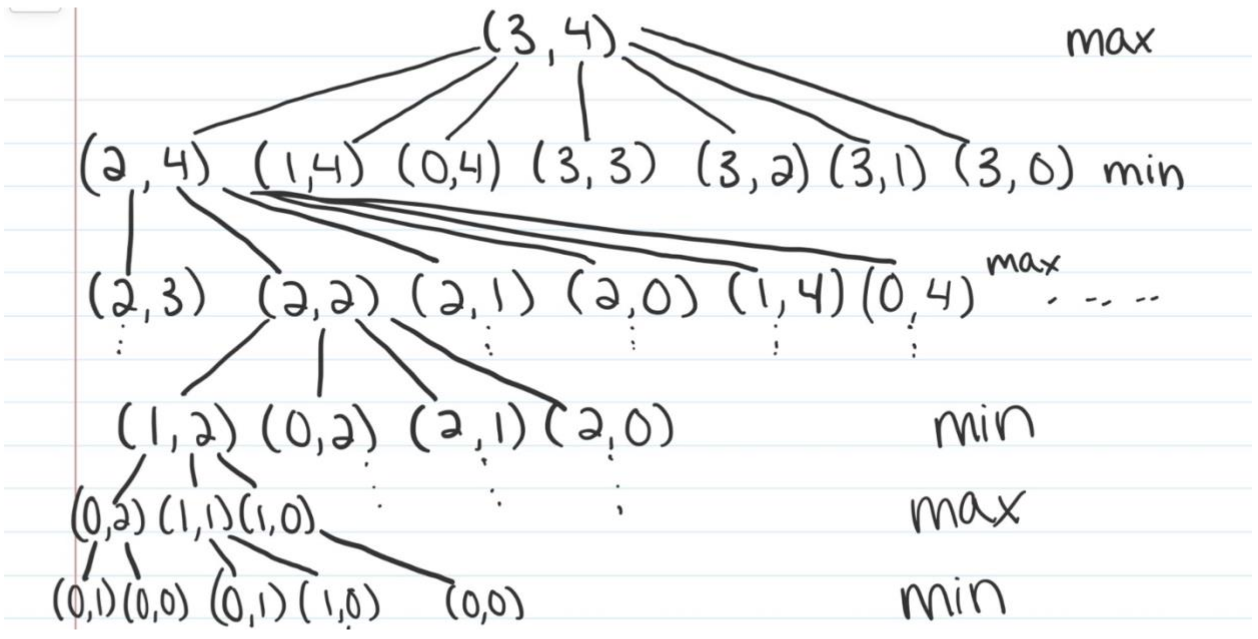
Devon Reing

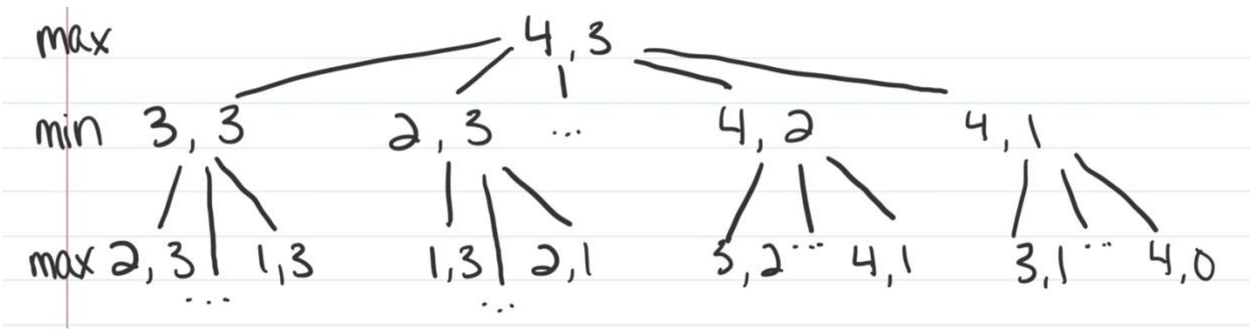
Minimax and Alpha Beta Pruning of Nim

Game Description:

Nim is a two-player game in which a number of stacks with a varying number of objects such as stones are removed until all of the stacks are gone. The number of stacks and objects in each stack are determined during the setup of the game and are entirely up to the players to decide. Each player on their turn can pick from any stack and remove as many objects as they would like. The last player to remove a piece (or the player that removes the last stone) wins the game. A player cannot abstain from their turn and must remove at least one object each time.

