CS 214: Chess Engine

Mentor : Chirumamilla Bhargav
Animesh Jain 220120005
Gurjaipal Singh 220120008
Tanishque Soni 220120025
Upakramanika Bisnu 225100039
Mahamkali Sri Phaneeswar 225100023
Shinde Manav Rajendra 225100032

April 10, 2024

Table of contents

Problem Statement

Problem

Chess is an abstract strategy game that involves no hidden information the entire 64 squares are observable by both player at all time and no elements of chance as the move chosen by the player get reflected on the board as it is.
The idea of creating a chess-playing machine dates to the 18th century. Since the advent of the digital computer in the 1950s, chess enthusiasts, computer engineers, and computer scientists have built, with increasing degrees of seriousness and success, chess-playing machines and computer programs.
Via this project we will have build our own chess engine which can play chess

Existing Methods

Various techniques have been used to create a chess engine which includes

1. Alpha-Beta Minimax Search with deepblue and stockfish
2. Monte Carlo Tree Search with Komodo Monte Carlo
3. Reinforcement Learning with alphazero

Evaluation Function

```
piece_value = {'p': -1, 'r': -5, 'n': -3, 'b': -3.125, 'q': -9, 'k': -10000,
               'P': 1,  'R': 5,  'N': 3,  'B': 3.125,  'Q': 9,  'K': 10000}
```

Our value assignment

Symbol					
Piece	pawn	knight	bishop	rook	queen
Value	1	3	3	5	9

Most common assignments

Value used by modern chess engines

					Source	Date	Comment
4	3.5	7	13.5	4	used by a computer	1992	Two bishops are worth more.[23]
3.05	3.33	5.63	9.5		AlphaZero	2020	[1]

Game phase	 pawn	 knight	 bishop	 bishop pair bonus	 first rook	 second rook	 queen	 second queen	Comments
Middlegame	0.8	3.2	3.3	+0.3	4.7	4.5	–	–	(both sides have a queen)
Threshold	0.9	3.2	3.3	+0.4	4.8	4.9	9.4	8.7	(one queen vs. zero, or two queens vs. one)
Endgame	1.0	3.2	3.3	+0.5	5.3	5.0	–	–	(no queens)

Temporal valuation of the pieces

Evaluation Function

```
[ -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0],
[ -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0],
[ -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0],
[ -3.0, -4.0, -4.0, -5.0, -5.0, -4.0, -4.0, -3.0],
[ -2.0, -3.0, -3.0, -4.0, -4.0, -3.0, -3.0, -2.0],
[ -1.0, -2.0, -2.0, -2.0, -2.0, -2.0, -2.0, -1.0],
[  2.0,  2.0,  0.0,  0.0,  0.0,  0.0,  2.0,  2.0 ],
[  2.0,  3.0,  1.0,  0.0,  0.0,  1.0,  3.0,  2.0 ]
```

```
[ -2.0, -1.0, -1.0, -0.5, -0.5, -1.0, -1.0, -2.0],
[ -1.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -1.0],
[ -1.0,  0.0,  0.5,  0.5,  0.5,  0.5,  0.0, -1.0],
[ -0.5,  0.0,  0.5,  0.5,  0.5,  0.5,  0.0, -0.5],
[  0.0,  0.0,  0.5,  0.5,  0.5,  0.5,  0.0, -0.5],
[ -1.0,  0.5,  0.5,  0.5,  0.5,  0.5,  0.0, -1.0],
[ -1.0,  0.0,  0.5,  0.0,  0.0,  0.0,  0.0, -1.0],
[ -2.0, -1.0, -1.0, -0.5, -0.5, -1.0, -1.0, -2.0]
```

```
[-5.0, -4.0, -3.0, -3.0, -3.0, -3.0, -4.0, -5.0],
[-4.0, -2.0,  0.0,  0.0,  0.0,  0.0, -2.0, -4.0],
[-3.0,  0.0,  1.0,  1.5,  1.5,  1.0,  0.0, -3.0],
[-3.0,  0.5,  1.5,  2.0,  2.0,  1.5,  0.5, -3.0],
[-3.0,  0.0,  1.5,  2.0,  2.0,  1.5,  0.0, -3.0],
[-3.0,  0.5,  1.0,  1.5,  1.5,  1.0,  0.5, -3.0],
[-4.0, -2.0,  0.0,  0.5,  0.5,  0.0, -2.0, -4.0],
[-5.0, -4.0, -3.0, -3.0, -3.0, -3.0, -4.0, -5.0]
```

```
[  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0],
[  0.5,  1.0,  1.0,  1.0,  1.0,  1.0,  1.0,  0.5],
[ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5],
[ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5],
[ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5],
[ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5],
[ -0.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -0.5],
[  0.0,   0.0, 0.0,  0.5,  0.5,  0.0,  0.0,  0.0]
```

```
[ -2.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -2.0],
[ -1.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0, -1.0],
[ -1.0,  0.0,  0.5,  1.0,  1.0,  0.5,  0.0, -1.0],
[ -1.0,  0.5,  0.5,  1.0,  1.0,  0.5,  0.5, -1.0],
[ -1.0,  0.0,  1.0,  1.0,  1.0,  1.0,  0.0, -1.0],
[ -1.0,  1.0,  1.0,  1.0,  1.0,  1.0,  1.0, -1.0],
[ -1.0,  0.5,  0.0,  0.0,  0.0,  0.0,  0.5, -1.0],
[ -2.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -2.0]
```

