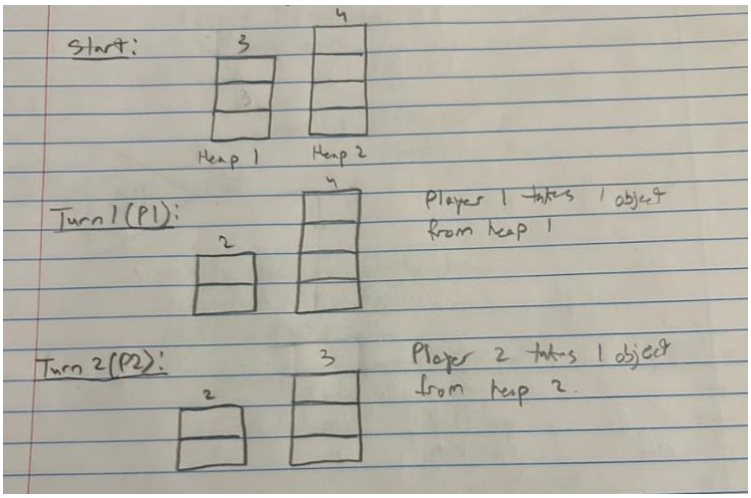
Assignment 4

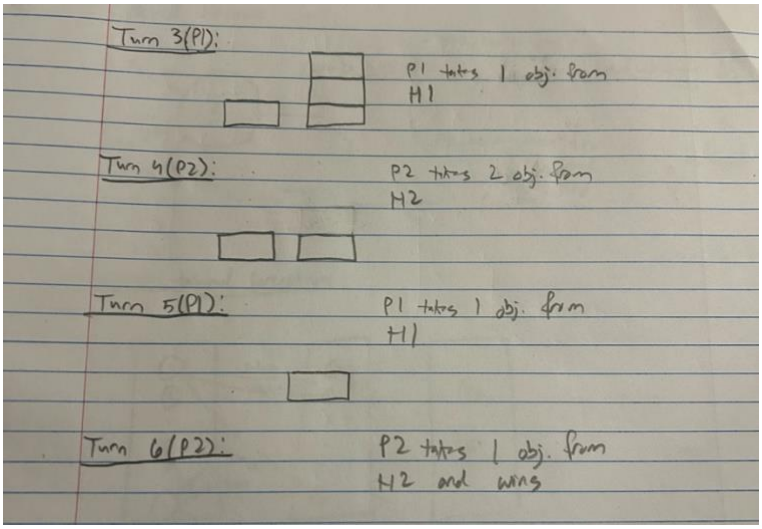
Purpose: Analyze the game of Nim using Minimax and Alpha-Beta Pruning algorithms. Explore optimization techniques of Minimax using iterative deepening. Formulate a Minimax heuristic and compare to standard Minimax algorithm.

Problem 1

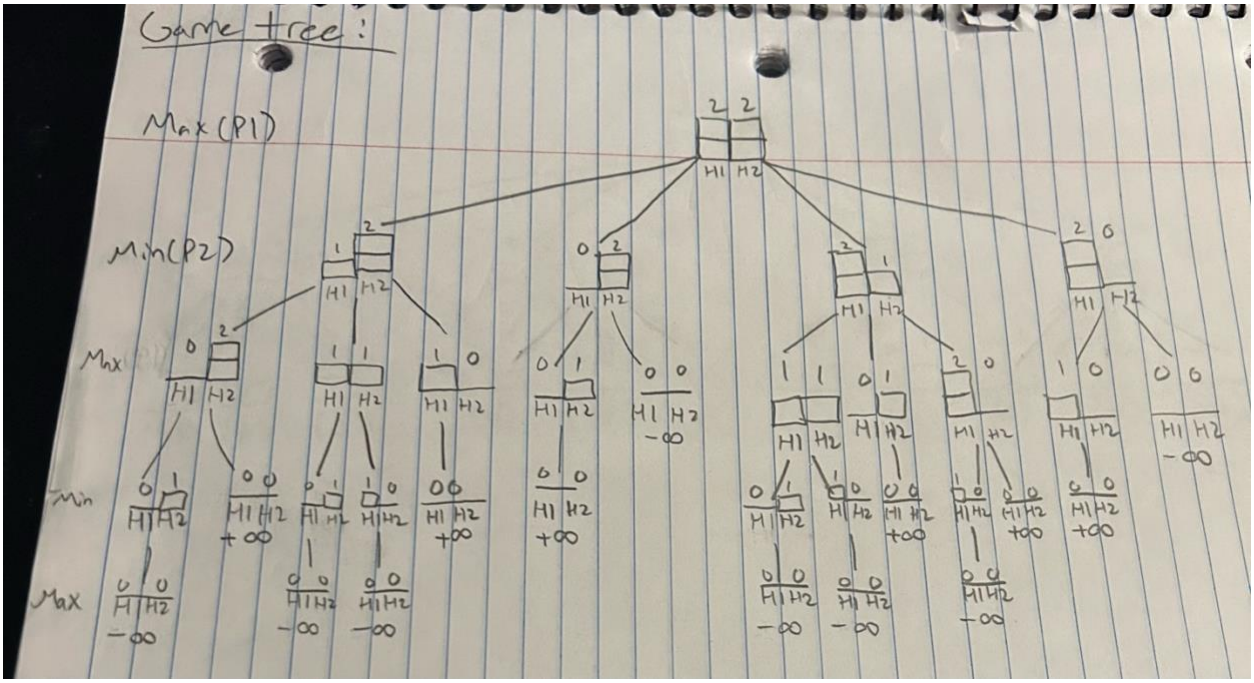
1.) Game description:
   - Rules -> Nim is a 2-player game where each player takes turns removing objects from a heap. The game will start with a chosen number of heaps and objects in those heaps. Each turn, a player must take at least one object from a single heap. The player to remove the last object wins the game.

