Faculty of Informatics and Computer Science

Logic and Artificial Intelligence

Online Chess Agent

Final Project Documentation

Presented by:

Name	ID	Major
Abdelrahman Essam	155295	SE
Marawan Mohamed	152282	CS

BUE, El-Sherouk City, Cairo, Egypt

04-2019

Prt 1: Chess Agent.

Problem description and background:

Chess is a board turn based game played by two players. The chess grid consists of 64 squares, each player side consist of 8 of these squares. In addition, each player starts with 16 pieces which consist of the following 8 pawns, 2 nights, 2 bishops, 2 rooks, 1 queen and a king and each one have different moves and different priorities so the queen is much more important than a pawn . The goal of the game is to threat the other player's king to a move which cannot be stopped ("CHECKMATE"). Finally, if CHECKMATE state is reached the game ends and by eating the opponent's chess pieces it makes his chance of losing higher. Moreover, the two players have different pieces one is "White" and the other is "Black", usually the white player starts the game. Furthermore, the game has rules to the movements of pieces, so each type of these pieces has restricted movement specified by the rules. In addition, there are also rules about taking opponent's pieces of the board.

Project Summary: Main components.

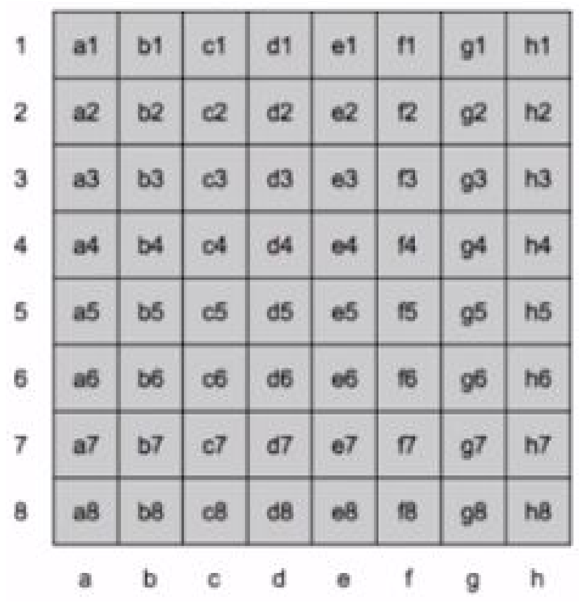
Agent:

The Project is an Agent That play a reasonable game of chess, providing it with data set for example what is it’s priorities and goals, which is the most valuable pieces for it and for the opponent also some rules like en passant, castle moves etc.. and the agent is responsible to find the best move using the min-max algorithm.

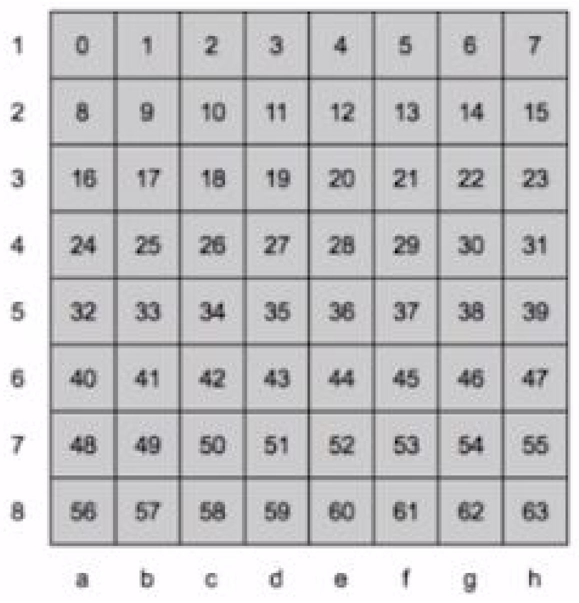
Move Gentation

The Agent Use the Move generation function which is a basic part of a chess agent with many variations concerning a generator to loop over moves inside the search routine, which depends on the board representation.

