CJ Silverman and Ray Tomatsu

CS 420

Project 4

Project 4 Report

Algorithm Pseudocode:

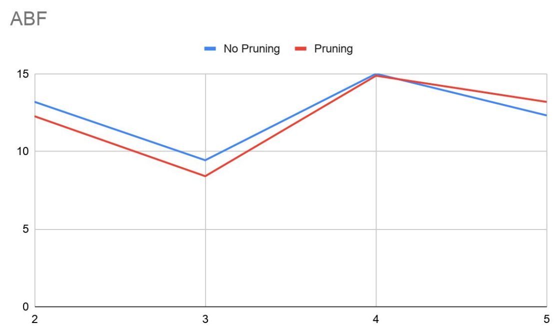
```
determineMove(Board, char, int)
Fill a node of successive moves until the specified int depth
If there is only one one-step node:
    return that board
If there are no one-step nodes:
    Game is over, return original board with a value of 0
If there is more than one one-step node:
    Run determineMove(Board, boolean, int, int, int)


determineMove(BoardNode, boolean, int, int, int)
If it is a leaf, return that value
Otherwise, if it is max (signified by boolean):
    Set moveVal to a large negative number
    For each successor to the BoardNode:
        Run this function and set the returned value to an int
        Set moveVal to the max of moveVal and that value
        Set alpha to the max of moveVal and alpha
        if beta is less than alpha
            return moveVal
    return moveVal if it goes through every successor
If it is min (signified by boolean):
    Set moveVal to a large positive number
    For each successor to the BoardNode:
        Run this function and set the returned value to an int
        Set moveVal to the min of moveVal and that value
        Set beta to the min of moveVal and beta
        if beta is less than alpha
            return moveVal
    return moveVal if it goes through every successor
```

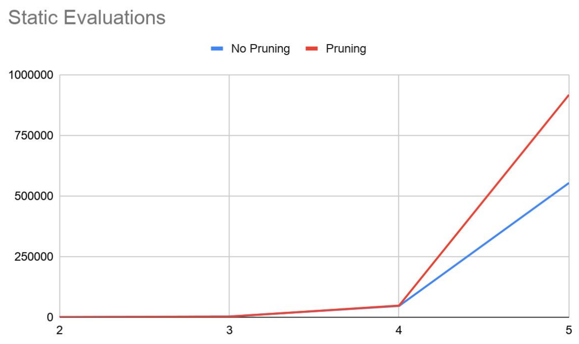
Evaluation:

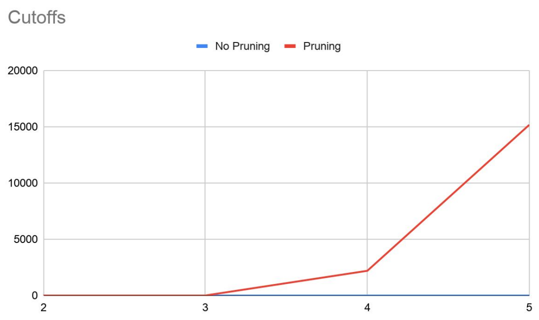
For our evaluation function, the value of a board is based on the number of valid moves it can make. However, different moves are weighted differently, mainly ones that are not mutual. These moves are moves that can be made from the edge of the board in the direction opposite of that edge, as that player can make that move but the other player can not. These moves have a value of 2 in the evaluation function. Jumps, while they take out opponent pieces, still only count for one move, as there is no proof that it will be a benefit in the long run. Aside from that, the program simply searches for all the pieces that match the AI's color, and checks 2 spaces away from it in all directions.

Data:

For this data, we ended up having some problems with our code that we were theoretically not able to fix. First, our leveling format is a bit different, and by the time we realized this, it was far too late to change it, in that the program will automatically work from level 1, meaning level 2 is the lowest it can go. The other possible error is with level 6. When we were running it using BlueJ, BlueJ limits the heap size to a virtual windows machine, so at any level higher than level 5 eventually the heap would fill up. We were unable to successfully run it in another method besides BlueJ, so the code is set up in a way that it should work at any depth, however we were not able to collect data beyond a level of 5.

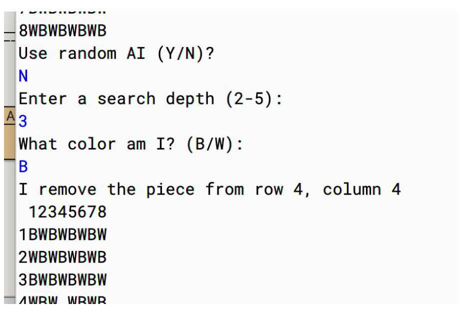




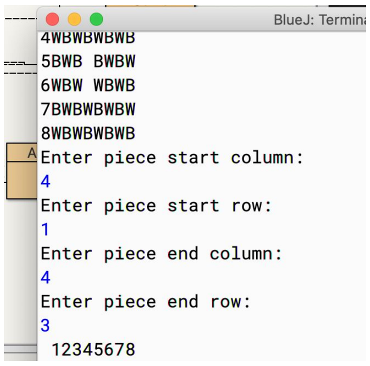


These three graphs show the different information with and without pruning against a random agent. From just glancing at it, the data is not very consistent, and if anything pruning made the function slower. However, this is not necessarily the case. For Average Branching Factor, this checks all of the branches of any trees made at all levels, and then averages the number. The number of branches typically doesn't go higher than 15, but the average ends up having a lot more of these higher values which raise the overall branching factor. For static evaluations, the randomness is what is likely causing this uptick. One of the three values averaged to make this graph was above a million, while another was just above 400000. The interesting thing is that pruning is definitely occurring, at least after a depth of 3 as represented in the cutoffs graph. Therefore the randomness might be hiding the efficiency that is taking place. With those 15000 cutoffs at a depth of five, it's impossible to say how many Nodes skip the recursive processing. However, based on the code and to an extent the data, there has to be efficiency created by the pruning.

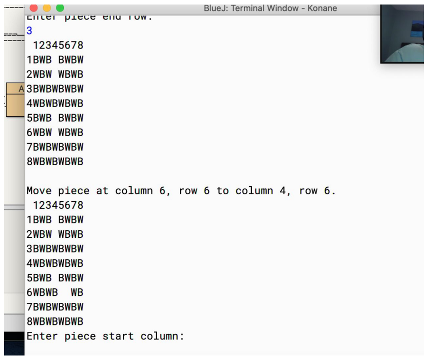
Screenshots of Running:



```
8WBWBWBWB
Use random AI (Y/N)?
N
Enter a search depth (2-5):
3
What color am I? (B/W):
B
I remove the piece from row 4, column 4
 12345678
1BWBWBWBW
2WBWBWBWB
3BWBWBWBW
4WBW WBWB
```



```
4WBWBWBWB
5BWB BWBW
6WBW WBWB
7BWBWBWBW
8WBWBWBWB
Enter piece start column:
4
Enter piece start row:
1
Enter piece end column:
4
Enter piece end row:
3
 12345678
```



```
Enter piece end row:
3
 12345678
1BWB BWBW
2WBW WBWB
3BWBWBWBW
4WBWBWBWB
5BWB BWBW
6WBW WBWB
7BWBWBWBW
8WBWBWBWB

Move piece at column 6, row 6 to column 4, row 6.
 12345678
1BWB BWBW
2WBW WBWB
3BWBWBWBW
4WBWBWBWB
5BWB BWBW
6WBWB  WB
7BWBWBWBW
8WBWBWBWB
Enter piece start column:
```