ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

---------------o0o---------------

BÁO CÁO ĐỒ ÁN
LAB02 – TIC-TAC-TOE

GVHD: Phạm Trọng Nghĩa, Nguyễn Thái Vũ

Môn: Cơ sở trí tuệ nhân tạo

Tên và MSSV người thực hiện đồ án:

Nguyễn Mạnh Hùng – 20127030 – 20CLC04

TP. HỒ CHÍ MINH, THÁNG 6, NĂM 2022

MỤC LỤC

I. PERFORMANCE EVALUATION

All the requirements and tasked has been completely done and run successfully.
The project suceed in debugging and ready for the report in 24/5/2022.

Tasks	Performance evaluation
Implement adversarial search algorithms for AI	100%
Create tic-tac-toe games with 2 maps (5x5,3x3)	100%
Implement GUI for the games.	100%

II. OVERVIEW ABOUT PROJECT

Project is about the tic-tac-toe games. Player run the game and have 2 maps to play, 3x3 and 5x5. After choosing a map, player will compete to an programed AI and try to beat it by playing tic-tac-toe games.

Version Python IDE was used in this project is PyCharm (with debugger version is Python 3.10)

Libraries were used in this project are pygames, numpy, copy.

Name of most used variables and Classes in this project:

- screen: a variable to display the UI of the games and show the match result on screen.
- img: use to implement the images of logo and button.
- width, height: define the screen size (width x height).
- msg, text: show the notification or message on screen board.
- Board: create an 2D-array and store the game status in computer memory.
- AI: a bot object to play against with player in tic tac toe game.
- Game: object that inherits Board and AI classes in order to operate the game.
- Button: object for button for choosing maps.

Note: map3.py and map5.py is two map that execute the games in 3x3 and 5x5 gameplay respectively. Whereas, button.py is the file whose function is showing and operating button function in choosing gameplay phase. At last, run.py is the main file to run and execute the whole game system.

Image files: download.jpg is for the logo of the game, and firstbutton.jpg and secondbutton.jpg for the showcase of two buttons.

Link video demo:
https://drive.google.com/file/d/1Dp5E_fI87Ei1eZPRqRKN9Sv3r0P8kx5T/view?usp=sharing

III. PERSONAL PROJECT

1. Personal Idea

The idea of program:

When running the run.py, there will be a screen with 2 maps to play, 3x3 and 5x5. In order to choose a map to play, player just click the proper button on the screen then the program will set the correct map.

In gameplay, player just drag your mouse to any position on whe board that you want and click it to mark your turn then AI will respond and play back. To win the game, you have to mark 3 in a row in map 3x3 and 4 in a row in map 5x5.

When the game is over, the program show the notification on the board to inform who is the winner or game is draw. To replay the game, just press space bar key on your keyboard to have a rematch.

1.1. Algorithms

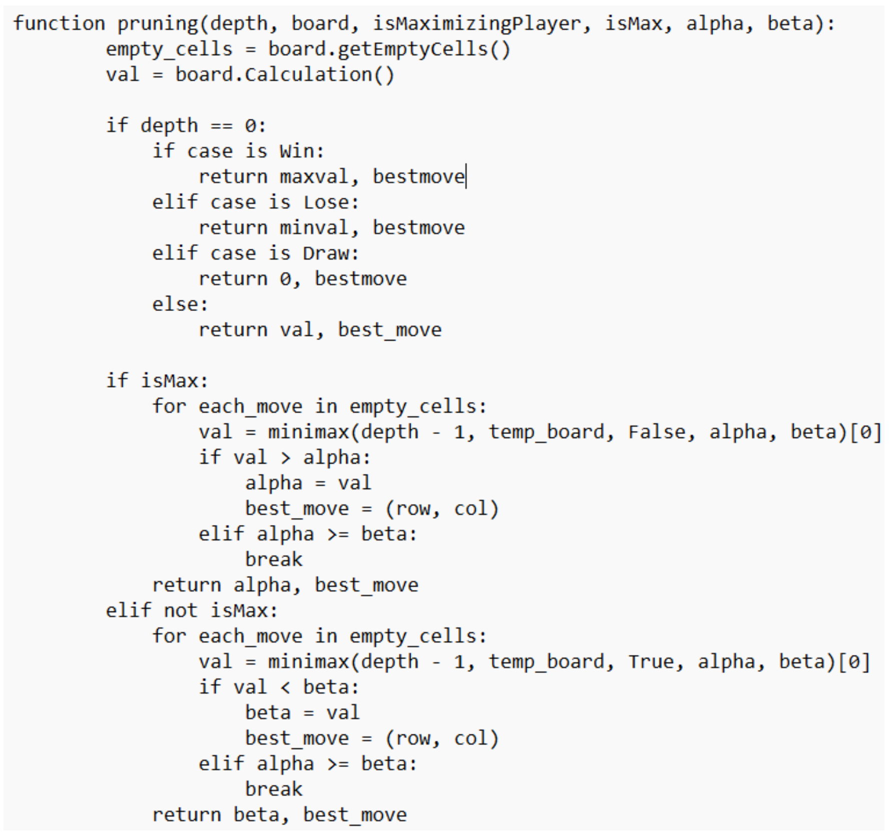
- **Minimax in 3x3**
  - *Idea:* The key is a back and forth between the two players, where the player whose turn desires to pick the move with the maximum score. In turn, the scores for each of the available moves are determined by the opposing player deciding which of its available moves has the minimum score. And the scores for the opposing players moves are again determined by the turn-taking player trying to maximize its score and so on all the way down the move tree to an end state. Pseudo code for this algorithm in project:

```
function minimax(board, isMaximizingPlayer):
    if(CheckStateGame(curMove) == WIN_GAME)
       return MAX, curmove
    if(CheckStateGame(curMove) == LOSE_GAME)
        return MIN, curmove
    if( CheckStateGame(curMove) == DRAW_GAME)
       return DRAW_VALUE, curmove
     if isMaximizingPlayer :
         bestVal = -INFINITY
         for each move in board :
            value = minimax(board, false)
            bestVal = max( bestVal, value)
        return bestVal, move

     else :
        bestVal = +INFINITY
        for each move in board :
             value = minimax(board, true)
             bestVal = min( bestVal, value)
         return bestVal, move
```