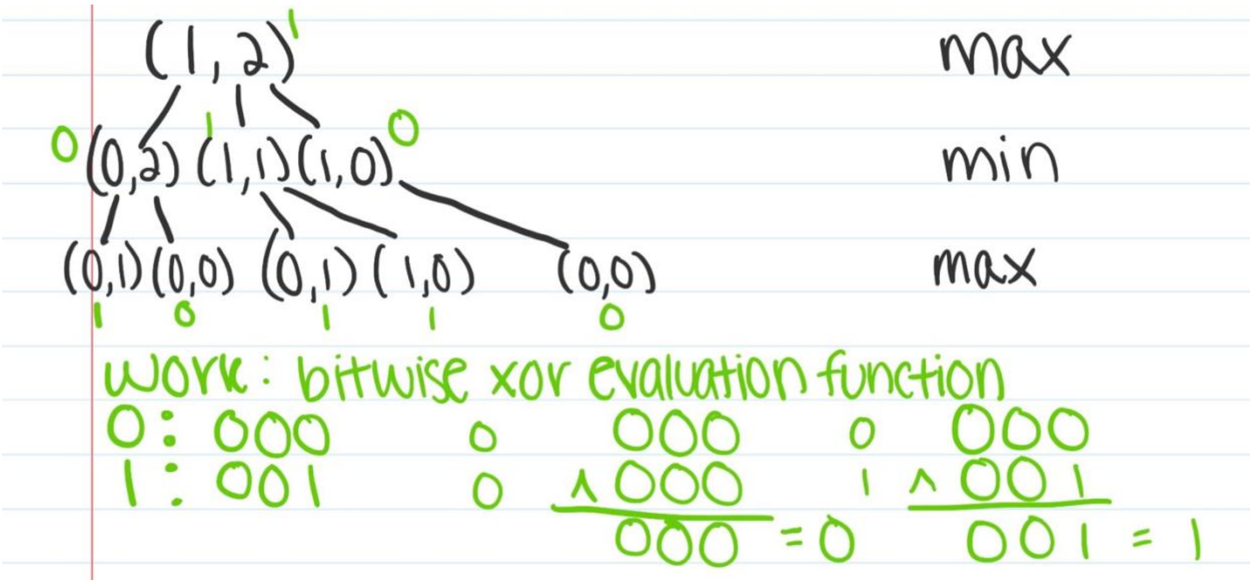
Game Tree:

