Chloe Dorward and Seth Markarian
2. a.
starting moves:
- player determines whether to start or defer to AI
- if player starts, player picks a coordinate to remove black piece from, board is updated
- ai then picks random adjacent piece, board is updated
- if player defers, ai picks random legal coordinate to remove black piece from, board is updated
- player then picks adjacent piece, board is updated

player playing:
- possible legal moves are found and given to player
- if no legal moves, end_the_game set to 1
- player chooses a move, board is updated
- current player swaps

ai playing ai:
- possible legal moves are found
- if no legal moves, end_the_game set to 1
- random move selected from legal moves
- board is updated
- current player swaps

ai playing:
- possible legal moves are found
- if no legal moves, end_the_game set to 1
- if random ai:
    - random move selected from legal moves
    - board is updated
    - current player swaps
- if minimax ai:
    - run minimax with current state and depth = 0
- if alpha beta pruning ai:
    - run minimax_ab with current state, alpha be -inf, beta be inf, and depth = 0

Minimax:
- Check depth if equal to certain number (ex: 4):
    - If true increment static eval count
    - Get static evaluation
- Elif its player 0:
    - Increment calls count
    - Loop over all successors
        - Increment branch count
        - Recursively call minimax with current successor state and depth + 1

```
                Return last move of successor
    Else:
            Increment calls count
            Loop over all successors
                Increment branch count
                Recursively call minimax with current successor state and depth + 1
                Return last move of successor

Minimax_ab:
    Check depth if equal to certain number (ex: 4):
        If true increment static eval count
        Get static evaluation
    Elif its player 0:
        Increment calls count
        Loop over all successors
            Increment branch count
            Recursively call minimax_ab with current successor state, new alpha,
            new beta and depth + 1
            Check if new value is greater than alpha
                If so, assign best move to the successor's last move
            Check if new alpha is greater than or equal to current beta
                If so, increment cutoff count
                Return the beta and best move as a tuple
        Return the alpha and current best move
    Else:
        Increment calls count
        Loop over all successors
            Increment branch count
            Recursively call minimax_ab with current successor state, new alpha,
            new beta and depth + 1
            Check if new value is less than beta
                If so, assign best move to the successor's last move
            Check if new beta is less than or equal to current alpha
                If so, increment cutoff count
                Return the beta and best move as a tuple
        Return the beta and current best move
```

b. static evaluation:

Our static evaluation function returns how many more moves a player has left compared to the artificial intelligence. If the player has more moves than the ai left, then that is positive, and if the player has fewer moves than the ai left, then it is negative. Since this evaluation gets minimized, it means the minimax returns the option which yields the option where the ai is left with more moves. If there is no option where ai has more moves, it yields the option where the player has the least amount more moves than the ai.

code of function:

```
def static_evaluation(self):
    player_moves = self.find_moves(0)
    ai_moves = self.find_moves(1)
    if ai_moves == 0:
        return float("inf")
    if player_moves == 0:
        return float("-inf")
    return len(player_moves) - len(ai_moves)
```

c. Theoretically with Alpha Beta pruning, fewer nodes are explored and time is saved.
Based on our data, however, more static evaluations were done using Alpha Beta pruning, and the branching factor was higher for both depth two and depth 6.
Minimax:

Depth	Static Eval	Avg. Branching	Number Cutoffs
2	452	4.990825	N/A
4	23414	4.92978330	N/A
6	65081	2.5349517	N/A

