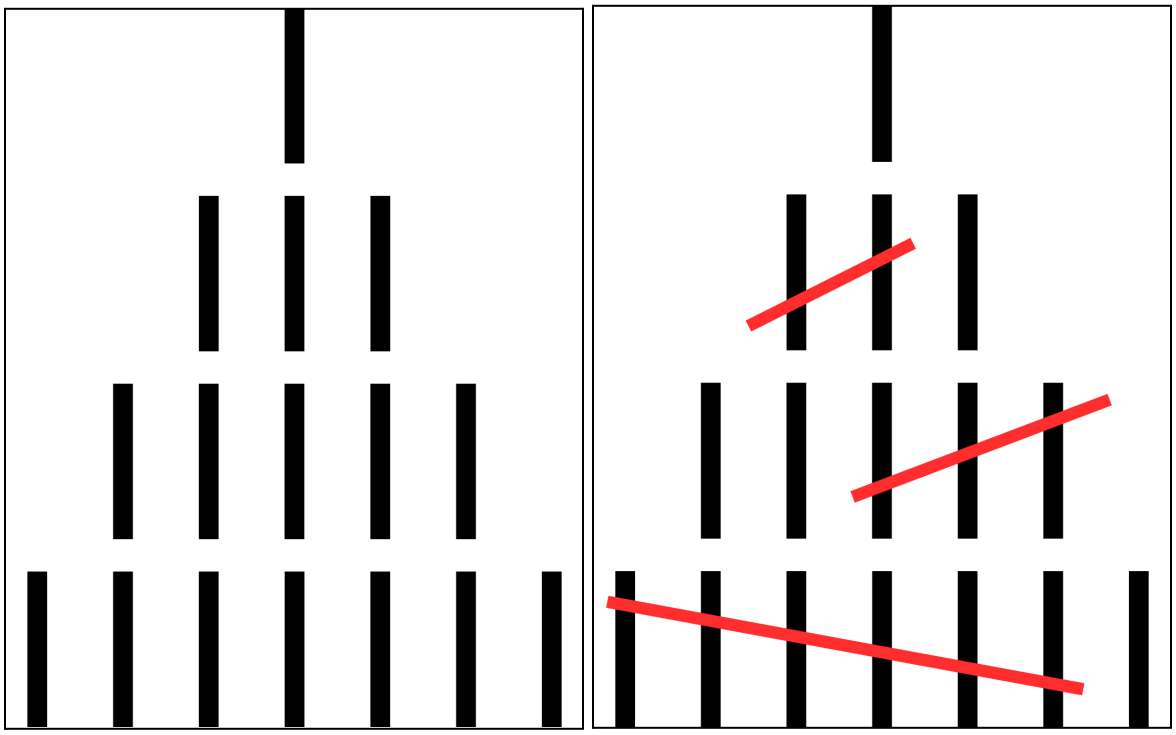
Misere Nim

Zachary Singerman, ZS2661



Misere Nim, translating to “How pitiful!” in Latin, is a game where two players each take turns taking a number of sticks from the formation above to not be the last one to be forced to take a stick. The two players take turns removing one or more sticks from any one row of sticks. Each player can take as many consecutive sticks as they want from one row on their turn, however, they must start on the edge of the row. The player who takes the very last stick from any row loses.

I wanted to do this game because there are so many options to take at each move and they would take a long time with a regular minimax alpha-beta pruning algorithm, learned in my artificial intelligence course. By parallelizing the process, the best move in any situation, or in games like this, the number of sticks to leave your opponent with, can be found much more easily and quickly.

Project overview

My project will parallelize the regular minimax alpha-beta pruning algorithm while also making sure to keep track of which numbers of sticks left on the board lead to a winning game as much as possible.

The minimax algorithm will allow for two computer players to play against each other, each trying to win and each analyzing all of their possible moves. By using alpha beta pruning, we will be able to eliminate the obviously poor moves that lead to losses.

Whenever there is a winning path found, the program will backtrack through the moves that were made and add the number of sticks left after each Max-Player's move to a running list of tuples that count how many times each number has appeared in a win. This is similar to homework 3 that required the number of times a specific word in a file appeared. Anything pruned will consequently not be added to the count of winning numbers.

Algorithm and Approach to Parallelism

Each player always has "how ever many sticks are remaining" number of moves left. So, in a game as featured above, there are immediately 16 different possible paths, each with many paths of their own. By using the minimax algorithm with the alpha beta pruning of the trees, it will be possible to cut down on the number of total possible paths being computed.

In terms of parallelism, I expect to use the Eval Monad and rpar and rseq. These strategies are more easily implemented into a minimax algorithm than the Par Monad functions are. I will also parallelize on the top two levels only as the benefits of alpha-beta pruning would be wasted by parallelizing that deep into the tree.

To maintain my running list of tuples of (Sticks Left, Count), I would have each node return (Minimax Value, Stats), where Stats is a Map Int Int of the (Sticks Left, Count) of each node. This would allow all of the stats and values to funnel back to the top of the tree to understand the best possible moves and the best practices for winning a game of Misere Nim.