ARTIFICIAL INTELLIGENCE REPORT

[TIC-TAC-TOE GAME]

Submitted by:

Neeraj Appujani: 18ucc091

Submitted to:

Mr. Nitin Kumar
Dr. Poulami Dalapati
Dr. Roshni Chakraborty

CONTENTS

Page no.

1) Introduction---------- 2

2) Scope---------- 2

3) Game Tree---------- 3

4) Algorithms applied

- Minimax Algorithm---------- 3
- Alpha-beta pruning---------- 7

5) Experiments/Results---------- 10

6) Insights/Analysis---------- 13

7) Conclusions---------- 14

8) References---------- 14

9) Contributions---------- 15

10) Demo G-drive Link---------- 15

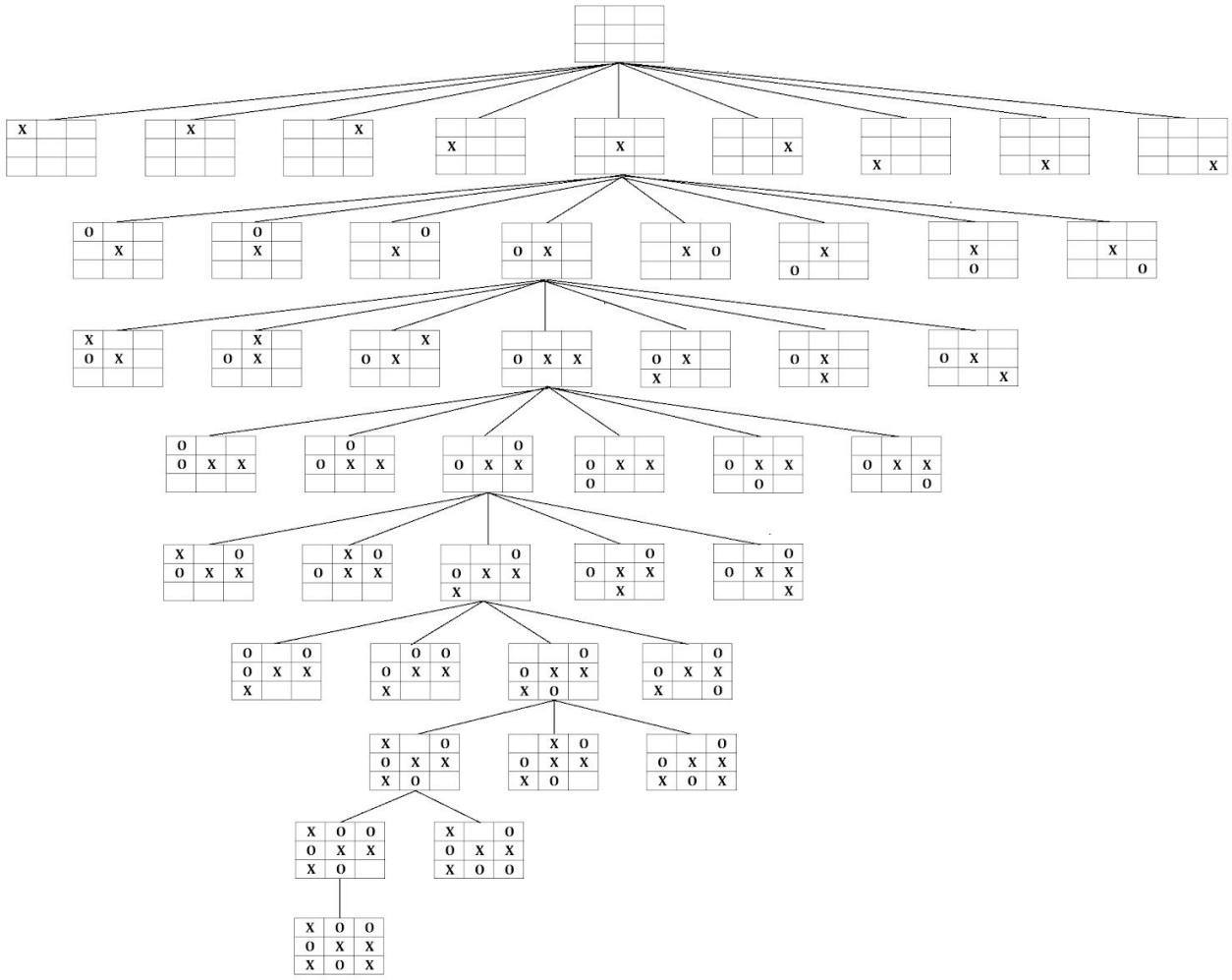
Project: Design tic-tac-toe game between two players where the first player will always have the winning strategy irrespective of the opponent's move.