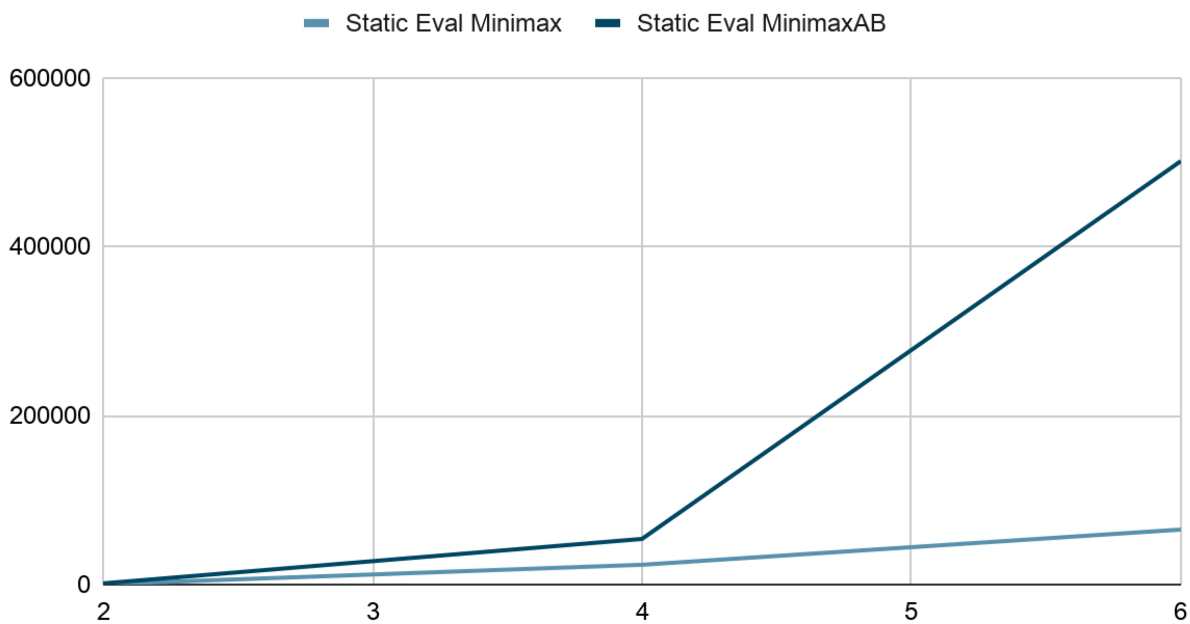
Minimax Alpha Beta:

Depth	Static Eval	Avg. Branching	Number Cutoffs
2	1365	7.64851485	350
4	54018	4.57355	11945
6	501348	3.73078829675	141494

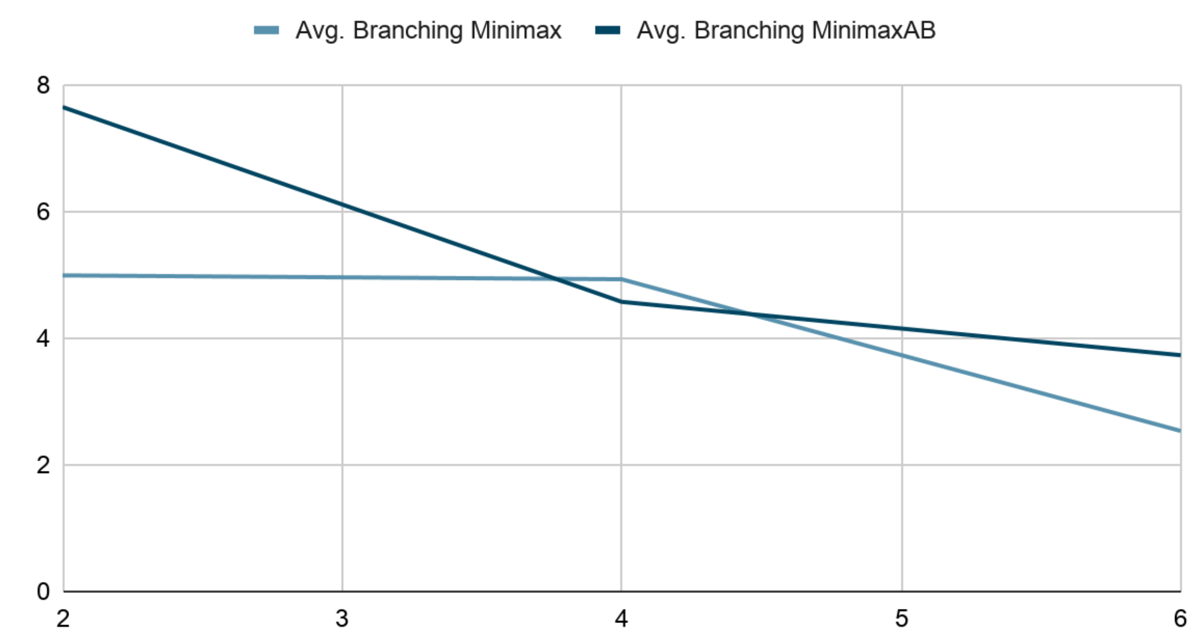
Minimax and Minimax Alpha Beta:

Depth	Static Eval Minimax	Static Eval MinimaxAB	Avg. Branching Minimax	Avg. Branching MinimaxAB
2	452	1365	4.990825	7.64851485
4	23414	54018	4.92978330	4.57355
6	65081	501348	2.5349517	3.73078829675

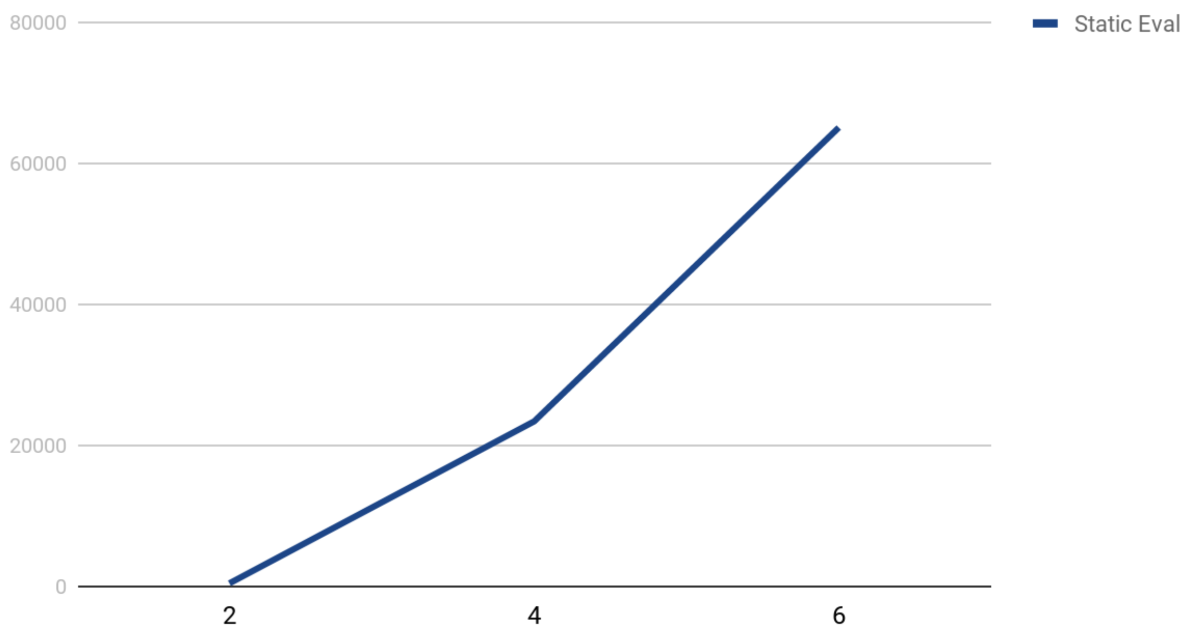