   - Number of heaps: 2
     - Heap 1 = 3 objects
     - Heap 2 = 4 objects

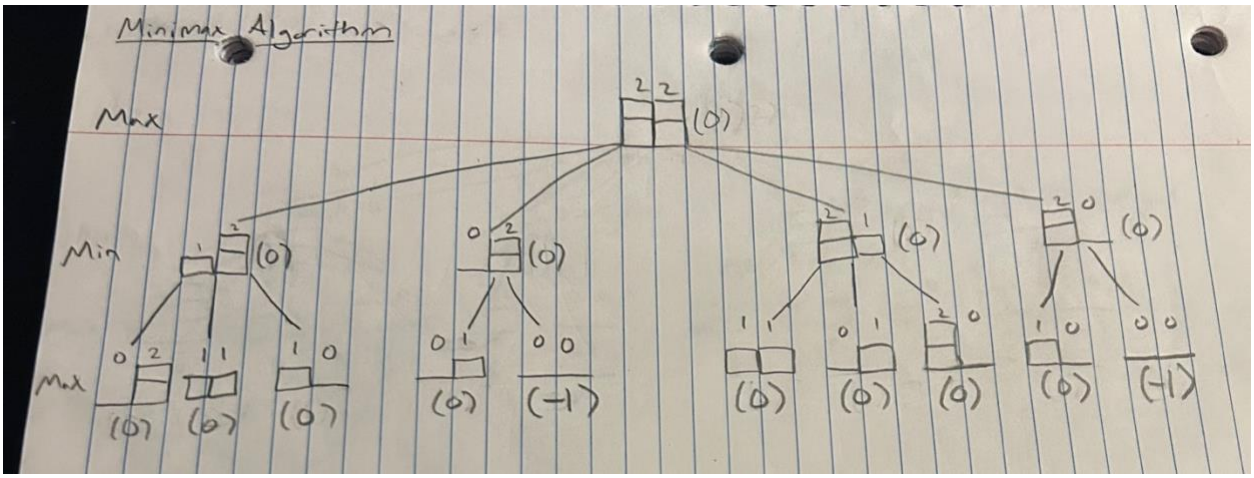




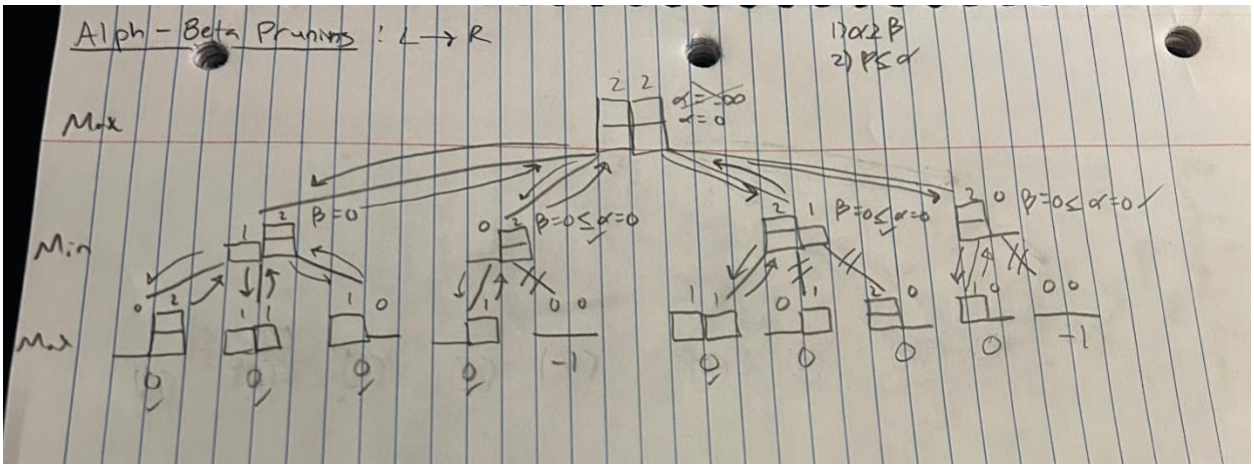
2.) Game Tree:



3.) Minimax Algorithm:



4.) Alpha-Beta Pruning:

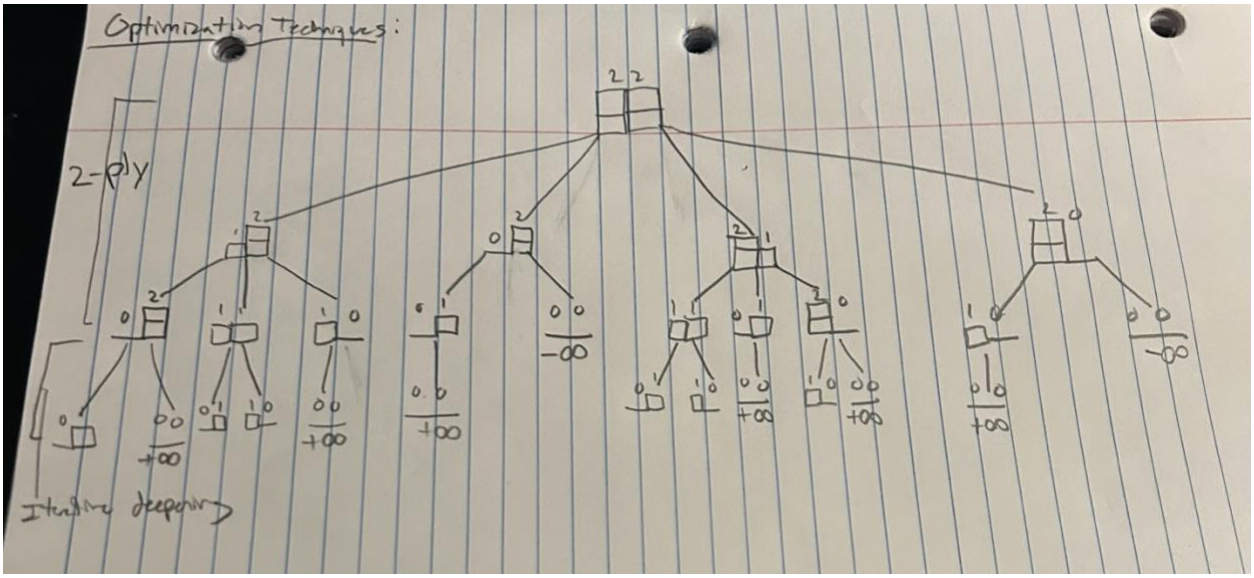


5.) Analysis:

- **Minimax algorithm** is an effective way to determine the optimal strategy in a game of Nim because it searches through every node to determine the best possible outcome. The algorithm becomes inefficient as the number of heaps and objects per heap increases. The tree will grow exponentially, making minimax impractical for large configurations of Nim because its time complexity will become too great to manage.
- **Alpha-beta pruning** is a more efficient minimax algorithm because of its feature to eliminate branches, reducing the number of nodes that need to be explored. Like minimax, its effectiveness lies in the number of heaps and objects in those heaps. But unlike minimax, direction of which the moves are searched also effects the algorithm. With proper ordering and direction, the alpha-beta pruning will be able to significantly reduce the number of nodes evaluated. However, it is still prone to the exponential complexity of large games of Nim, making it impractical to search that number of nodes.

Problem 2

1.) Optimization Techniques:
   - Using iterative deepening in a minimax algorithm would greatly improve its performance in larger game trees. By gradually increasing the search depth one level at a time, it is able to find the best possible move in shallower searches. In deeper search, these solutions might be found later in the search greatly increasing the computation time. If you were to add alpha-beta pruning to the iterative deepening, it would improve the search even more as it would identify promising moves sooner to eliminate branches.
   - When examining a 2-ply game of Nim, the maximizing player begins by exploring all nodes within a depth of 2 (for a 2-ply game). With iterative deepening, if no immediate winning path is found, then it depends to three plies, then four, and so on, until a winning path is discovered. This ensures the algorithm explores efficiently without over committing computational resources to a deep search.
   - Examining the picture of the 2-ply game with heaps [2,2], we can see that after the first search of the tree does not discover any winning path. It then iteratively deepens to 3-ply, where a winning path is found. This is where it would stop its search.



2.) Heuristic Evaluation:

- A heuristic evaluation that would estimate the value of game states in Nim would be as follows: f = num of heaps remaining + (num of objects / total num of heaps). This heuristic would determine which branch max would choose based on the highest f-value. For min, you would change the sign (-) so it would choose the lowest.
- This heuristic would work faster than a standard minimax algorithm. Though operates faster, the standard minimax algorithm would be able to 100% guarantee an optimal path while this heuristic would not guarantee the best path in all cases. In general, it would operate faster but miss the goal in larger cases.