Introduction: We have a two player game i.e. Tic tac toe in which each player chooses either X or O and takes turns in a 3x3 grid. Each player has only one chance per move. The game is played on an alternate basis, like first X will play, then O and so on or vice versa. The player who fills three of their own marks in the grid either vertically,horizontally or diagonally(from both upper corners to the opposite lower corners) . The player will win or loose or the game draws. Draw situation occur when no one is able to place their corresponding marks in the required fashion.
For example: The following example game is won by "X".

Scope: The game play will be simple.There will be a straightforward square board divided into 9 tiles or grid spaces. Each tile will have a numbering from one to nine. Each player will be assigned either an "X" or an "O". In one move, a player has to select one position out of these nine positions. Selected positions will be marked as "X" or "O" , depending upon who is playing. The game is over when any player satisfies the required winning condition or no moves are left. When the game is over, output will display the winner or announce a draw if no one wins. A restart option will be displayed when the game is over, if selected yes , then it will restart your game.

Tree For Tic Tac Toe Game:

All the possible combinations of the Tic-tac-toe game tree are shown below in the example:



(Winner - X)

Algorithms used:

The following algorithms are applied in tic-tac-toe game project:

a) **MINIMAX ALGORITHM:** It makes the use of recursive or backtracking algorithm which can be applied in making decision and game theory.Games

which use AI are mostly the applications of Min-Max algorithm like chess, checkers, tic tac toe etc along with enormous games involving two players. In this game, a player ensures the following two strategies:

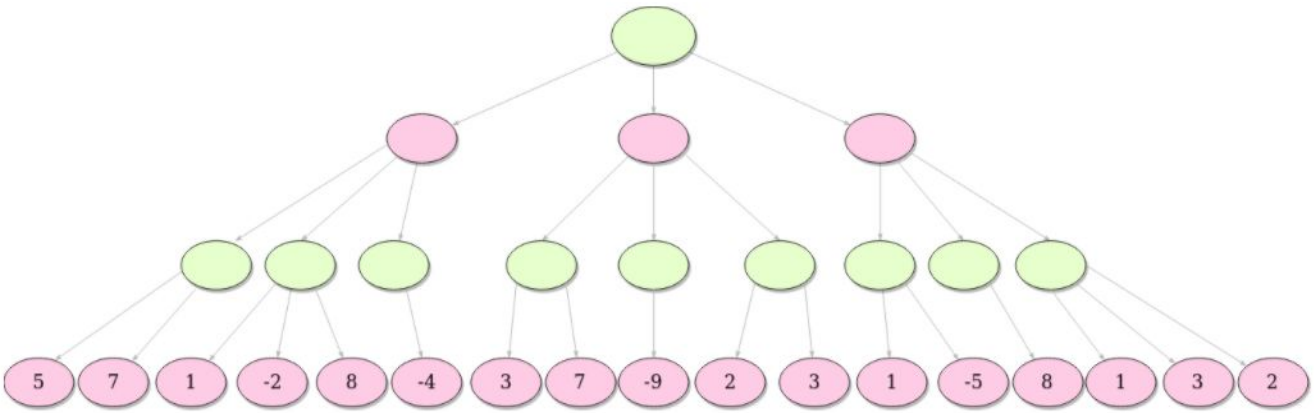
- Maximizing the player's own winning chance.
- Minimizing the opponent's losing chances.