Static Eval, Minimax vs MinimaxAB



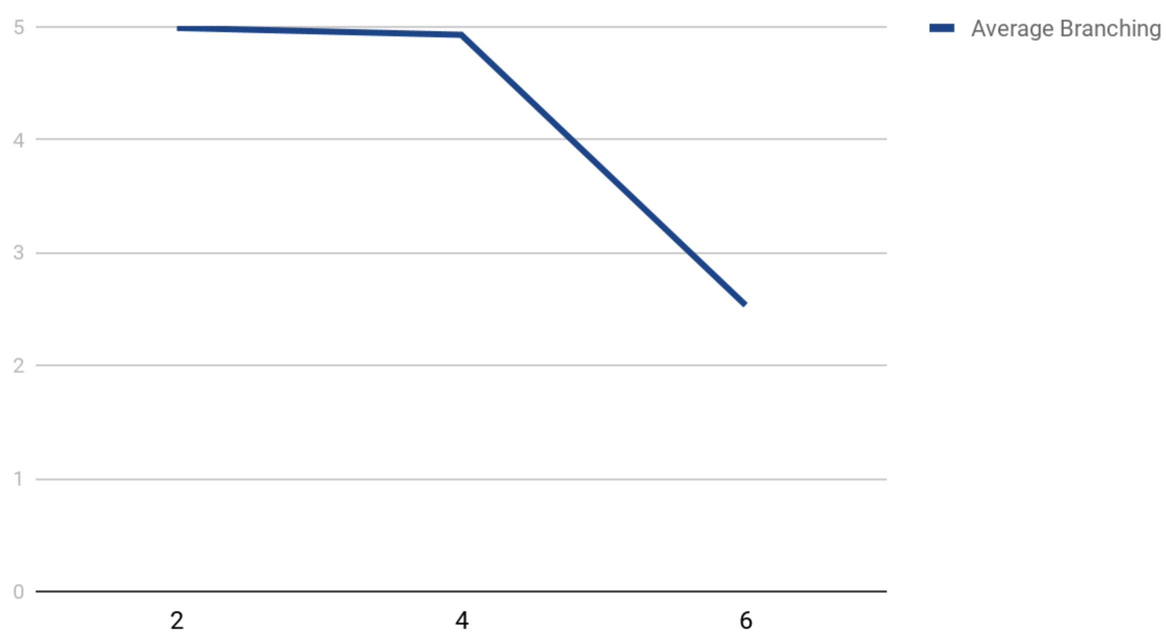
Avg. Branching, Minimax vs MinimaxAB



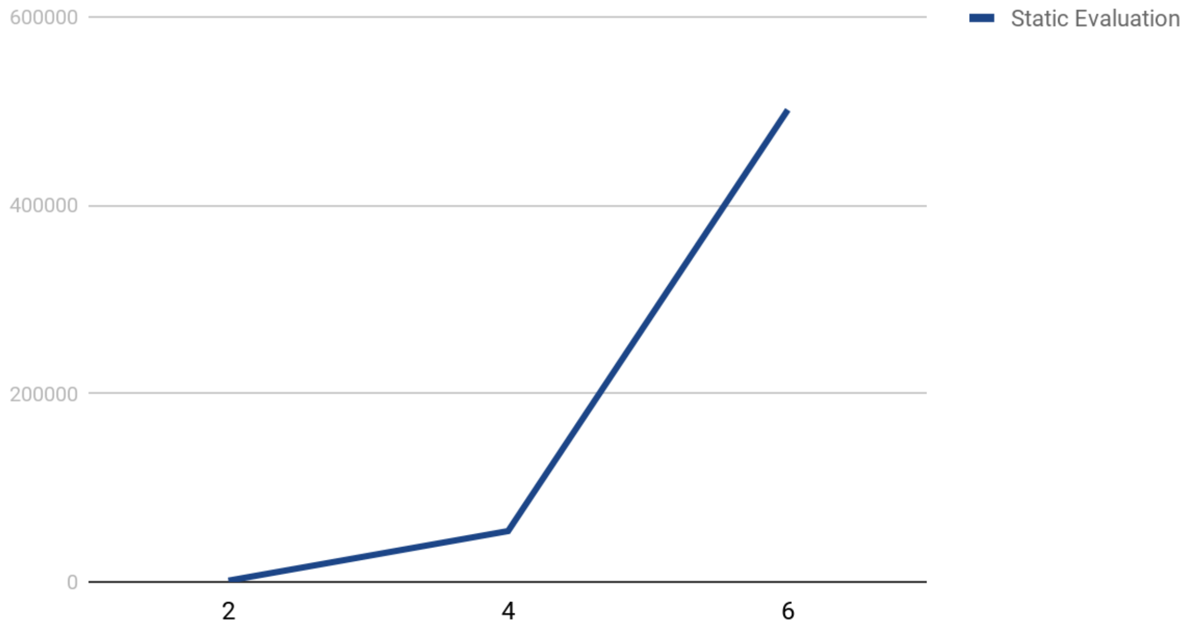
Minimax, Static Evaluation vs. Depth



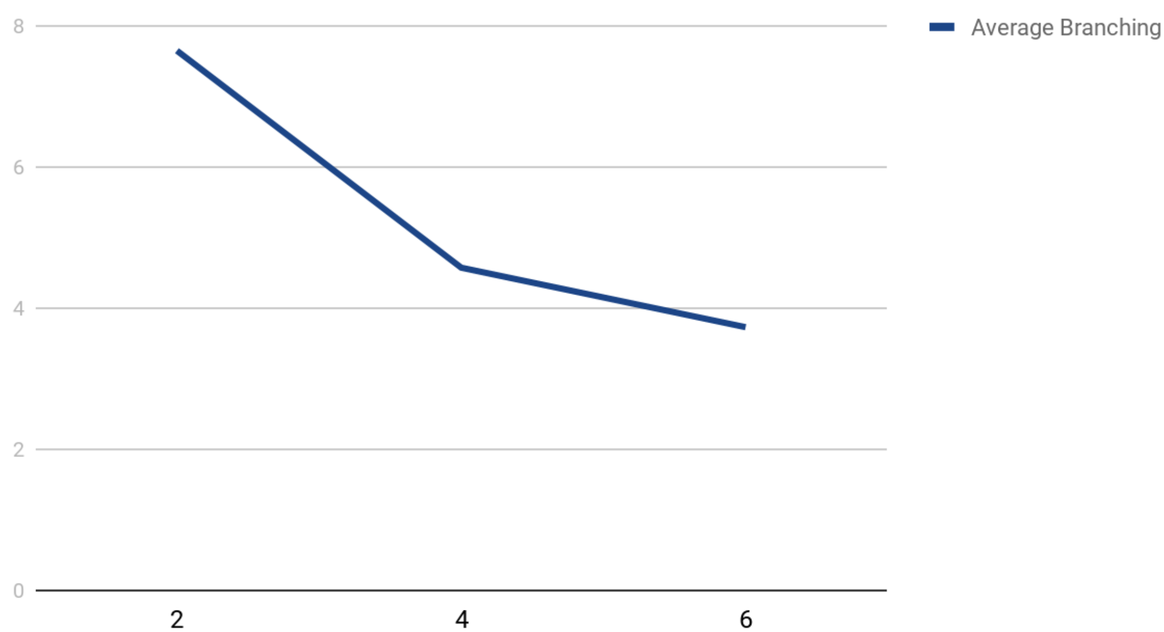