Game Board



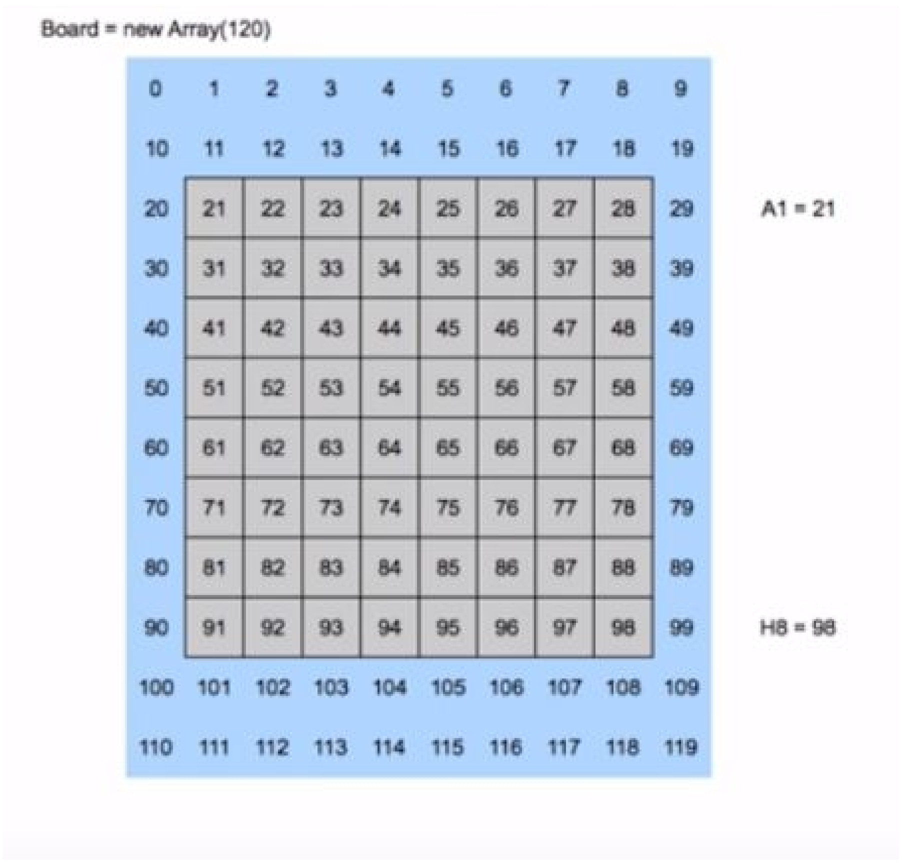
1	a1	b1	c1	d1	e1	f1	g1	h1
2	a2	b2	c2	d2	e2	f2	g2	h2
3	a3	b3	c3	d3	e3	f3	g3	h3
4	a4	b4	c4	d4	e4	f4	g4	h4
5	a5	b5	c5	d5	e5	f5	g5	h5
6	a6	b6	c6	d6	e6	f6	g6	h6
7	a7	b7	c7	d7	e7	f7	g7	h7
8	a8	b8	c8	d8	e8	f8	g8	h8
	a	b	c	d	e	f	g	h



1	0	1	2	3	4	5	6	7
2	8	9	10	11	12	13	14	15
3	16	17	18	19	20	21	22	23
4	24	25	26	27	28	29	30	31
5	32	33	34	35	36	37	38	39
6	40	41	42	43	44	45	46	47
7	48	49	50	51	52	53	54	55
8	56	57	58	59	60	61	62	63
	a	b	c	d	e	f	g	h

Normally this how the game board looks like 64 square represented by files and ranks a to h and 1 to 8, but this will not be the board included in the chess agent dataset.

After searching for the best presentation for the chess game board this was founded.


Board = new Array(120)

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99
100	101	102	103	104	105	106	107	108	109
110	111	112	113	114	115	116	117	118	119

A1 = 21

H8 = 98

A 120 squares board that will benefit the agent implementation the surrounding squares (in blue) will be indication if a piece is going off board moreover, there is a function which convert the 120 squares table indexes to 64 one And for the reason for the to rows at the top and bottom.

is the Knight’s moves so if it’s at the top or bottom square of the board some of its moves will be exceeding the offboard squares.

