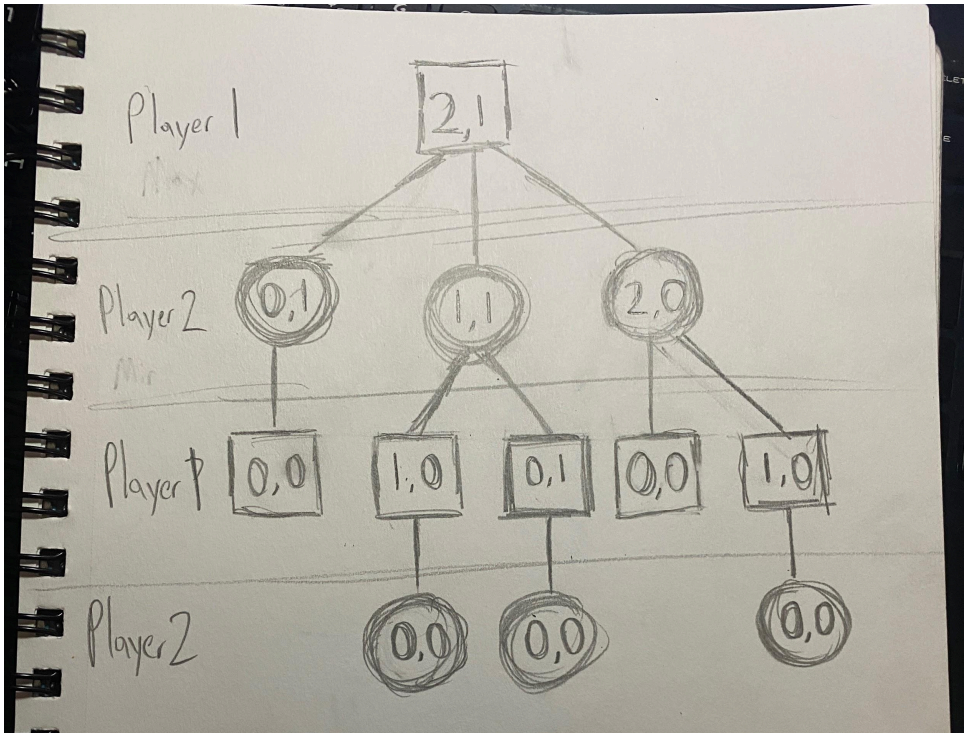
James Letts

Professor Babak Forouraghi

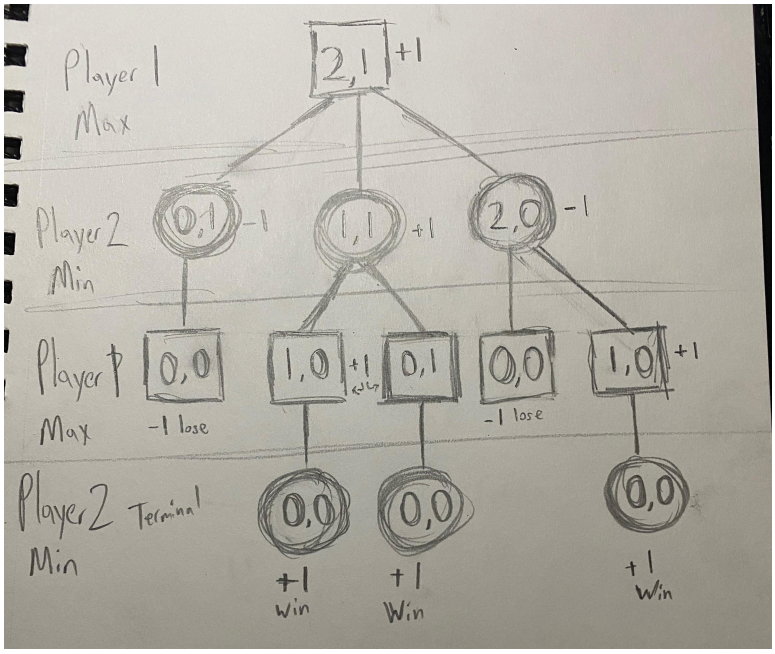
Artificial Intelligence CSC 680 G01

11/06/2025

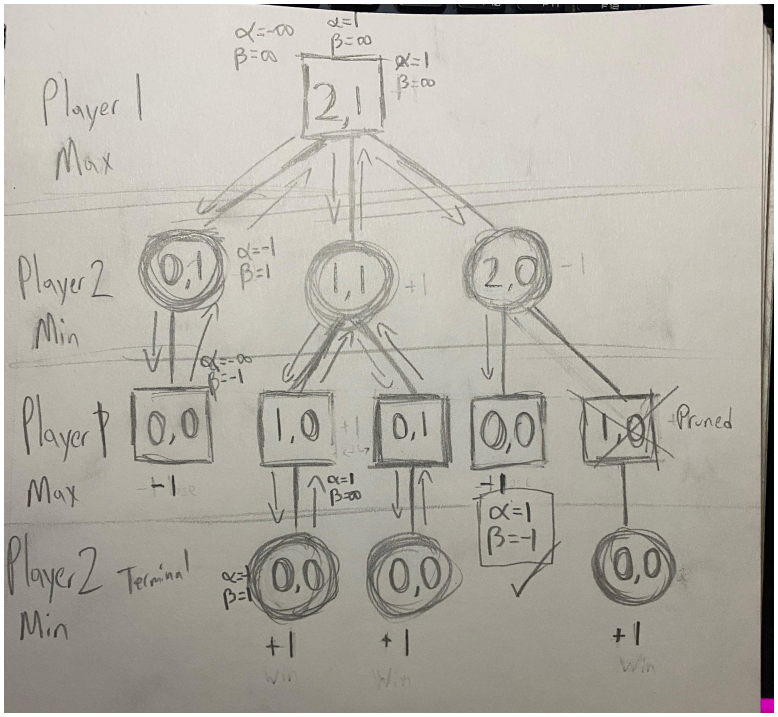
1. **Problem 1:**
    1. Nim is a game where two players are given two or more stacks/heaps of differing amounts of stones/objects. On each turn, a player can take as many stones from one heap as they want, from one stone to the entire heap, but they cannot take from more than one heap in one turn. The aim of the game is to be the last person to take a stone from the heap, with the last person taking the stone being the winner.
    2. The following is a game tree depicting a game of Nim with two heaps, one heap having 2 stones the other having 1 stone.



