Checkmate with King and Queen

School Name: Pinon Elementary School

Team members grade(s) in school: 5th, both members

Team Number: Pinon-1

Team Members:

Shaun Kilde: amanikilde@gmail.com

Daniel Kim: daniel360kim@gmail.com

Area of Science: Computer Science

Computer Language: Python

Mentor: Scott Kilde

Abstract

This project was a fun and great challenge for us as 5th graders to learn a new computer language that will be useful for us in the future. Our problem is not unknown in the world but for us is new and unknown on how a computer solves chess problems. Our work in this was fun and great.

Our future is to fully get the model working and then to extend it into “machine learning” where the model will use past games to make better solutions to current board configurations.

Problem Statement

Computerized chess, although originally known for human vs. their own computer or phone on an app, there are many different supercomputers used to play the world's best chess players (Grand Masters). These supercomputers, such as Deep Blue[1] are used to discover new medical drugs, to do extensive financial remodeling to analyze trends, handle broad database searches, and perform immense calculations needed in many fields of science. Modern supercomputers such as the current most powerful supercomputer Tianhe-2 , has the performance of 33,860 trillion calculations per second. In attempting this project, our group is undertaking the research and simulation of using algorithms on a chessboard and making a checkmate.

Our target is to program the computer using the algorithms and creating a checkmate going against itself, with a White King and a White Queen, vs a Black King. We hope that the computer simulation will learn from the past game experiences and get better and better at creating an endgame. The computer will push the King into both and edge or a corner and checkmate using the checkmate combinations. The checkmate combinations are where the Black King is, and where the White King and White Queen are when there is a checkmate. For example, if the Black King is at h5 on the chessboard, the White King can be at f4, f5, or f6, and the White Queen can only be at g5 on the chessboard. This will allow the computer to make a checkmate and win the game.

Methods Used

Our initial work was to look at the history of computerized chess while also beginning to learn Python program; which neither of us in the team have any background with. To learn Python we went through much of *Teach Your Kids to Code: A Parent-Friendly Guide to Python Programming* by Bryson Payne. Python was selected based on recommendations to us from mentors as the language to use for future expansions we expect to this project.

[1] http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/deepblue/

Concurrent to this we developed pseudo-code that would outline what we would use to simulate the model in Python. The pseudo-code method utilized was Input-Process-Output diagrams.

Research: History of Computerized Chess

On May 11, 1997, a IBM manufactured supercomputer Deep Blue won a world chess champion game after a six game match: Deep Blue won three times, one for the champion, and three draws. This match lasted for a few days, and had massive interest from media coverage from around the globe. Deep Blue was such an advanced computer, it was able to analyze 50 billion moves in three minutes. Behind the match was very crucial computer science that pushed forward the capability of computers able to handle the complex computations needed to help and discover new medical drugs, to do the extensive financial modeling to analyze trends, handle broad database searches, and perform immense calculations needed in many fields of science[2].

During 1985, in Hamburg Germany, Garry Kasparov, known as one of the greatest chess players of all time, Kasparov played against 32 different chess computers at the same time in what is acknowledged as a simultaneous exhibition. He walked one machine to the next, making moves for the period of more than five hours. The four leading chess supercomputing manufacturers had sent their top models, along with eight computers named after Kasparov from the electronics firm Saitek. The state of computer chess was not that hard for Gary when he received a perfect score of 32-0. Although he had a distressing moment while he was playing one of the “Kasparov” brand models. However, Gary found out a way to trick this machine, he made the machine make a sacrifice, that the machine did not need to do. 11 years later, during 1997, Garry Kasparov barely defeated the IBM produced Deep Thought. Then IBM doubled its efforts and increased the computer’s processing power, and Kasparov lost the rematch against Deep Blue which made headlines around the world.

Input-Process-Output (IPO) Diagrams (See Other work products)

IPO diagrams are diagrams that you put something in and then process it for the output. In this project, Inputs are generally where the pieces are, Process processes the input to find all the possible moves, including the opponents, and choose the best move to get the output, Outputs are the new space or file using the process.

2

- Black Evasive IPO will evaluate the potential for moving the Black King out of the way of any checkmate and into any move that might cause a draw in the game (as this is the best case situation for Black)
- File Open IPO will handle file operations for saving the game moves and summary to a CSV format. It will begin at the start of the program as one of the first modules to run. There will be call to it for writing the game data and closing the file. It will be assumed that all game data will reside in memory until a Checkmate, Draw, or Stalemate occurs wherein a flag will be set to write the data to file, close the file and end the game..
- Mate IPO This module will evaluate the potential for moving the White Queen into a mating position with the back-up protection of the White King. If possible the move will be made with signification that the game is complete with a check mate.
- White Supportive IPO checkmates the Black king.
- PinningOps IPO will pin the black king into a corner or edge.