Move encoding

Encoding Moves inside the chess agent refers to both game records or game notation, and search related generation. During generation, moves are stored inside move lists, and best moves are the ones high inside the search.

Fen position Parsing.

Forsyth–Edwards Notation is a way for describing the board and the pieces on it which compose of 6 sections, each slash indicate a new rank however the letters indicates the piece's type and color as upper case is white and lower case is black. After passing all the rank there is the side w for white and b for black then the castling permission followed by the en passant and fifty mpve and number of plys.

And here is an example for the starting game board using FEN String.

rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1

DataSet.

The closest thing to dataset included in the agent are as follow:

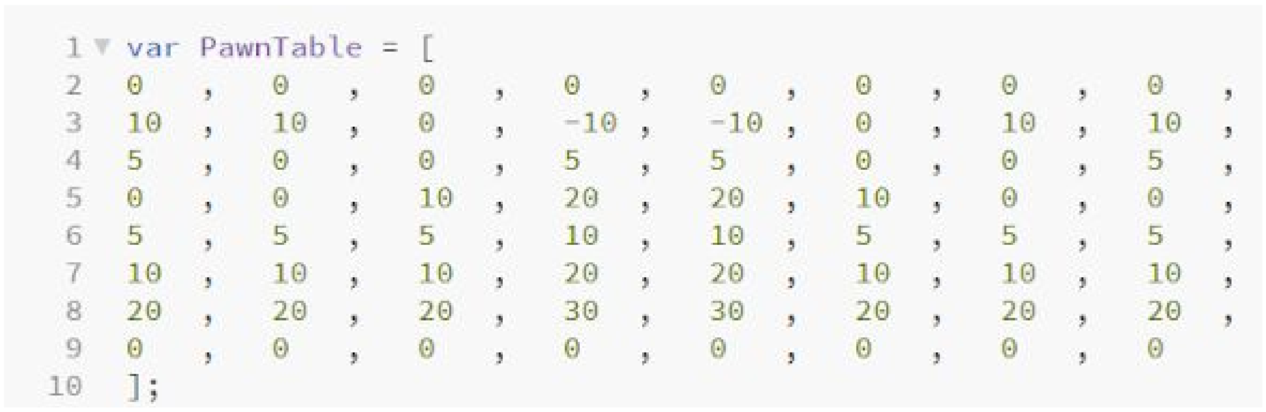
The game board and it’s positions.

Side indication whether it’s white or black

The pieces values as the higher the values goes the higher its priority is and in reverse the higher the the value of the victim is to be attacked.

The best position for a piece ignoring the fact of if it’s been attacked or if there is a piece to be captured.

For example:



```
var PawnTable = [
0   ,   0   ,   0   ,   0   ,   0   ,   0   ,   0   ,   0   ,
10  ,   10  ,   0   ,   -10 ,   -10 ,   0   ,   10  ,   10  ,
5   ,   0   ,   0   ,   5   ,   5   ,   0   ,   0   ,   5   ,
0   ,   0   ,   10  ,   20  ,   20  ,   10  ,   0   ,   0   ,
5   ,   5   ,   5   ,   10  ,   10  ,   5   ,   5   ,   5   ,
10  ,   10  ,   10  ,   20  ,   20  ,   10  ,   10  ,   10  ,
20  ,   20  ,   20  ,   30  ,   30  ,   20  ,   20  ,   20  ,
0   ,   0   ,   0   ,   0   ,   0   ,   0   ,   0   ,   0
];
```

A screen shot form the code indication what is the value of each move for the white Pawn and there is an array for each piece from each color.