The minimax decision of the current state is used by this algorithm. In this game, the best move a player can make is decided by the minimax, by working in the reverse manner from the end of our game. Main objective is to minimize and maximize the loss.

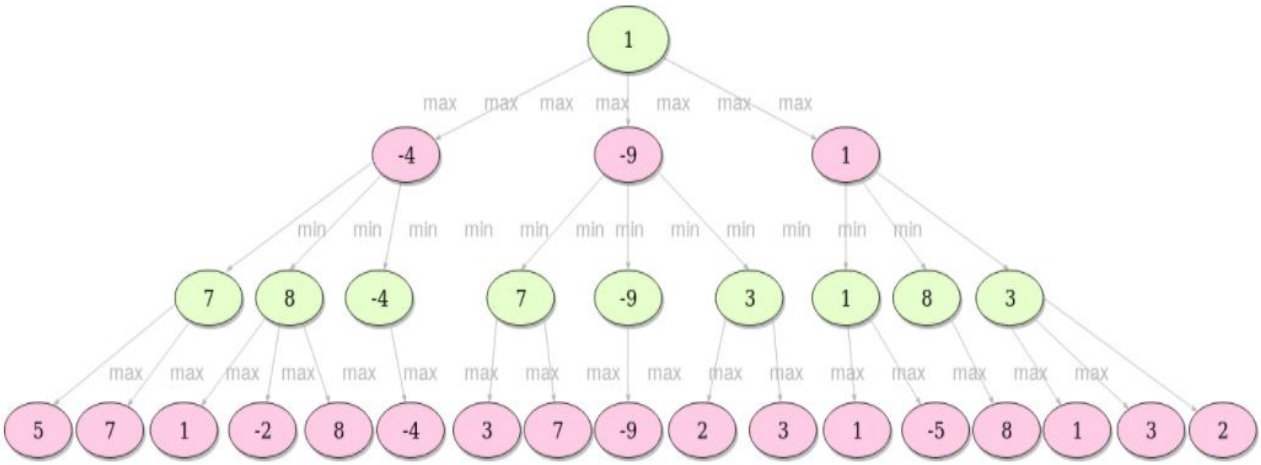
A sequence of decisions and actions made by the algorithm will be the most optimal and it will lead to goal state.

Suppose two players are assigned MAX and MIN. MAX moves first and takes turns followed by MIN, then MAX, then MIN and so on. Moves taken by MAX are independent of the moves taken by MIN. A state of maximum value will be preferred by MAX and the state of minimum value will be preferred by MIN. Our objective should be the chain and order of the actions and the decisions that should lead the desired player to a terminal state of maximum utility.

Below two diagrams shows how minimax algo works. Nodes in green are MAX nodes and in pink are MIN nodes.



Starting with only leaf nodes



Assigning values to parent nodes based on Minimax algorithm

Pseudo code for minimax Algorithm:

minimax(who play the game,current state of board)

if(game end in present state)

return who win game

Child node = valid moves for player in this state

if(turn of the max)

