Project 1: Tic Tac Toe

Xuejian Ye

September 30, 2018

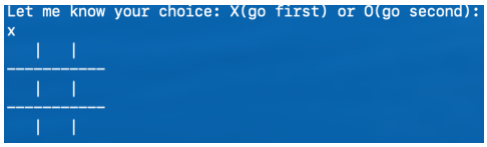
Table of Contents

1. Project Summary:

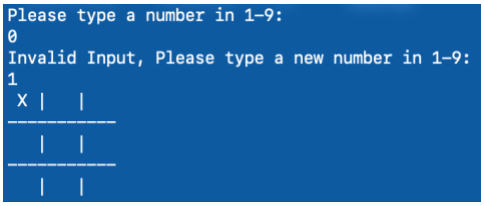
This project is made up of three part: basic TTT, advanced TTT and super TTT.

1.1. Basic TTT:

a) Language: Python3
b) Individual work
c) How to run project:
   i. Run it in terminal
   ii. First it will ask you to choose 'X' or 'O' and the game will really begin after you type 'X' or 'O'

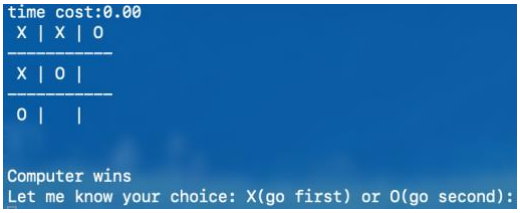


   iii. When it is player's turn to move, it will show the board and the words to prompt to type the number [1-9] (your preference position) and program will check the number inputted and input will be invalid if the number is not between 1 and 9.



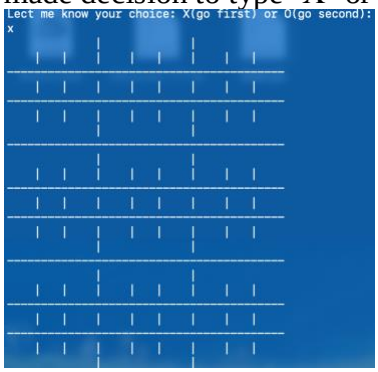
   iv. When it is computer's turn to move, it will show the result of moving and the time it costs in this movement and the number about position.

   v. When the game is over, program will show the result (who win this game) and start a new game if you want to try again.

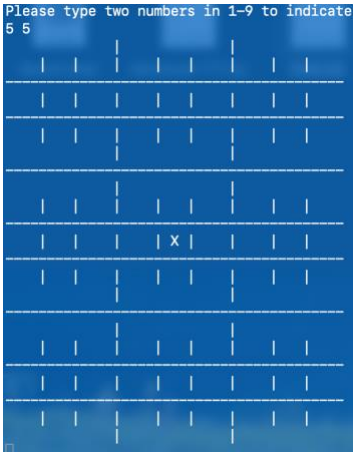


1.2. Advanced TTT:

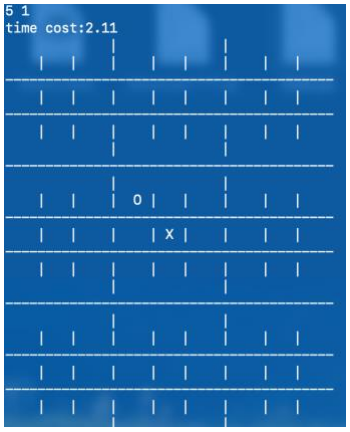
a) Language: Python3
b) Individual work
c) How to run project:
    i. Run it in terminal
    ii. First it will ask you to choose ‘X’ or ‘O’ and the game will really begin after you have made decision to type ‘X’ or ‘O’