Special Rules

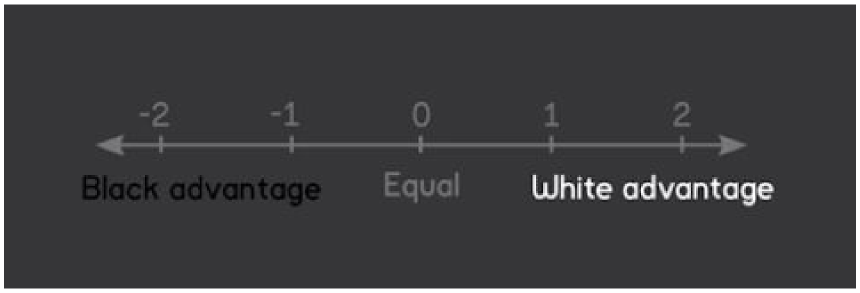
1. Castle move
2. En passant
3. Fifty moves draw
4. 3-fold repetition.
5. Promotion (only to queen)

The Moves available moves for each piece.

Algorithm used Min-max:

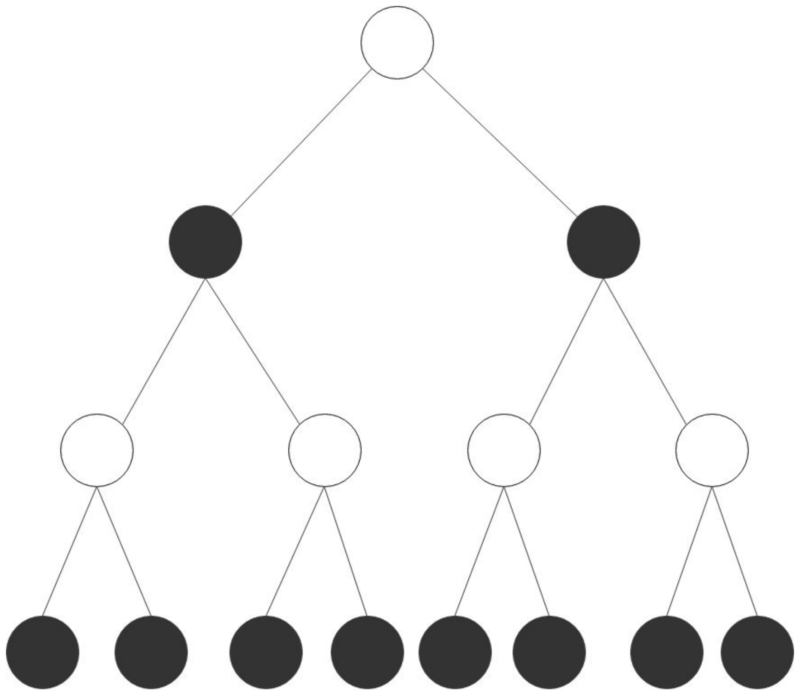
As chess is a turn based games which needs to search for algorithm that looks ahead for possible future positions before deciding what move it wants to make in the current position. However, min-max provide that by walking through the tree trying to get the best score for your position.

Why does it called min-max?