- *In coding:* the minimax function in AI class demonstrates clearly about the flow of the algorithms and the DragonFang function in the same class is the main class runs the algorithm and responds a resonable move in playing against player.

- **Alpha – beta pruning in 5x5**
  - *Idea:* this algorithm is similar to minimax in the way looking for the goal state. However, it seeks to reduce the number of nodes that needs to be evaluated in the search tree by the minimax algorithm. Therefore, it's going to be faster and more efficient in this map. Instead of going to the final state, we just let AI think 4 steps ahead by setting depth is 4 in this algorithm because of the affect to time and space complexity. Moreover, the will be some fuctions to evaluate the score which measures the win probability for each cases in the game when they are not in the final state.
    Pseudo code for this algorithm in project:



```
function pruning(depth, board, isMaximizingPlayer, isMax, alpha, beta):
        empty_cells = board.getEmptyCells()
        val = board.Calculation()

        if depth == 0:
            if case is Win:
                return maxval, bestmove
            elif case is Lose:
                return minval, bestmove
            elif case is Draw:
                return 0, bestmove
            else:
                return val, best_move

        if isMax:
            for each_move in empty_cells:
                val = minimax(depth - 1, temp_board, False, alpha, beta)[0]
                if val > alpha:
                    alpha = val
                    best_move = (row, col)
                elif alpha >= beta:
                    break
            return alpha, best_move
        elif not isMax:
            for each_move in empty_cells:
                val = minimax(depth - 1, temp_board, True, alpha, beta)[0]
                if val < beta:
                    beta = val
                    best_move = (row, col)
                elif alpha >= beta:
                    break
            return beta, best_move
```

- *In coding:* the alpha – beta pruning function in AI class demonstrates clearly about the flow of the algorithms. CountSequence, StateVal, CalMainAndLines and CountSubDiag are the heuristics functions whose mission is calculating the score which measures the win probability for AI after playing 4 steps ahead. The DragonFang function in the same class is the main class runs the algorithm and responds a resonable move in playing against player.

1.2. Conclusion

In this project: b = 9 in 3x3 map and b = 16 in 4x4 map, m = 4 in 4x4 and m = 5 legal move in 3x3.

Algorithms	**Time complexity**	**Space complexity**
Minimax	$O(b^m)$	O(bm)
Alpha – beta pruning	$O(b^{m/2})$	O(bm)

REFERENCES

- Teaching slides on Moodle.
- Book: Artificial Intelligence A Modern Approach (4th Edition)
- Reference source code: GitHub - vineetjoshi253/TicTacToe-MiniMax: A python implementation of Tic-Tac-Toe using MiniMax Algorithm.
- How to make button in pygame: https://pythonprogramming.altervista.org/buttons-in-pygame/
- Pygame tutorial for beginners: https://www.youtube.com/watch?v=FfWpgLFMI7w