return maximum score_count after execute minimax on all the child node

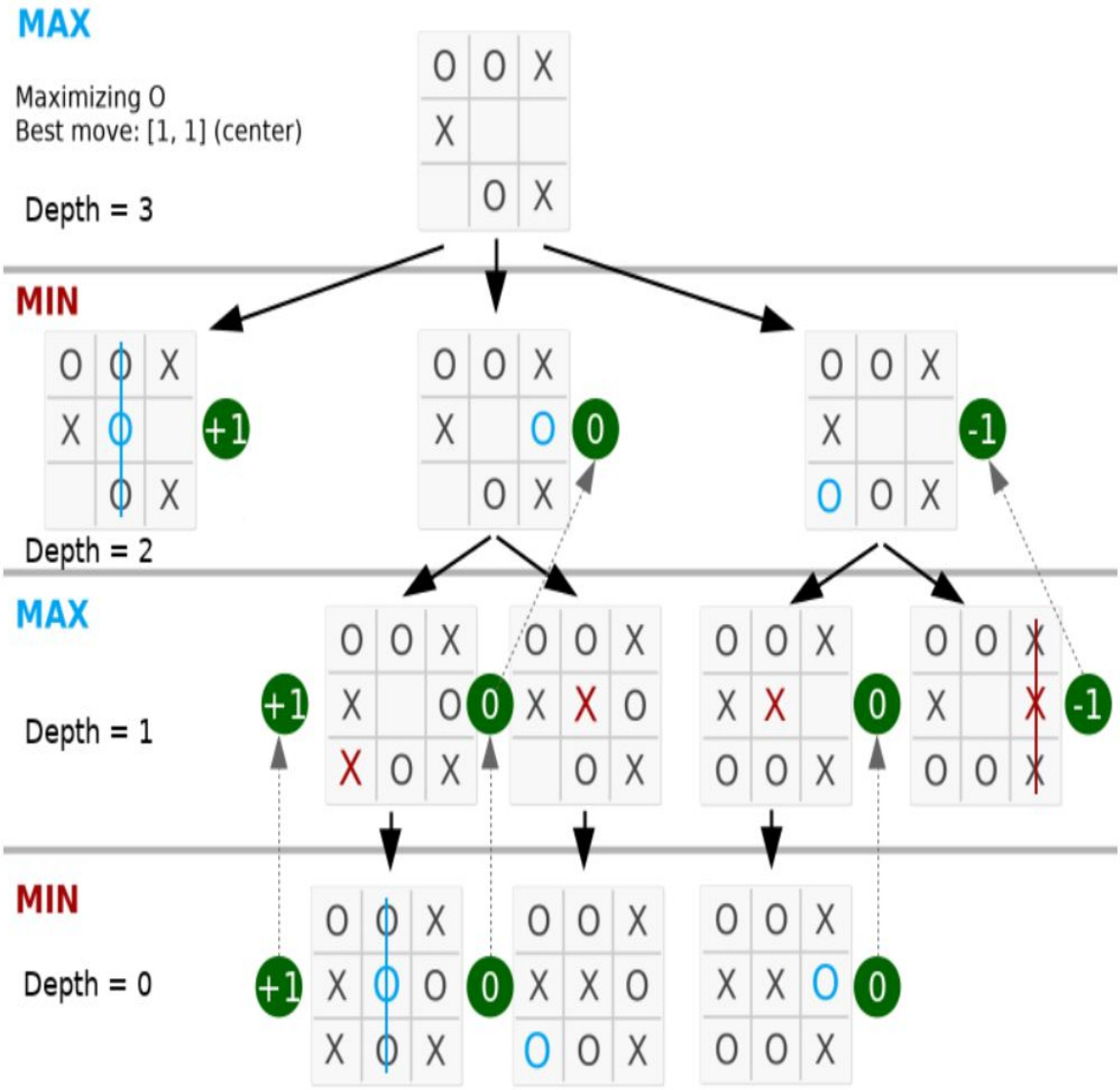
else (turn of the min)