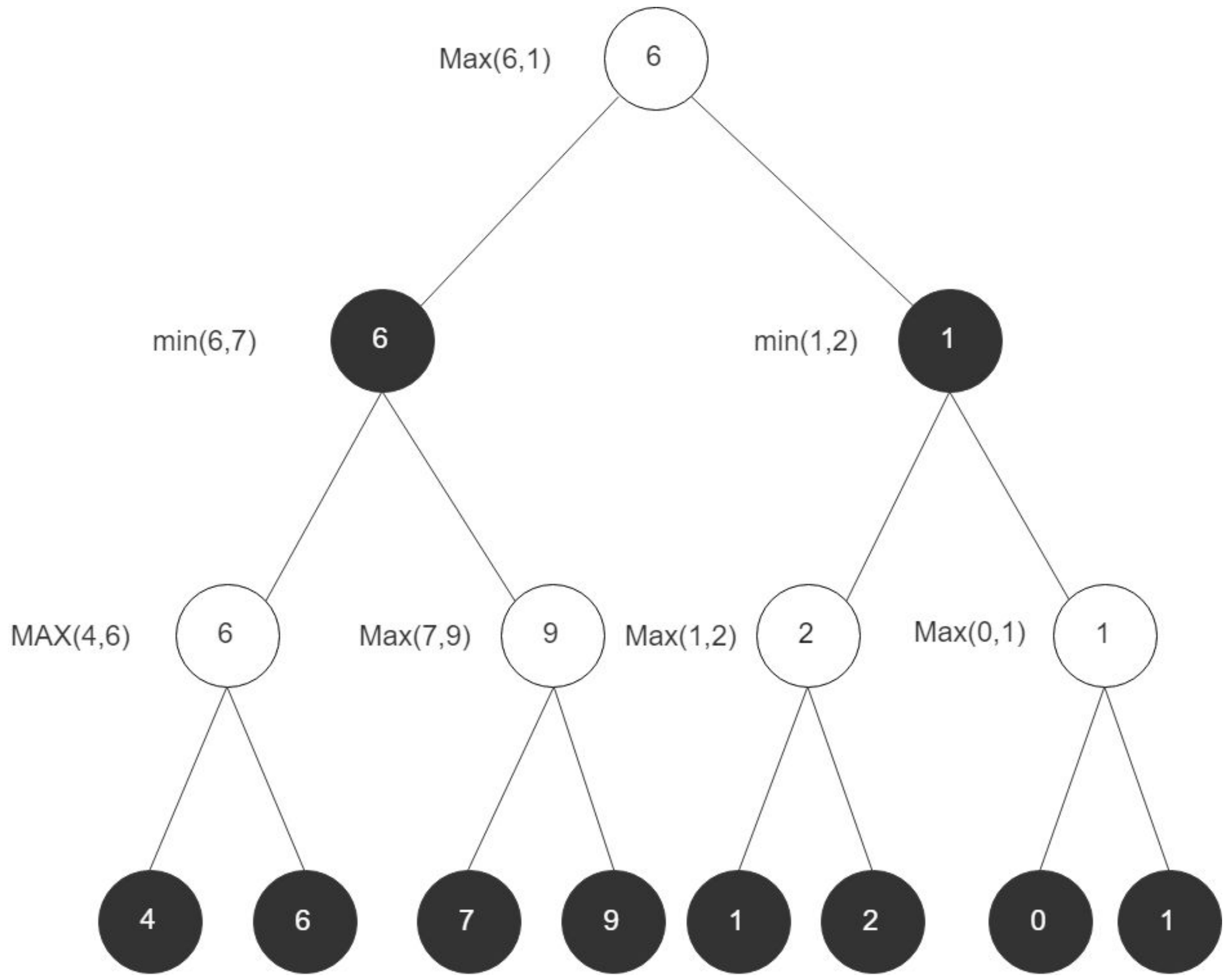


So white will be the maximizer because he wants to get the biggest score possible and black would be the minimizer because he wants to make the score for white as low as possible.

Example:



As shown in the figure above looking at min-max in terms of tree and each side (BLACK and WHITE) has tow legal moves in a given position and the positions going to a depth of three.
So white is the maximizer and white is the minimizer.



As shown as the second figure which represent walking through this tree in min-max fashion starting from the bottom row that the white side(maximizer) choose from the to whighted move then the black do the opposite as a minimizer choosing the minimum weight ending up with 6 at the top of the tree.

And this is how the min-max algorithm work so it can be concluded that this is a brute-force approach as it searches every move available in the game tree and that’s needed in game like chess. So at the ai turn the agent will check for every possible move and choose the one which will lead to move forward the game in his advantage and prevent the other player from wining.

