Page No. :
Date :

Name : Jyoti Mukesh Giri

Class : BE - IT

Roll.No : 19

Subject : IS Lab

DOP	DOA	Marks	sign

- Min Max with Alpha Beta for Tic Tac Toe:

- The goal of Tic-Tac-Toe is to be the first player to get three in a row on 3x3 grid
- X always goes first
- Players alternate placing "Xs" and "Os" on board until either :-

i) One player has three in a row horizontally, vertically or diagonally
ii) All nine squares are filled.

- Programmer created in "winningstates" named set containing a list of all possible win con conditions inside "properties.py", if a players places "Xs" or "Os" in any of the list, they are declared winner.
The winning states are:

winning states = ([0,1,2], [3,4,5], [6,7,8]
[0,3,6], [1,4,7], [2,5,8],
[0,4,8], [2,4,6])

- Programmer has created a dummy bot which chooses positions randomly at DummyBot.py, The GameBoard initalises, the free space to None (List of Nones).
- Programmer also created a minmax bot which use minmax algorithm with AlphaBeta pruning to decide the best move.

The main.py starts by initialization of two objects of MinmaxBot and DummyBot. The code then creates a variable Judge which called TicTacToe Judge to which both objects are passed, the TicTacToeJudge.py decides the winner.

Programmer also created a Helper method, Helper.py which gets the opponent's position to bot and gets the available moves to play. & imports properties Py mentioned earlier.

- Input :- → No input from user (As both the bots, DummyBot and MinmaxBot play game)
- Output :- i) Winner Name which can be :
  a) Bot one (minmax Bot)
  b) Bot two (Dummy Bot)
  c) Draw (when all positions are filled with no winner)

The winner is decided if the bot's position is in the set of list of WinningStates()

Analysis of claim by Programmer that it uses Minmax with alpha beta pruning.

1. This claim comes from the BestMove() method in MinmaxBot.py as it uses recursion to find the next best move.
2. It starts by getting the winner() state and checks if game already coded by comparing the winner variable with SELF char, self-opponent or Draw state and returns. 1, -1, 0, respectively.
3. The method then starts a for loop which iterates through all possible moves in gameboard.
4. After every move, the BestMove() calls itself recursively to figure out next best move by the MinmaxBot.
5. The Bot then plays the move on best move and updates the Alpha, Beta variables.
6. The Alpha Beta variables are checked with value is greater than Alpha, Alpha is assigned to value and if its lower than Beta, Beta's value is updated to value

Thus the claim by programmer that it uses minmax with AlphaBeta Pruning is correct.

Output :-

i) Bot one

'x'	'0'	'x'
'0'	'0'	None
'x'	'0'	'x'

ii. Bot one

'0'	'x'	'x'
'0'	None	None
'0'	None	'x'

iii. Bot two

'0'	'x'	None
'0'	'x'	'x'
'0'	'0'	'x'

iv. Bot two

'x'	'x'	'x'
'x'	'0'	'0'
'0'	'x'	'0'

v. Bote one

'x'	'0'	None
'0'	'x'	None
'x'	'0'	'x'

vi. Bot one

'0'	'x'	'x'
'x'	'0'	None
'x'	'0'	'0'

vii. Draw

'0'	'x'	'x'
'x'	'0'	'0'
'x'	'0'	'x'

viii. Bot one

'0'	'x'	None
None	'x'	'0'
None	'x'	None

ix. Bot two

'0'	'x'	'x'
'x'	'x'	'0'
'x'	'0'	'0'

x. Draw

'x'	'0'	'x'
'0'	'0'	'x'
'x'	'x'	'0'