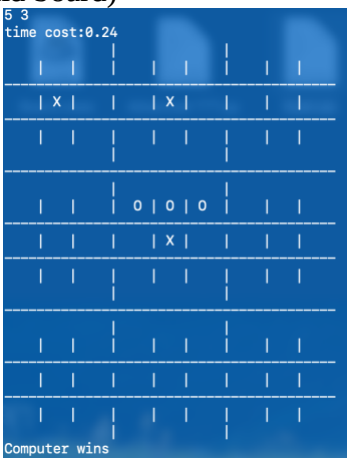
    iii. When it is player's turn to move, it will show the board and the words to prompt player to type two numbers (separated with space), for example: 1 6. And each number should be between 1 and 9, if not, computer will check whether this input is applicable. Then it shows the board after moving.



    iv. When it is AI's turn to move, it will show the result of moving and the time it costs in this movement and the number about position. (in this plot, AI choose NO.5 grid and NO.1 position)



v. When the game is over, program will show the result (who win this game) and start a new game if you want to try again.(AI win directly by wining a single child board)



1.3. Super TTT:

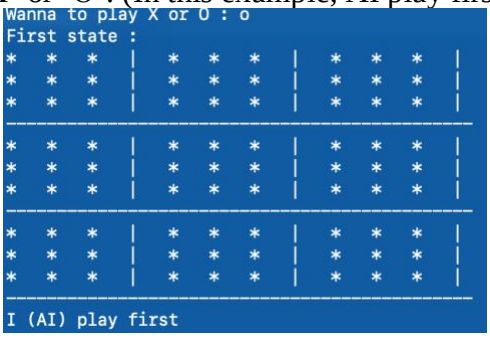
a) Language: Java

b) Teamwork with Yanhao Ding and Wei Zhou. This part is completed by Yanhao Ding and my work is primarily on debugging to optimize program.

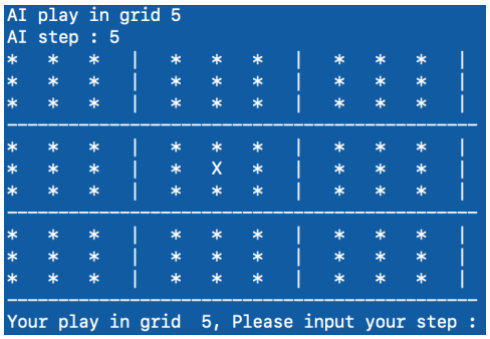
c) How to run project:

i. Run it in terminal

ii. It will ask you to choose ‘X’ or ‘O’ and the game will really begin after you type ‘X’ or ‘O’. (In this example, AI play first)

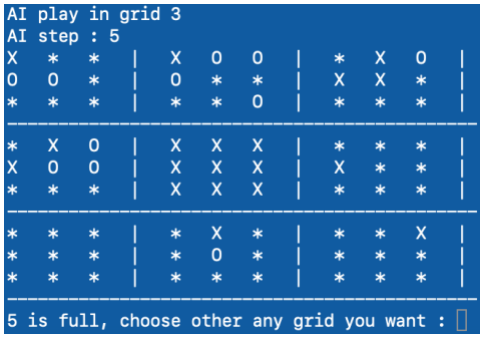


iii. When it is computer's turn to move, it will show the result of moving and the number about position.

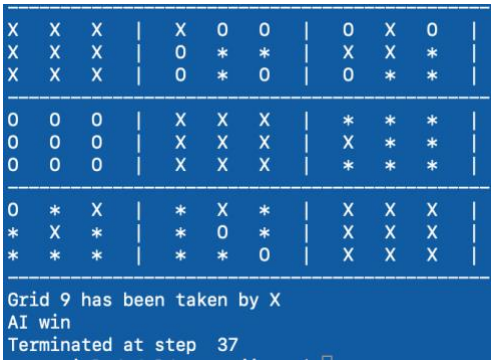


iv. When it is player's turn to move, t will show the board and the words to prompt player to type one numbers to choose the position you want to play, and player have no choice to choose grid which is depended on AI. (In this example, AI play first)

v. Who (player or AI) win one child board in the 9 boards will possess the whole grid and mark all the elements in this grid to be ‘O’ at same time . The opponent could only choose other boards to place ‘X’ or ‘O’.



vi. Finally, after fierce battle, AI firstly wins 3 child boards in row diagonally.