return minimal score_count after execute minimax on all the child node

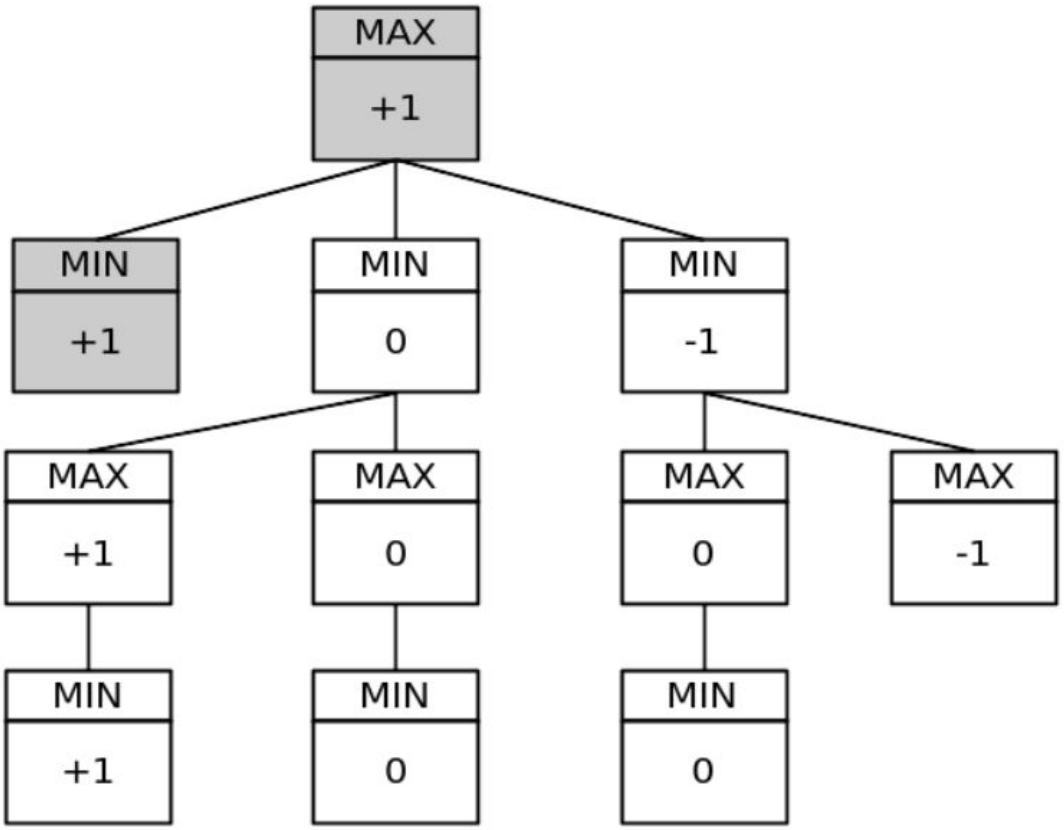
How long does the MINIMAX algorithm take?

Not much time is taken in the example of a 3x3 tic tac toe.The minimax will simply return the score of the board if the game gets finished in its current position because there would be not much left.Else it will evaluate each possible move i.e. it will recursively call itself and traverse through each possible child.Now for the best move possible , it will be chosen such that it will lead the board with the max positive score for the player 1, and with the most min score for the player2 of our game.

Below is the example tree which shows the points in the different different situations.
+1 for win, -1 for loss and 0 in the case of draw.



Below is the simplified game tree:



b) **Alpha-beta pruning** : When we use and apply alpha-beta pruning to a standard minimax algorithm it enhances the speed of the minimax algorithm by removing or pruning all the nodes that are possibly not affecting the final decision because it will return the standard move as before. Fasting the algorithm makes it a very useful concept for our game.

Alpha: It denotes the best choice so far for the player MAX as we want to get the highest possible value here.

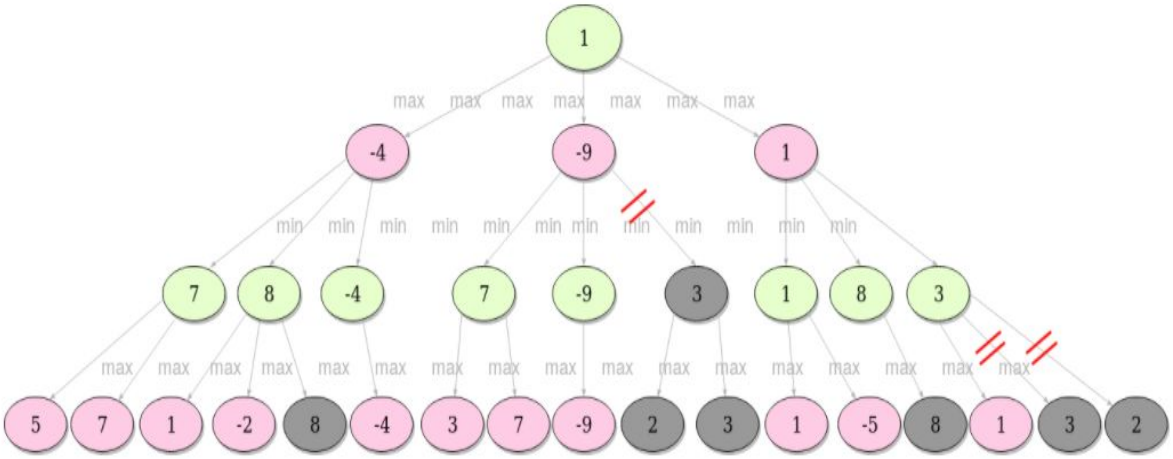
Beta: It denotes the best choice so far for MIN as we want to get the lowest possible value here.