Discussion

Verification and Validation: Checkmate Combinations
The table[3] below shows **some** checkmates to **help** verify our code. It does **not** have all the possible checkmates. It is our current expectation of the resulting games; however, it is possible that new or unidentified mates will occur. We will check the validity of the solution by taking the record of the game and move it out physically to see that it truly is a mate.

Black King	White King	White Queen
a1	a3, b3, c1, c2, or c3	a2, b1, or b2
a2	c1, c2, or c3	b2
a3	c2, c3, or c4	b3
a4	c3, c4, or c5	b4
a5	c4, c5, or c6	b5
a6	c5, c6, or c7	b6

[3] See Other Work Products for a layout of the chessboard with the algebraic notation used.

a7	c6, c7, or c8	b7
a8	a6, b6, c6, c7, or c8	a7, b7, or b8
h1	f1, f2, f3, g3, or h3	g1, g2, or h2
h2	f1, f2, or f3	g2
h3	f2, f3, or f4	g3
h4	f3, f4, or f5	g4
h5	f4, f5, or f6	g5
h6	f5, f6, or f7	g6
h7	f6, f7, or f8	g7
h8	f6, f7, f8 g6, or h6	g7, g8, h7
b1	a3, b3, or c3	b2
c1	b3, c3, or d3	c2
d1	c3, d3, or e3	d2
e1	d3, e3, or f3	e2
f1	e3, f3, or g3	f2
g1	f3, g3, or h3	g2

Results

Our project was not fully completed, as we not able to get a fully functioning model. We completed coding all IPO's except the Pinning and White Supportive. The reason is that we first needed to ensure that the Black Evasive movement was properly operating. In our code we found that when multiple game tries were selected a non-convergence of a solution would occur. It should not happen as we ensure there is a setup of the game that does allow for Black to move. We spent some time on trying to resolve this but ran out of time to resolve this problem. Further it should be noted that we are new to python and this has impacted our resolution also.

Conclusions

The present conclusion is that we need more Python understanding as using this language was a challenge for us, however, was very enjoyable.

References Used

Software

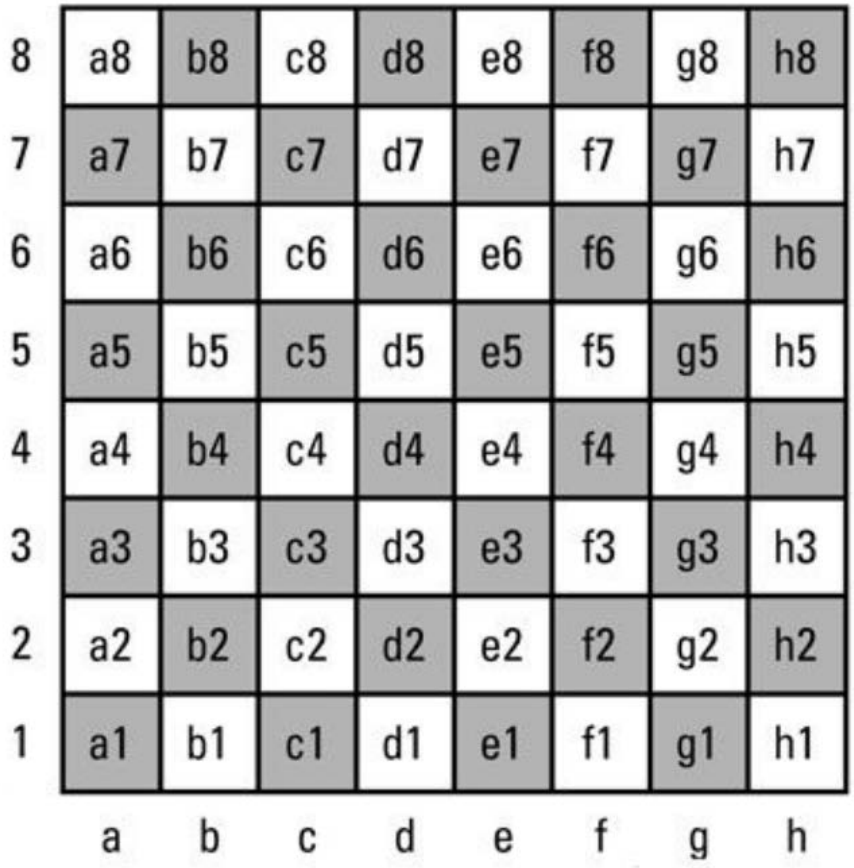
- Python 3: Coding language
- Anaconda for Python 3: scientific library support for python
- Google Drive: to share project with members
- Gimp: for photo image editing
- Idle: for python coding