Minimax, Average Branching vs. Depth

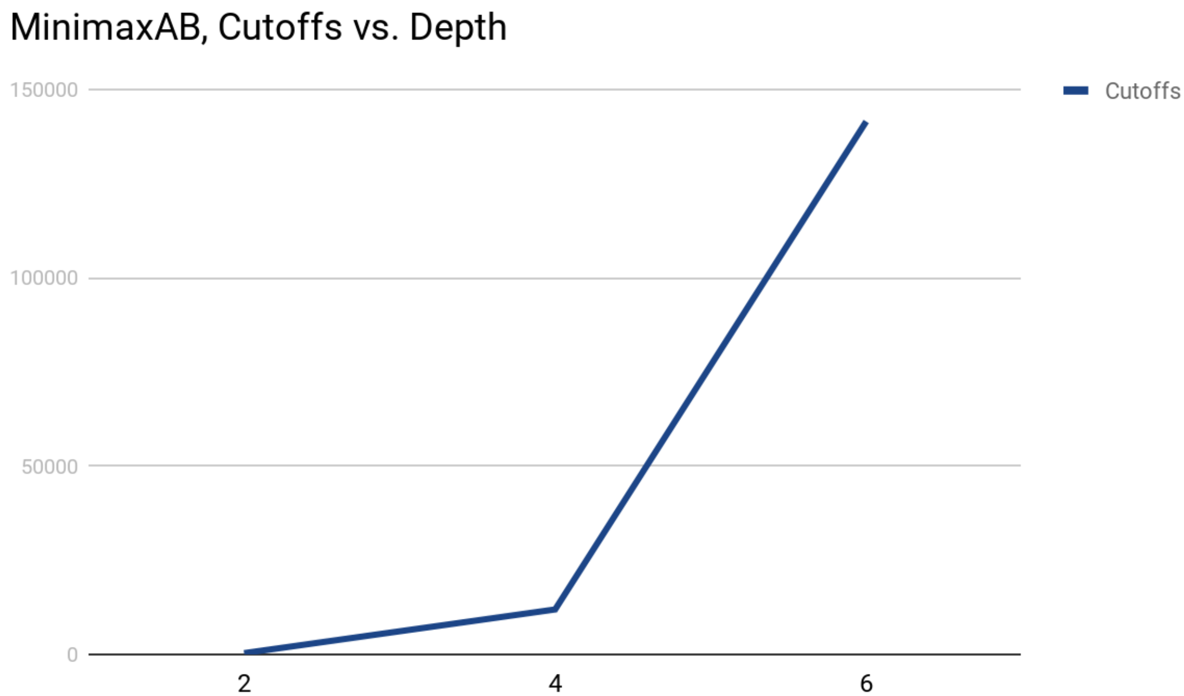


MinimaxAB, Static Evaluations vs. Depth



MinimaxAB, Average Branching vs. Depth




MinimaxAB, Cutoffs vs. Depth
150000
100000
50000
0
2
4
6
Cutoffs