Note: The important thing to note that each and every node in the tree using alpha -beta pruning has to record its corresponding alpha-beta values.It is intuitive to infer that

the alpha value of a node can be updated only when it's MAX's turn and beta value can be updated only when it's MIN's turn.We initially took the alpha values as negative infinity and beta values as positive infinity.

When alpha value is greater than or equal to beta value it means that your opponent's move is not a winning move as compared to your best move so there is no need to further evaluate this move. Hence we do not evaluate that parent node and its subsequent child nodes, or we can say, we prune that complete tree with the parent node satisfying the condition of alpha >= beta.The same pruning condition applies if you are the MIN player and you find the move that yields alpha.

Below is the example of same tree with alpha-beta pruning:



It will be definite that this algorithm will return you a move if we take alpha = -infinity and beta = +infinity , and update these values as we examine more nodes.

Pseudo code for Alpha-beta pruning :

Alpha-Beta(who play game, current state of board,alpha,beta)

if(game end in present state)

return who win the game

Child node = valid passes for player in this state

if(turn of the max)

Check For every child node

sum_total = Alpha-Beta(player pass, current child node,alpha,beta)

if sum_total is greater than alpha = sum_total(can pass it optimally)

if alpha is greater then beta return value of alpha (threshold)

return alpha (optimal pass we have)

else (turn of the min)

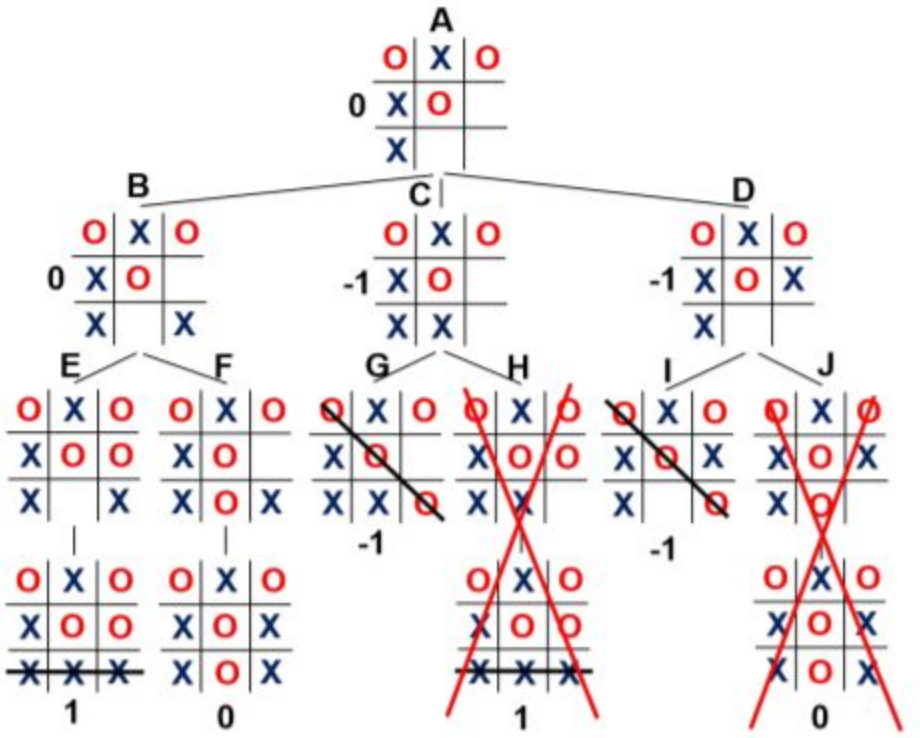
Check for every child node

sum_total is equal to alpha-beta(different player play it's pass,child node,alpha,beta)

if sum_total is less than beta then beta = sum_total(less optimal pass for opposite player)

if value of alpha is greater than or equal to beta then return beta (threshold)

