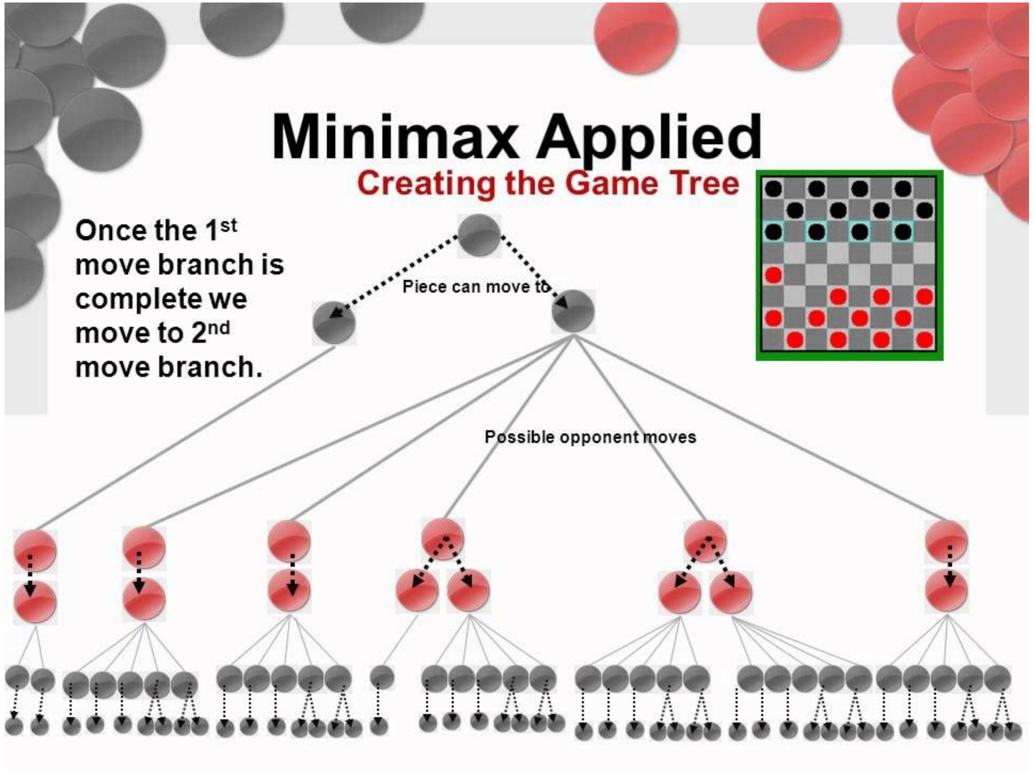
Project Proposal

Henok Bekele
Jacob Anderson

Introduction

We will be developing a checkers game that utilizes a threaded minimaxing search algorithm as the opposing A.I.



Technology

We are implementing a GUI that allows players to control the game

We want to add a difficulty selector that will ultimately change the number of moves considered by the algorithm, making it easier/harder depending on the selector.

Tracking the number of moves considered by the algorithm and display that number to the user

This will be coded in either java or c++

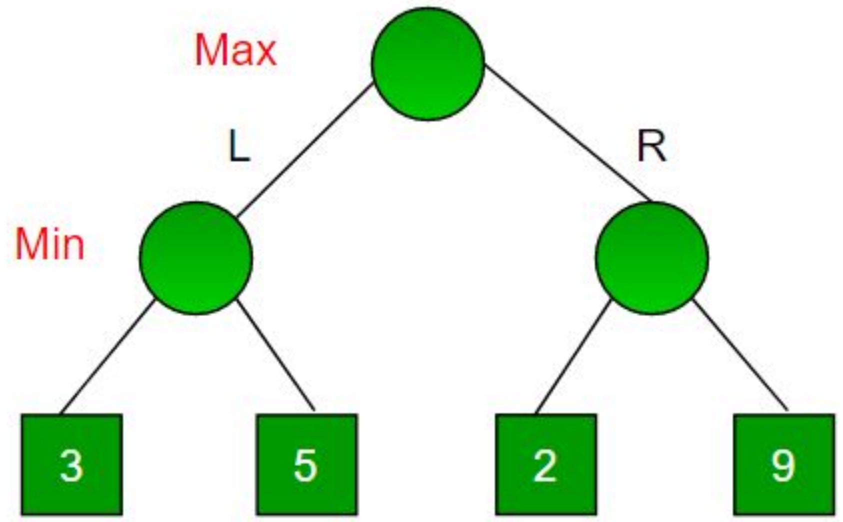
Algorithm/Threading

Every board space will be assigned a value based on where the players checker pieces are

The A.I will be looking to achieve the highest score possible while also considering your next move to be the most minimal value achievable.