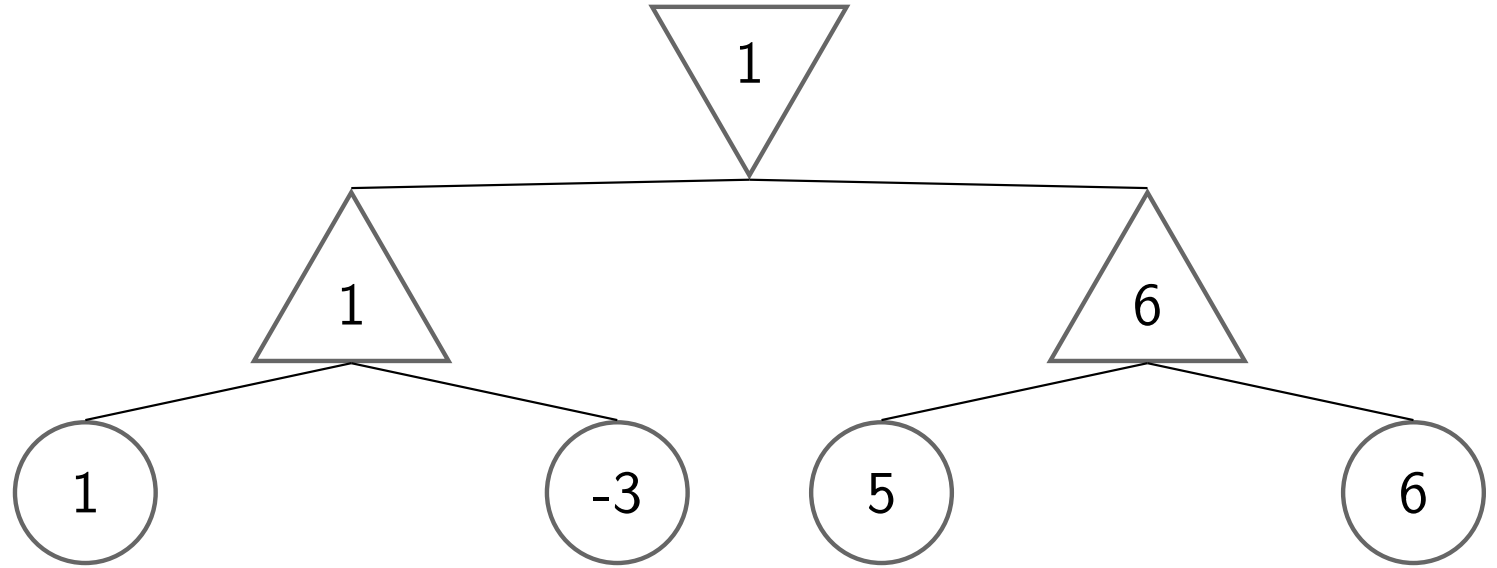
```
[0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0],
[5.0,  5.0,  5.0,  5.0,  5.0,  5.0,  5.0,  5.0],
[1.0,  1.0,  2.0,  3.0,  3.0,  2.0,  1.0,  1.0],
[0.5,  0.5,  1.0,  2.5,  2.5,  1.0,  0.5,  0.5],
[0.0,  0.0,  0.0,  2.0,  2.0,  0.0,  0.0,  0.0],
[0.5, -0.5, -1.0,  0.0,  0.0, -1.0, -0.5,  0.5],
[0.5,  1.0, 1.0,  -2.0, -2.0,  1.0,  1.0,  0.5],
[0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0]
```