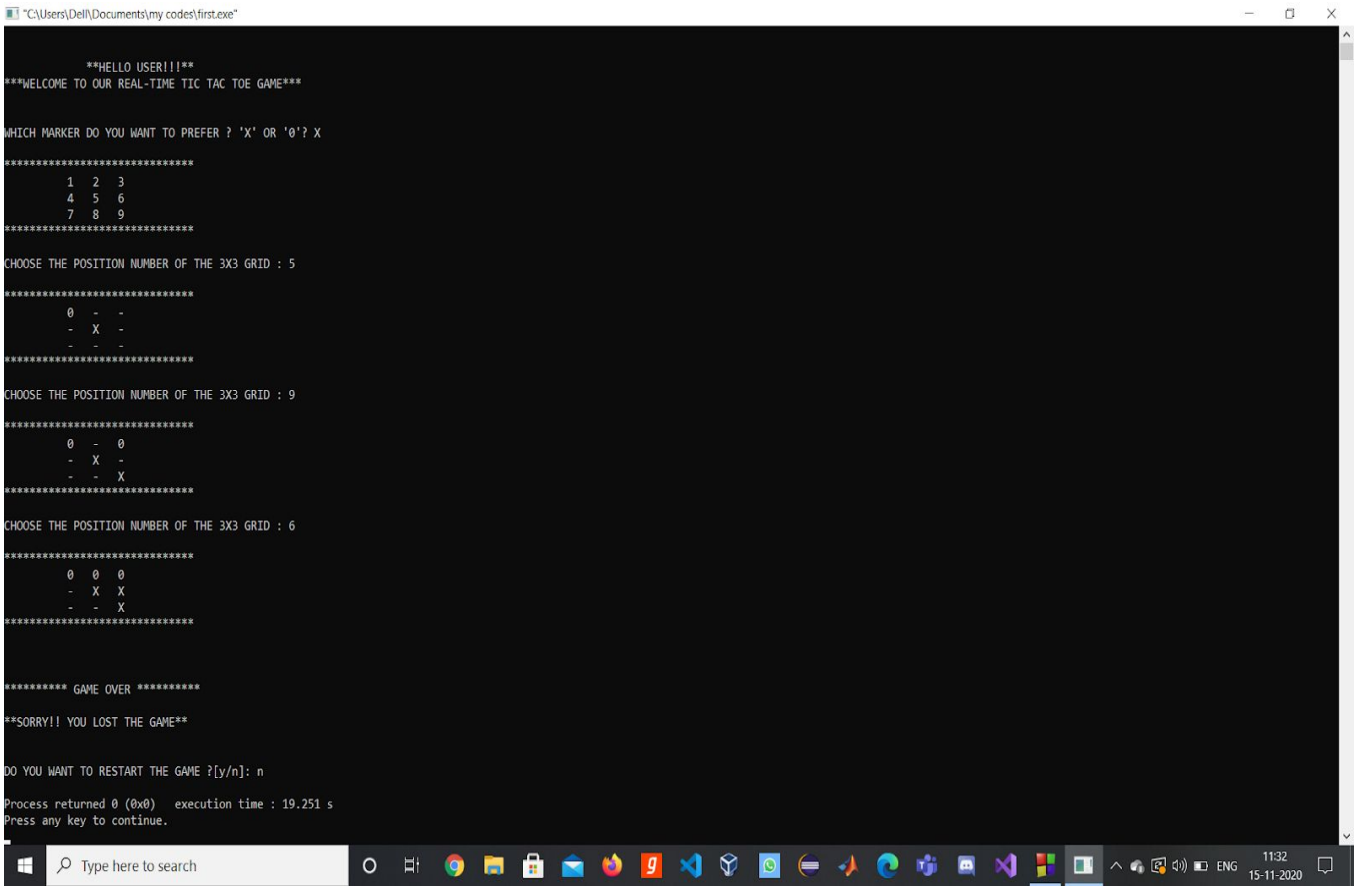
return beta (optimal pass for opposite player).