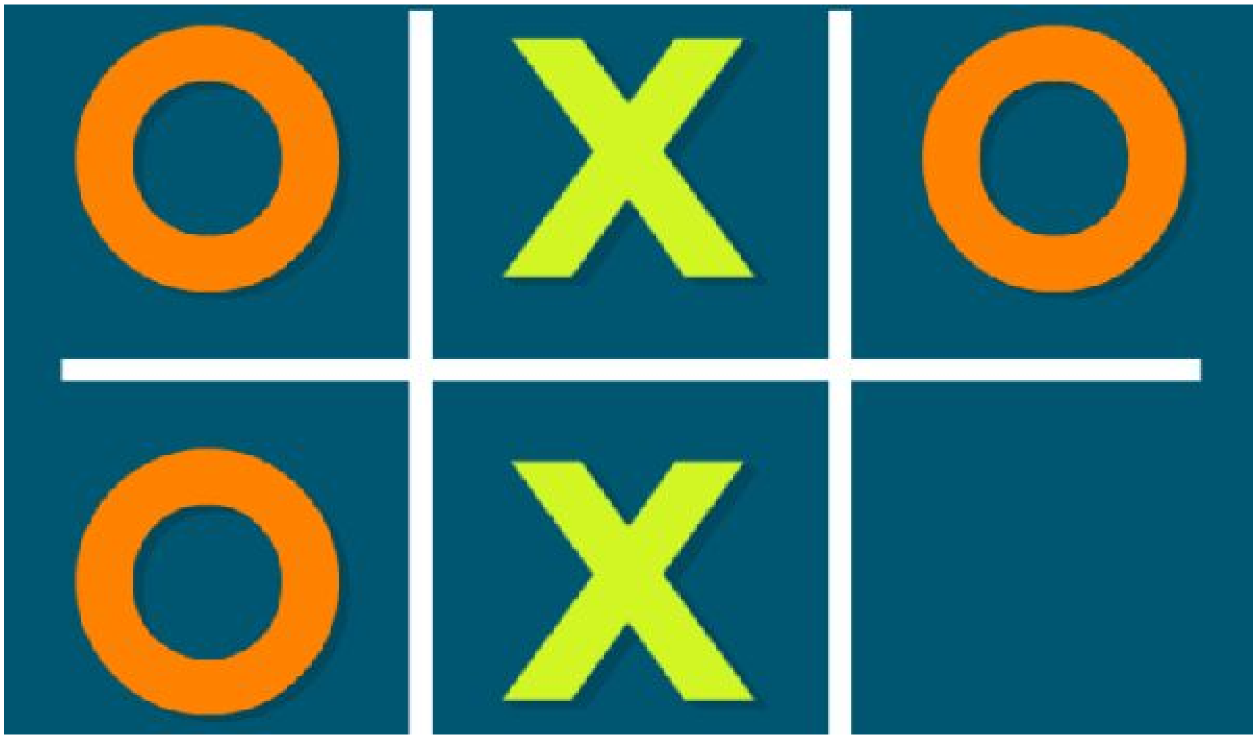
Alpha Beta Filter:

We can think of alpha beta as way for altering the min-max algorithm to be more efficient and prevent wasting of time by eliminating branches that the agent will detect that there is no need to search in them.

Nega-max

Is the combination of min and max together and the difference is rather from returning the score form whites point of view the maximizers you always return it from the point of view of the side to move and then the result of the function is negated so it fits the other POV here is an example (not the exact one added to the code.)

Prt 2: Tic Tac Toe: agent.



Tic tac toe is another turn based game played by two player each one have a symbol x or o
And the player who gets three symbol in one row horizontally vertically or diagonally wins the game.

The same algorithm was used but not the (nega max) to search for the best move in each turn but in different environments.

Software and Tools used to accomplish this project:

1. **Programing Languages:**

   This agent will be developed in JavaScript as it's intended for this agent to be published and be able to work online however, HTML and CSS in addition to bootstrap frameWork will be also used to handle the project design and user interface.

2. **IDE**

   For integrated development environment, we chose Atom text editor.

3. **GitHub**

   For publishing the agent online.

Resources Found:

1. http://web.archive.org/web/20080216031116/http://www.seanet.com/~brucemo/topics/topics.htm
2. https://www.chessprogramming.org/Alpha-Beta
3. https://www.youtube.com/watch?v=EI6vb82I9-o&t=154s
4. http://web.archive.org/web/20080216031116/http://www.seanet.com/~brucemo/topics/topics.htm
5. https://www.chessprogramming.org/Main_Page

Project Link: https://abdelrahmanessama.github.io/index.html