Project 1 Report

What I did in this project:

I finish this project independently

- Implement the minimax algorithm;
- Implement the depth limited alpha-beta pruning algorithm;
- Design and implement two heuristic functions which serves for advanced and ultimate level TicTacToe game;
- Design and implement the class, methods, and logical relation among classes;
- Write this report.

The detailed instructions are given below.

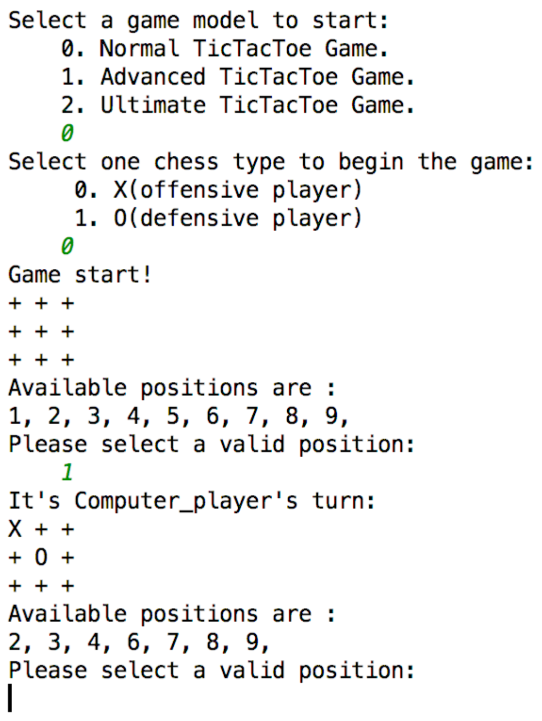
Basic TTT game

Rules

Basic TTT game is two-player game, X and O, who make a move in 3*3 board by turns. The winner will be the player whose chess number is three in horizontal, vertical, or diagonal direction.

Implementation

The minimax algorithm is used to implement the basic level of TTT game. As we know, minimax searches all way down to the leaf nodes and then back up to the root. That is, minimax algorithm can always find the best move. However, the drawback is that its time complexity is exponential. Therefore, in my project, there will be an obviously pause every time when computer player takes X to make the first move.



```
Select a game model to start:
    0. Normal TicTacToe Game.
    1. Advanced TicTacToe Game.
    2. Ultimate TicTacToe Game.
    0
Select one chess type to begin the game:
     0. X(offensive player)
     1. O(defensive player)
    0
Game start!
+ + +
+ + +
+ + +
Available positions are :
1, 2, 3, 4, 5, 6, 7, 8, 9,
Please select a valid position:
    1
It's Computer_player's turn:
X + +
+ O +
+ + +
Available positions are :
2, 3, 4, 6, 7, 8, 9,
Please select a valid position:
```