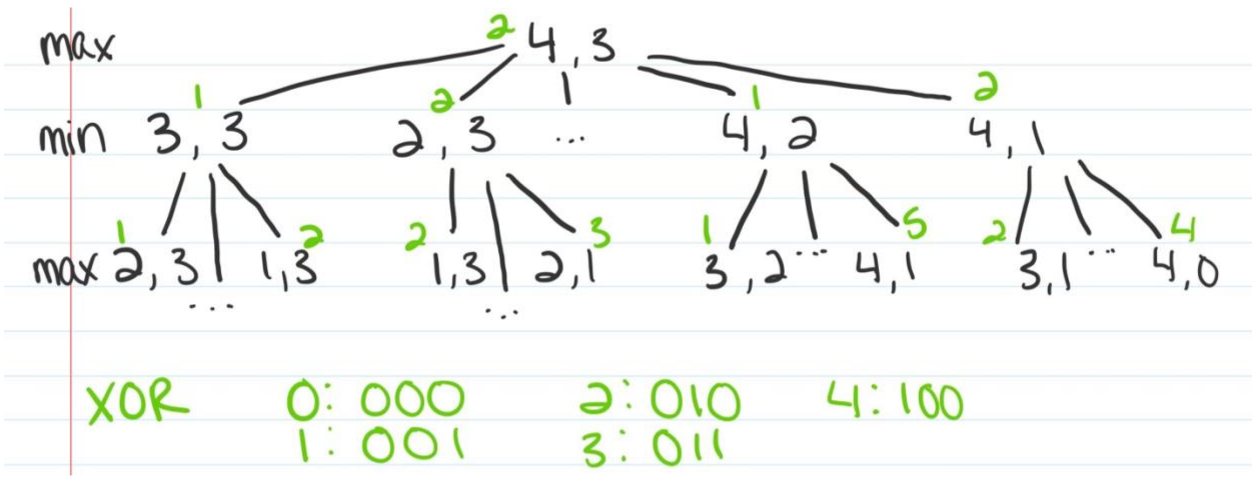




Above are two subsections of the entire game tree for a game of nim consisting of two piles with 3 and 4 stones. Both game trees, while incomplete, shows how quickly the game of nim can expand exponentially in terms of options and game states. In the first game tree, one state is expanded at each level until some terminal nodes are reached at the sixth level. In the second game tree, a 2-ply tree is expanded with some options missing from the tree denoted by "…". In each tree, max is at the top of the tree to signify that max will go first in the game, and then min and max will trade off turns. Both of these trees will be explored in some capacity using minimax and alpha-beta pruning optimization techniques.

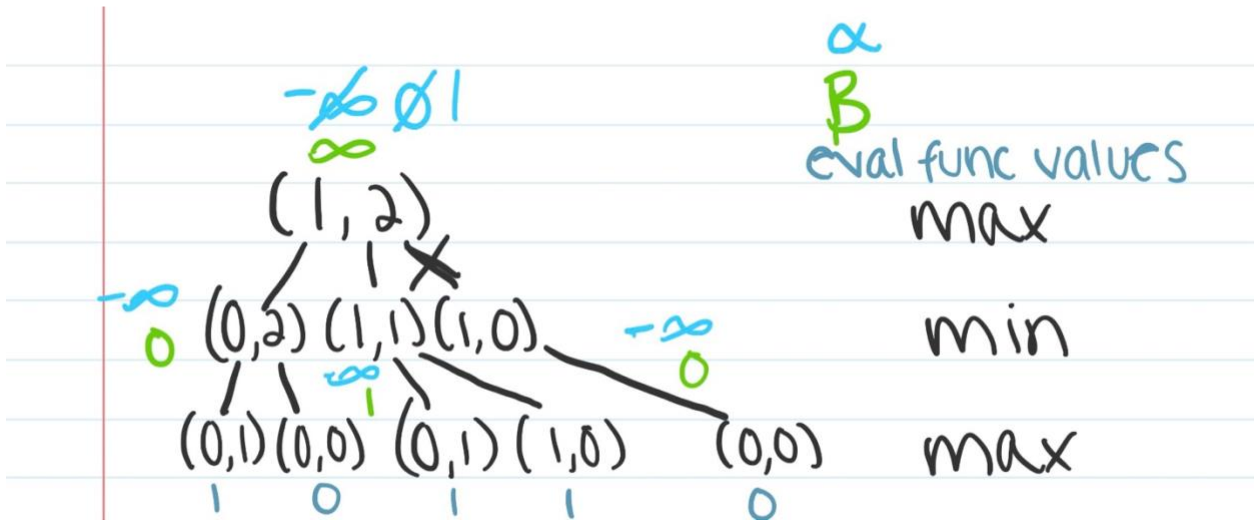
Minimax Game Tree:

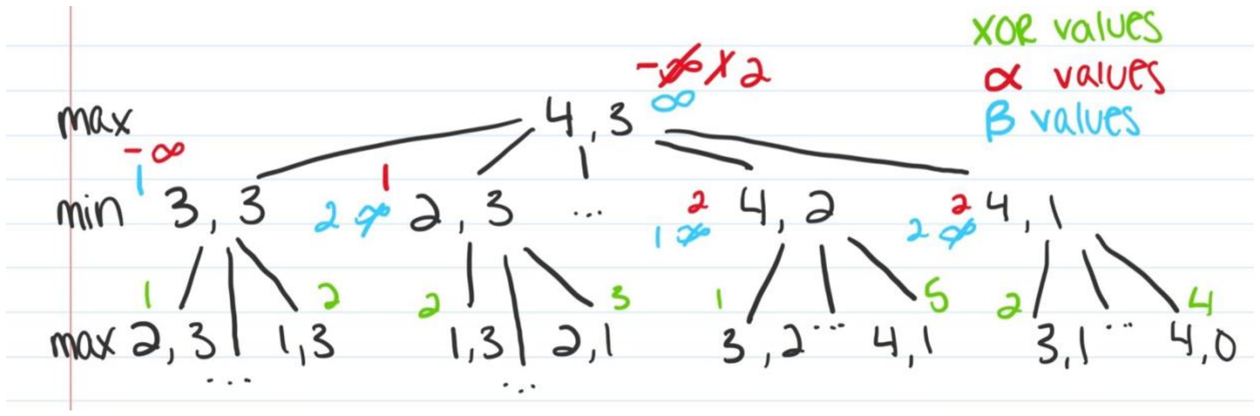




For the minimax optimization technique, a bitwise XOR evaluation function was used. The number of stones in each pile were put through an XOR sum function which then provided a number to be used for picking the min/max at each level. Since the last row in both cases is max, the level above will be min, meaning that the minimum evaluation function value from the child nodes is passed up to be the parent node's value for the next pick. Both of these game trees explore 2 levels of the game tree at varying points in the progression of the game tree. In the first, a subtree with terminal nodes is explored, whereas in the second tree, a 2-ply game is explored from the top, or root node, of the tree. The evaluation function applied to the last row of the trees provide varying values, with the first all being either 0 or 1 and the second having higher values up to 5. This is due to the higher number of stones remaining in the piles because the tree was cut off, meaning higher bitwise values are passed into the evaluation function. This allows for the possibility of higher values to be outputted by the evaluation function. The first graph will lead to a path of 1,2 -> 1,1 -> 0,1 being followed which is optimal as it is max's turn to pick then and will result in a win for max. The second graph will follow a path of 4,3 -> 2,3 -> 1, 3 which can lead to a win eventually for max as well. As both allow for a win for max, the minimax algorithm is effective at finding a win.

Alpha Beta Game Tree:





For the alpha-beta pruning optimization technique, two game subtrees were evaluated again with the original values for the lowest level being assigned from the bitwise XOR evaluation function. In the first game tree provided, a subtree with terminal nodes and two levels was explored. In this tree, one branch was able to be pruned. Using DFS and working left to right, by the time the algorithm was set to explore the rightmost branch (1,0 in the second level), an alpha value of 1 was set to be passed down. Since the value that would have been passed up was 0 and therefore less than 1, that was able to be pruned. This allows the algorithm to save some efficiency. Additionally, this method leads to a win for max as it will follow the path towards 1,1. Either option picked from the child nodes from there will lead to max choosing the last stone, resulting in a win for max. In the second tree, no values are pruned so the optimization method is not necessarily more efficient than the minimax algorithm. Since the tree does not lead to terminal nodes, it is hard to tell how effective it would be at winning the game for max, but based on previous precedent from the first tree, it most likely will also lead to a win.

Part 2

Optimization Techniques:

Using iterative deepening with minimax for Nim allows the computer to save space and time while still making informed decisions to bring max to a win. Iterative deepening can be used as a pruning method similar to alpha-beta pruning in that it will only explore the optimal branch further. Starting off with a 2-ply game, for example, once the best path to the bottom layer is found, the iterative deepening method can then work on only expanding that node's options another n levels. Minimax can then be applied again to the recently expanded branch to find the optimal node/path to further explore at the 2 + n level. You can continue to repeat this process until the terminal nodes are reached. By expanding the game tree iteratively using iterative deepening on the best path found by minimax, the game tree is able to be pruned, eliminating non-optimal paths from needing to be explored.

Heuristic Evaluation:

A heuristic based approach for a 2-ply game will be very similar to the original minimax approach I used earlier for the 2-ply game tree. Since the heuristic based approach is based on cutting off a large game tree and assigning a heuristic value, the same evaluation function (bitwise XOR) or another heuristic such as the total number of stones remaining can both be used. If the same evaluation function is used, the original 2-ply game will be the same as diagram 2 in the minimax analysis above. This can then be continued with iterative deepening as explained above. This prediction of the best path to be used provides a heuristic value that can help guide both min and max to pick the best child node possible to increase their probability of a win. If a different heuristic evaluation is used such as the total number of stones, the game will be lengthened, and efficiency will be sacrificed in order to attempt to save space. Using the total number of stones and picking the branch that has the most stones left allows the game to last longer and will likely allow the game to last long enough to allow a win to be possible for max. However, in this case, the branch chosen from the heuristic evaluation will be deepened as much as possible, taking up more space which is costly. It is also important to not though that it will also allow the game tree to be pruned as it will just focus on expanding the branch or path that is chosen by the heuristic at every step instead of expanding every path to apply minimax to the whole tree.