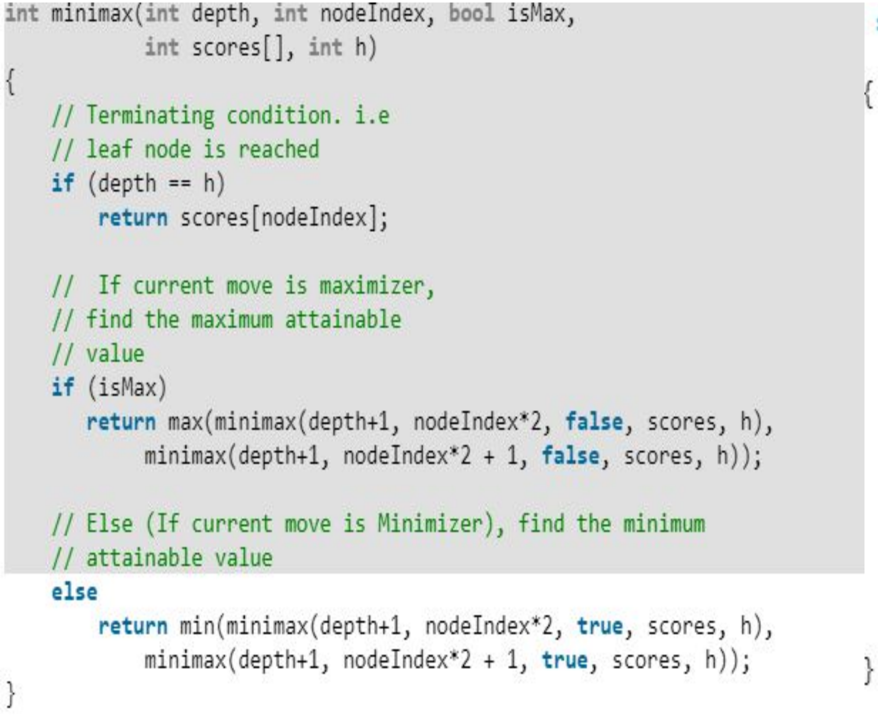
Through threading, we want the algorithm to be considering all of these moves simultaneously

(Potential race condition for tracking the number of moves considered?)



Source: https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/

C++

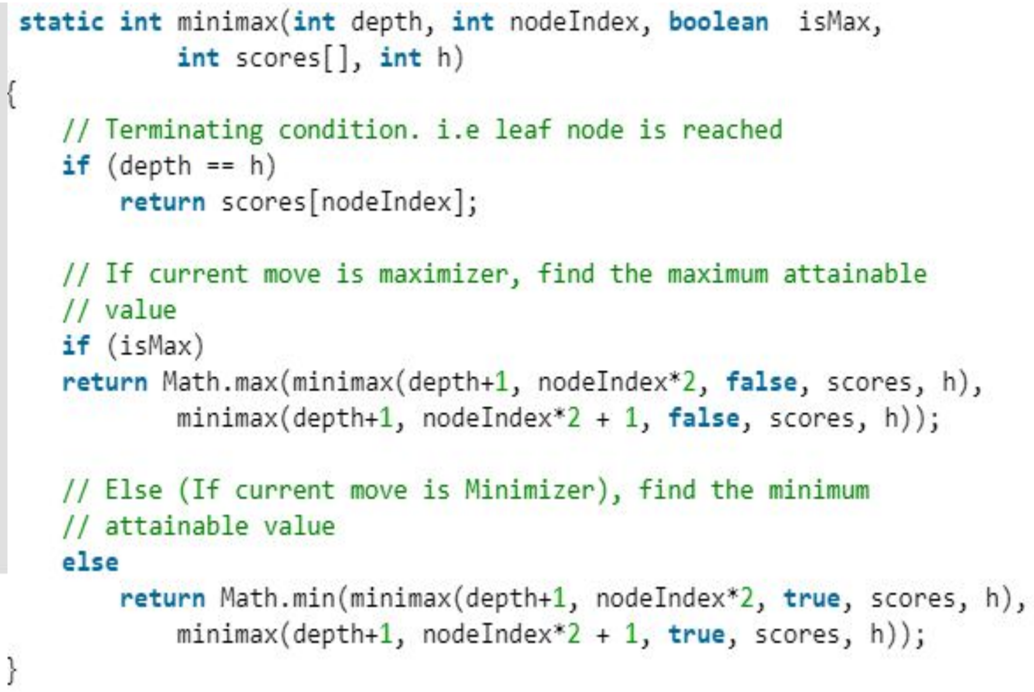


```cpp
int minimax(int depth, int nodeIndex, bool isMax,
            int scores[], int h)
{
    // Terminating condition. i.e
    // leaf node is reached
    if (depth == h)
        return scores[nodeIndex];

    //  If current move is maximizer,
    // find the maximum attainable
    // value
    if (isMax)
       return max(minimax(depth+1, nodeIndex*2, false, scores, h),
            minimax(depth+1, nodeIndex*2 + 1, false, scores, h));

    // Else (If current move is Minimizer), find the minimum
    // attainable value
    else
        return min(minimax(depth+1, nodeIndex*2, true, scores, h),
            minimax(depth+1, nodeIndex*2 + 1, true, scores, h));
}
```

Java



```java
static int minimax(int depth, int nodeIndex, boolean  isMax,
                   int scores[], int h)
{
    // Terminating condition. i.e leaf node is reached
    if (depth == h)
        return scores[nodeIndex];

    // If current move is maximizer, find the maximum attainable
    // value
    if (isMax)
    return Math.max(minimax(depth+1, nodeIndex*2, false, scores, h),
            minimax(depth+1, nodeIndex*2 + 1, false, scores, h));

    // Else (If current move is Minimizer), find the minimum
    // attainable value
    else
        return Math.min(minimax(depth+1, nodeIndex*2, true, scores, h),
            minimax(depth+1, nodeIndex*2 + 1, true, scores, h));
}
```

Goals

We know we did a good job if our code is able to execute and allow the player to interact with the game and play against the computer.

The difficulty selector properly changes the number of moves considered

The number of moves considered displays correctly

The End