3. The following is a game tree depicting a game of Nim with two heaps, one heap having 2 stones the other having 1 stone. This tree was evaluated using Minimax Search.



4. The following is a game tree depicting a game of Nim with two heaps, one heap having 2 stones the other having 1 stone. This tree was evaluated using Alpha Beta Pruning.

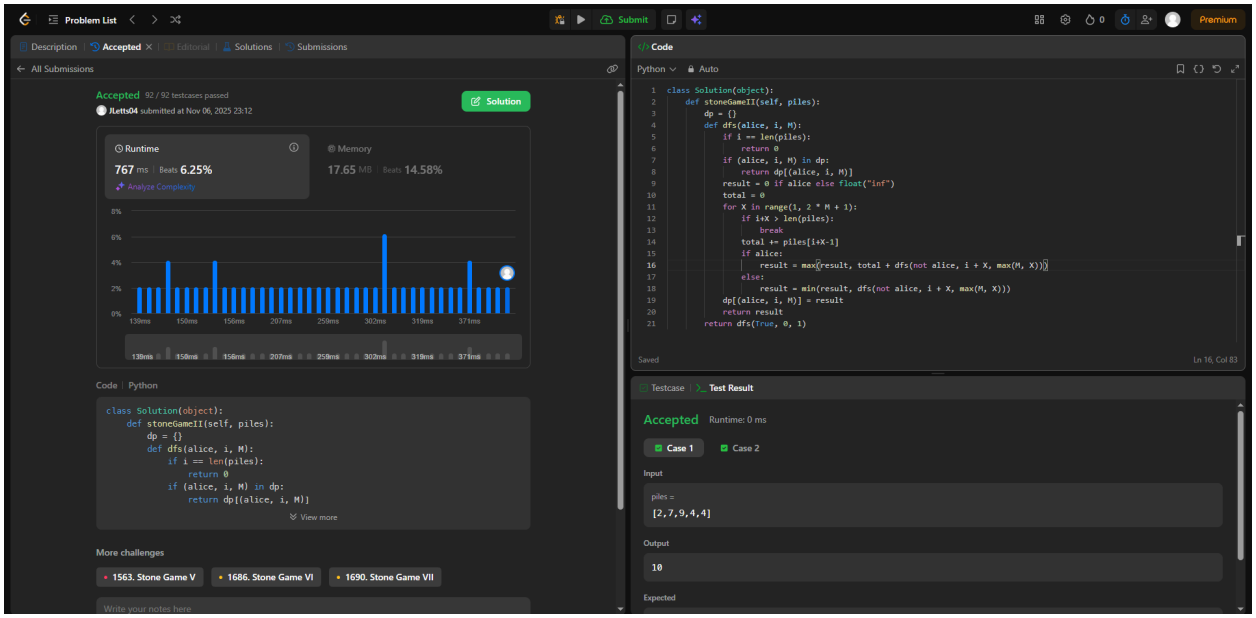


5. After looking at both of the searches, it is safe to say that the best move for Player 1 (Max) to make is to take one object from the first heap, giving us (1,1). This is due to this move being the only move that guarantees that no matter what Player 2 (Min) does, Min is unable to win. Normally, alpha beta pruning is able to out perform minimax search in the majority of situations, and it continues to be correct here, as it rules out the possibility of trying to win on the right, as it will very obviously lead to a loss, and thus showing 1,1 is the optimal path.

2. Problem 2:

1. The best way to optimize this problem would most likely be to use iterative deepening, which allows us to find the best path iteratively, and thus more efficiently. It would look at the first child nodes only first, and for our previous example, it would show that each of the child nodes have a score of -1 except for (1,1), so it has effectively found the most efficient path for Max to take before even going on to the next depth.
2. Heuristics can improve on this game tree with the use of the exclusive or, or xor, at all points of the tree. In Nim, if the xor is equal to 1 then it is a winning position, but if it is equal to 0 then it is not. When used in our (2,1) nim game, you can basically do the same thing as in the iterative deepening, and find the optimal path of (1,1) without needing to go to the child nodes.

3. Problem 3:




```python
class Solution(object):
    def stoneGameII(self, piles):
        dp = {}
        def dfs(alice, i, M):
            if i == len(piles):
                return 0
            if (alice, i, M) in dp:
                return dp[(alice, i, M)]
            result = 0 if alice else float("inf")
            total = 0
            for X in range(1, 2 * M + 1):
                if i+X > len(piles):
                    break
                total += piles[i+X-1]
                if alice:
                    result = max(result, total + dfs(not alice, i + X, max(M, X)))
                else:
                    result = min(result, dfs(not alice, i + X, max(M, X)))
            dp[(alice, i, M)] = result
            return result
        return dfs(True, 0, 1)
```

In the shown code, Alice is the Max, and Bob is the Min, with the code recursing on itself to alternate the turns and to see all moves. It checks for their best move using Max and Min respectively, and uses Memoization to avoid redundancy by storing the results.