2. Part 1: Basic Tic-Tac-Toe

Grid: 3*3

2.1.Algorithm:

Algorithm in basic TTT is exploring every possible state and use MINIMAX function to simulate the game as every player perform most optimally. Under such a condition, the program return the best choice for AI.

2.2.Project structure:

- **Initial State**: using list to construct an one-dimension board.
- **Terminal test**: it is used to test whether game is over and return the result of this game.
- **Minimax decision**: To check whether or not the current move is better than the best move we take the help the function which will consider all the possible ways the game can go and returns the best value for that move, assuming the opponent also plays optimally

3. Part 2: Advanced Tic-Tac-Toe

Grid: 3*3 → 9*9

One crucial constraint: If a player has just played at some position on some board, then the next player must play on the board in the corresponding position in the grid.

3.1.Algorithm:

Now, if we only use MINIMAX cannot solve this problem in acceptable time. The search depth is so deep, and this search will be very time-consuming.

So in this part, I use H-MINIMAX and Alpha-beta pruning algorithm to optimize this program.

3.2. Program structure:

- **Initial State**: using list to construct a two-dimension board.
- **Actions**: if AI play first, the center of board (5,5) is first AI move, which has the highest possibility to win. Get the previous player's move and find all the possible moves in the corresponding child hoard. If the child board is full, choose any other grids to move randomly.
- **Terminal test**: it is used to test whether game is over and return the result of this game. In my code, I set the search depth is smaller than 6 because it is impractical to require program to explore every move and it should stop searching under certain condition due to the depth of the search tree.
- **Utility**: For terminal state, return the final numeric value of the game and for non-terminal state, return the evaluation of each result possibility to win this game.
- **Strategy**: the basic part is similar with the basic TTT, suppose each player make the best decision in each step. Simulate the possible game trend and make the best moving decision to win this game with maximum probability.

- **Evaluation Function**: Scan all the child board and check the row horizontally, vertically, or diagonally, if there are two moves from same player and the other box is blank, and the player gets two points. Then calculate the scores of each player and the estimated possibility to win. Finally, return an estimated chance for AI to win.

4. Part 3: Super Tic-Tac-Toe

Differ from Advanced Tic-Tac-Toe, Super Tic-Tac-Toe need much more processes to find the best path with highest probability to win the game.

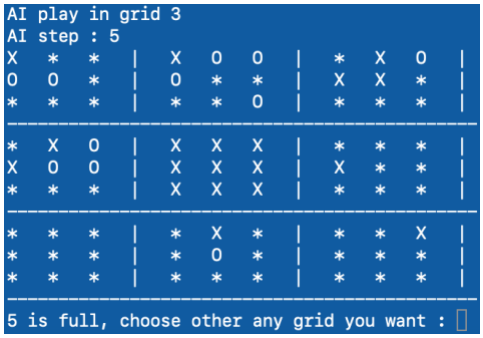
4.1. Algorithm

MINIMAX with alpha-beta pruning algorithm is still the main method but more complex.