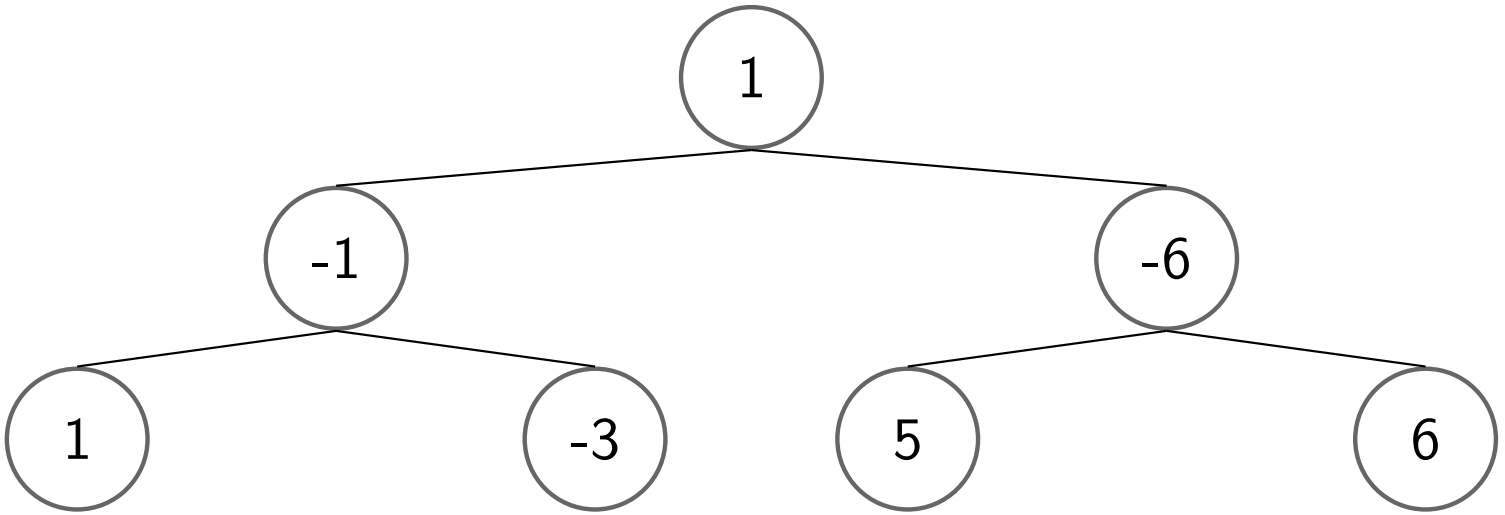
Spatial valuation of other pieces

Negamax

For our project we have used Alpha-Beta Minimax (Negamax) Search with Quiecence Search. Primary for its simplicity and speed on non-dedicated hardware

Minimax vs Negamax

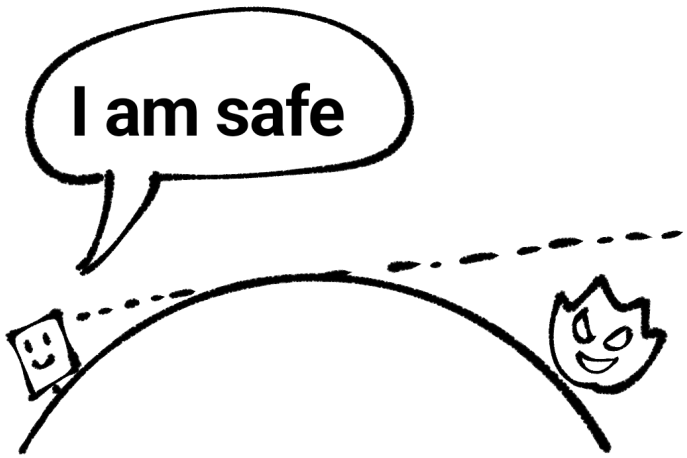




Quiescence search

Quiescence search

Quiescence search cont.




```python
def quiesce(self, alpha, beta):

    stand_pat = self.eval_function()

    if (stand_pat >= beta):
        return beta
    if (alpha < stand_pat):
        alpha = stand_pat

    for move in self.AI_Board_ref.legal_moves:
        if self.AI_Board_ref.is_capture(move):
            self.AI_Board_ref.push(move)
            score = -self.quiesce(-beta, -alpha)
            self.AI_Board_ref.pop()
            if (score >= beta):
                return beta
            if (score > alpha):
                alpha = score

    return alpha
```

Results

- With use of Quiescence search we can reach depths of up to **27** in reasonable amount of time
- We have also attached the game we played with this bot in the report against the chess.com engine at various levels of difficulty from ELO level 800 to 2000 and it performs very well against engine upto 1400 uptill mid game but it is not very good in end games

Future

Further progress can be made by improving the evaluation function and use of more specialized and advanced algorithms

As well as the same AI can implemented for more and more variants of chess such as 4 player chess, 3D chess and more just by changing the evaluation and move generation functions

Contribution of Authors

- Animesh : Wrote Evaluation Function and Did Integration of GUI and AI
- Upakramanika : Wrote the AI Engine with Minimax(Negamax) and Quiecence Search also did testing
- Manav : Write the code to look for checks and en passant, testing
- Gurjaipal : Did GUI,in particular Wrote Function that calculates all possible(valid)moves a specific piece can move on the board,including making the pieces move on the board, testing
- Tanishque : Did Base GUI Board.py, dragger.py , made ReadMe.md file
- Phaneeswar : Did GUI, mainly sound and themes, testing