Basic TTT game screenshots

Advanced TTT game

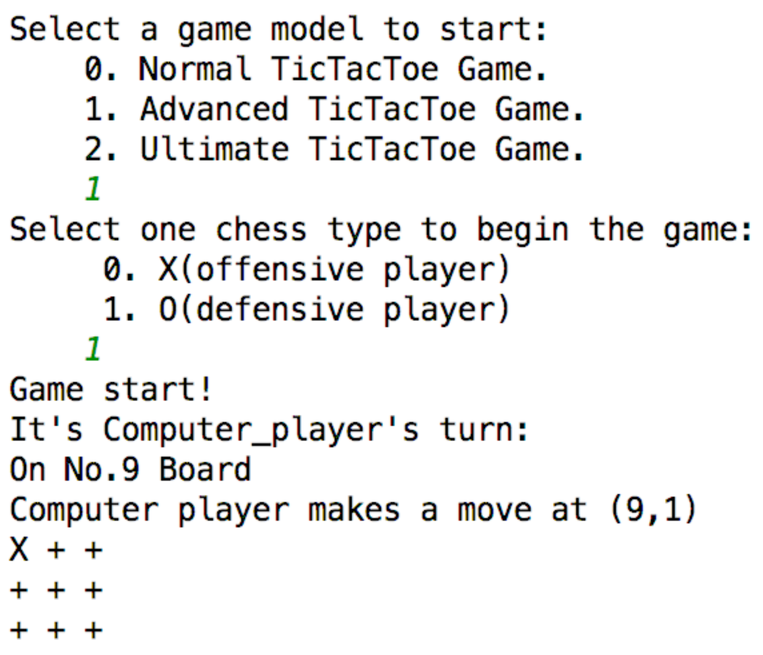
Rules

Advanced TTT game is a variant of basic TTT: there are 3*3 grids and each grid contains a small 3*3 board. In addition, except for first hand player and draw board, the board that current player is going to play on is decided by previous player's move. For example, the first player makes a move on No.3 board and position 5, next player has to make a move on No.5 board. The player who wins a single board will win the game.

Implementation

It is obviously that minimax algorithm can't work well under such condition. Therefore, I adopt alpha-beta pruning algorithm and set a depth limit to improve efficiency. The key point or difficulty to solve this advanced TTT game is not on implementing alpha-beta algorithm but on how to design a heuristic function. The intelligence of computer player is decided by the heuristic function. In my project, I design an evaluation method which works when depth is zero. The main idea is to judge the chance of wining on the given board. Therefore, I firstly fill all empty positions with current player's chess and then count how many times the current player could win under such condition. Similarly, I also fill all

empty positions with opponent player's chess and count how many times he could win this board. Finally, let two win-times numbers time score correspond to their player (e.g. for player who takes 'X', his win-times number could time 10 points) and add them together. The result of this evaluation reflect the condition of current board to some extent.
Besides, the search depth is also important. In my project, I set the depth limit to 3 by referring a report about alpha-beta algorithm which said that an alpha-beta agent of depth 2 won 79.8% against an alpha-beta agent of depth 1. An alpha-beta agent of depth 3 won 69.5% of games played against a player of depth 2 and 75.5% of games played against depth 1.
I firstly adopt alpha-beta algorithm on basic TTT game, and I find that the improvement of efficiency is obvious. Alpha-beta algorithm is similar to minimax when depth is fixed, but the difference is that alpha-beta algorithms takes considerably less time to reach the same result and hence we can use this extra time to search deeper levels of the game tree.



```
Select a game model to start:
    0. Normal TicTacToe Game.
    1. Advanced TicTacToe Game.
    2. Ultimate TicTacToe Game.
    1
Select one chess type to begin the game:
     0. X(offensive player)
     1. O(defensive player)
    1
Game start!
It's Computer_player's turn:
On No.9 Board
Computer player makes a move at (9,1)
X + +
+ + +
+ + +
```

Nine-board TTT game screenshot

Ultimate TTT game

Rules

The ultimate game is a harder version of nine-board TTT game. The winner will have to win three boards in vertical, horizontal, and diagonal direction.

Implementation

I still adopt alpha-beta pruning algorithm combined with depth limited search but design and implement some new heuristics.

My new heuristic takes into consideration some features of the given board: If current player will get score when he wins the given board and score depends on the position of given board on 3*3 grids: if it is center board, the score is 13; if it is one of the corner boards, the score is 9 or the score will be 5.

Additionally, I also take advantage of features of this game: the board that next player is going to make move is decided by previous player's move, but if the next board has already won by one of players or draw, current player could make a move anywhere he wants. Therefore, if you are sent to a board which is tie or won by someone else, you will get 3 points because it is a good situation. The previous heuristic function of alpha-beta algorithm is also adopted when depth is zero.

Another key points I must notice is the conditions when a board is tie or won so current player has to make a decision on which board to go. In my project, when such condition occurs, I will make an iterative operation on 3*3 grids and call alpha-beta algorithm to finds a best board.