4.2. Project structure:

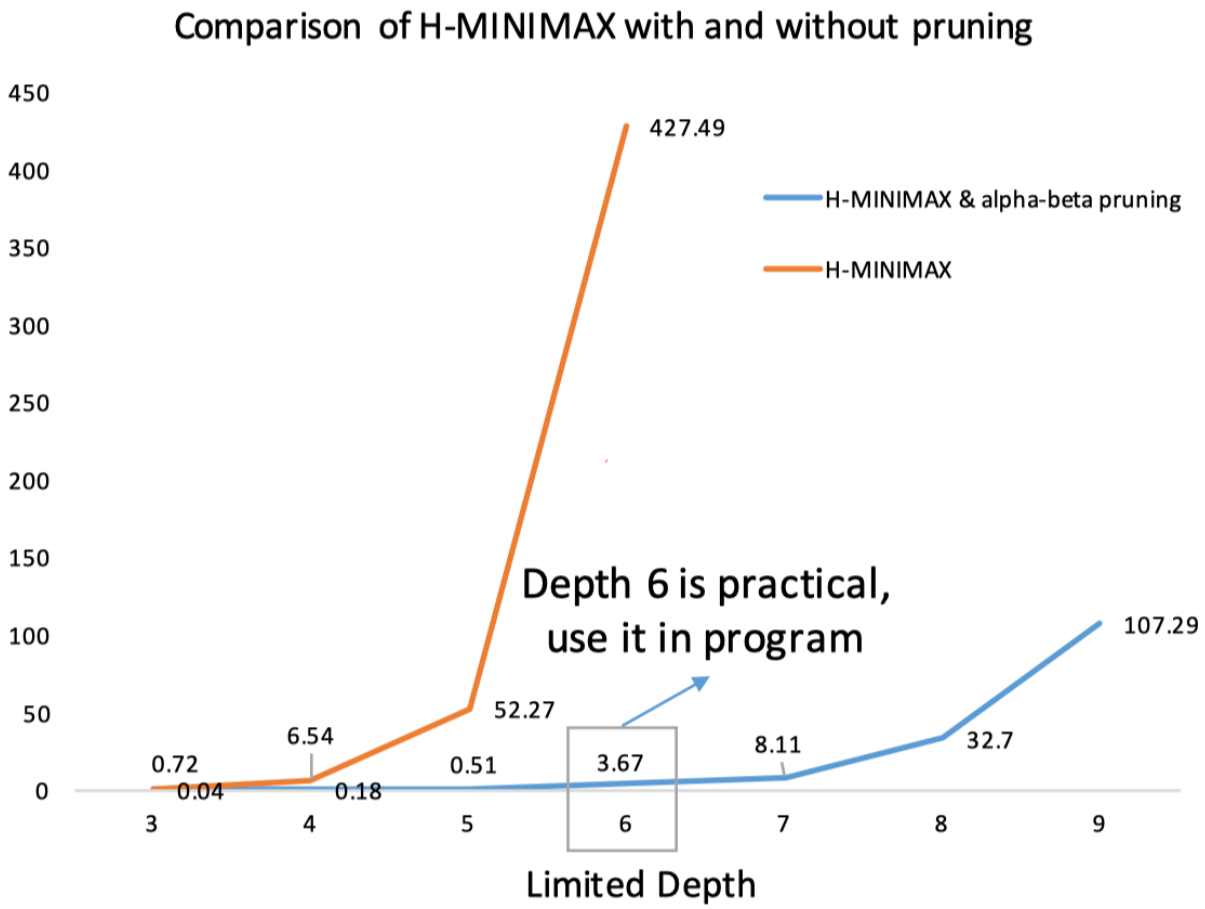
Differ from advanced TTT part

- **Child board status**: To keep tracing the status of each child board, I use a list with length of 9 to record each child status (initially, all of the value in this list is 0). If AI 'O' has won in the board 3, set list[2] = -1 and mark all the elements in grid 3 to be 'O' at same time. (If player 'X' has won in the board 3, set list[2] = 1 and mark all the elements in grid 3 to be 'X' at same time.). Such as the figure below.



When we collect the status of each child board, and then we could know who win this game by testing the values (1 or -1) of this list is in one line or column or diagonal in the 9 board.

5. Program Performance Analysis



In this figure, we can know that only using MINIMAX algorithm cannot complete the complex search mission. And using alpha-beta pruning has better performance could achieve greater efficiency. Using different evaluation function might have better performance but the function I use is acceptable and simple

As for the Super TTT, after many time test, we notice that our method is not good enough for AI to receive great performance. We still have possible to beat AI which demonstrates our algorithm still need more improvement.