References

- Teach your Kids to Code by Bryson Payne
- Web Sources
    - https://chessprogramming.wikispaces.com/Home
      This site provided information on formulas to check diagonal and antidiagonal position

Other work products

Chess Board Notation



Open Game File IPO Diagram

Author: Shaun Kilde Date: 01/24/17

Input *Describe the inputs*	Process *Detail the processes on the inputs*	Output *Describe the output(s)*

At startup: None
At Game Complete Flag = True
◦ File Name
◦ Starting White Potions
◦ Starting Black Position
◦ Total Moves Count
◦ Move Data Array
	1. At Startup:
	a. Call system to obtain:
	I. Date in YYYYMMDD format
	II. Time in HHMMSS format
	b. Create FileName with this general format:
	`KQvsKendgame_yyyymmdd-hhmmss.cvs`
	c. Create and open file with FileName
	d. Return
	2. If Game Complete Flag = True
	a. Write out data file with this format to FileName:
	`File Name; , <KQvsKendgame_yyyymmdd-hhmmss.cvs>`
	`GAME SUMMARY`
	`Starting white positions;`
	`,K xx ,Q yy`
	`Starting black positions; ,K zz`
	`Total moves; , <number of moves>`
	`MOVES:`
	`White , Black`
	`? , ?`
	`. , .`
	`EOF`
	b. Close File
	c. End Run of this program iteration
	I. Yes. Make Move
	i. Set Draw Flag = True
		File of game played

Black Evasive IPO Diagram

Author: Shaun Kilde Date: 02/07/17

Input *Describe the inputs*	Process *Detail the processes on the inputs*	Output *Describe the output(s)*
Current White King Position Current White Queen Position Current Black King Position Status: Possible Mate Flag	1. If Possible Mate Flag = False a. Look for all valid moves possible b. Limit moves to those with no checkmate potential c. Limit moves to those that move farthest from edges d. Randomly chose from remaining. 2. If Possible Mate Flag = True a. Look for all valid moves possible b. Is there a move that will create a Draw? I. Yes. Make Move i. Set Draw Flag = True ii. If no Draw, then: 1. Limit moves to those with no checkmate potential. 2. Limit moves to those that move farthest from edges 3. Randomly chose from remaining. II. No. Make Move i. Limit moves to those with no checkmate potential. ii. Limit moves to those that move farthest from edges iii. Randomly chose from remaining.	Black King Move

Mate IPO Diagram

Author: Daniel Kim Date: 02/06/17

Input *Describe the inputs*	**Process** *Detail the processes on the inputs*	**Output** *Describe the output(s)*
□ Current White King Position □ Current White Queen Position □ Current Black King Position □ Status: Possible Mate Flag	1. If Possible Mate Flag = False a. Return 2. If Possible Mate Flag = True a. Isolate mating move of Queen directly opposite Black King b. Is this move have Queen support back-up by White King for protection? I. Yes. Make Move i. Set Checkmate Flag = True II. No. i. Set Possible Mate Flag = False ii. Call PinningOps.py to reevaluate moves.	□ White Queen Move with set of Checkmate Flag *or* □ Call to PinningOps.py

Author: Daniel Kim Date: 02/04/17

Pinning IPO Diagram

Input *Describe the inputs*	**Process** *Detail the processes on the inputs*	**Output** *Describe the output(s)*
□ Current Queen White Position □ Current Black King Position □ Current White King Position	1. Find positions that are a knight's pattern away from the Black King. a. There potential will be 8 positions that are available. b. If Queen is already in a knight position away then call to KingSupport.py to move closer to the White King. 2. Down select invalid positions because they are off the board or occupied. 3. Evaluate if Queen can get into one of the positions. a. If Queen can move to more than 1 of these positions choose the one that will force the King closer to an edge or corner. b. If King is pushed to an edge or corner DO NOT move Queen but call KingSupport.py to support a Checkmate.	□ New White Queen Position *or* □ Call to KingSupport.py for support

Author: Daniel Kim Date: Sun Feb 26 2017

Description & Purpose of this IPO: White Supportive

IPO Diagram

Input *Describe the inputs*	**Process** *Detail the processes on the inputs*	**Output** *Describe the output(s)*
Position of White King Queen and Black King	Move White King behind where white queen is going to be in the in-your-face-checkmate.	Checkmated Black King.

Source Code

See CMasters.zip for the code as we currently have it.

Significant Achievement

The greatest achievement for us was the learning of python.

Acknowledgements

We want to thank our Mentor (Mr. Kilde) for working with us and moving us on to learning a new programing language.