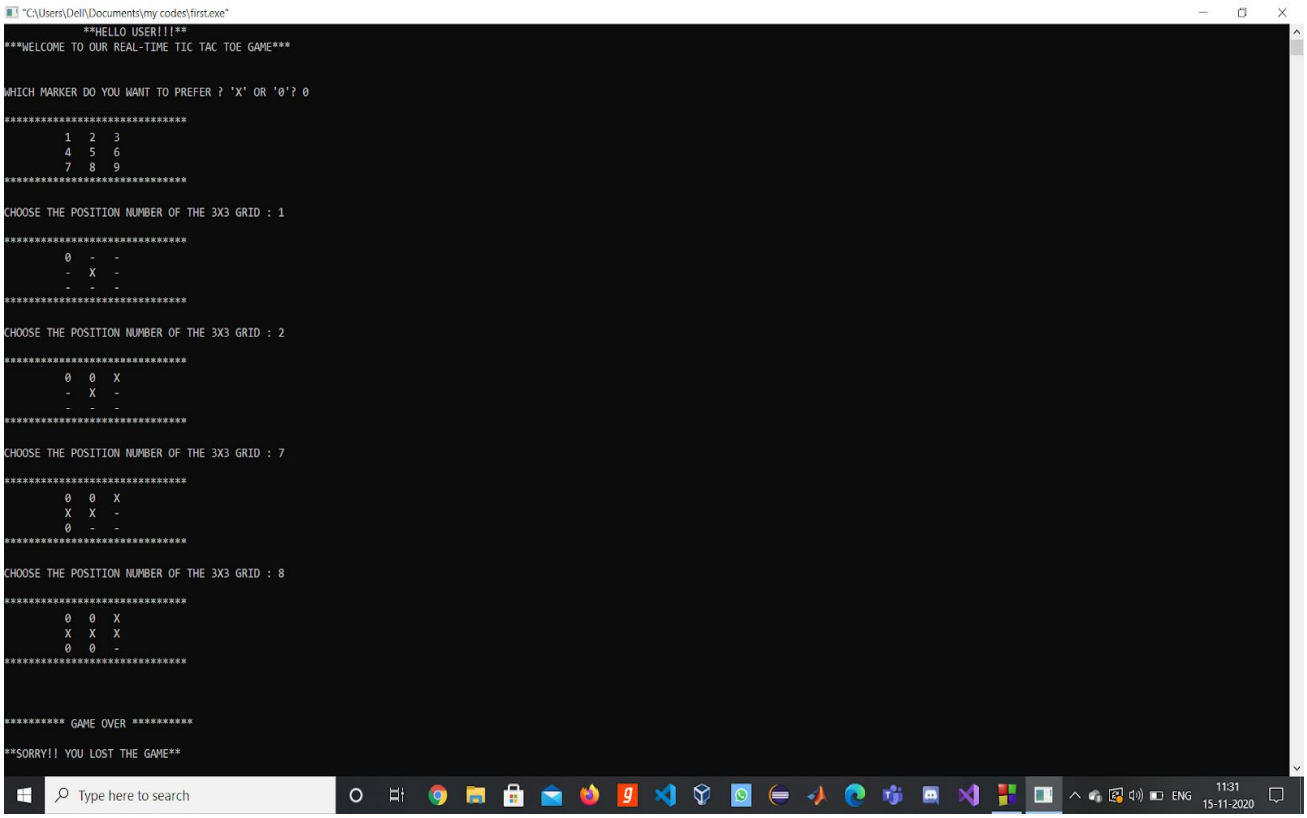


Here is an example of alpha beta pruning in tic-tac-toe. Path A-B-E to the leaf node is the most optimal path and results in a win. Path A-C-H and A-D-J are pruned and are not required to be explored.

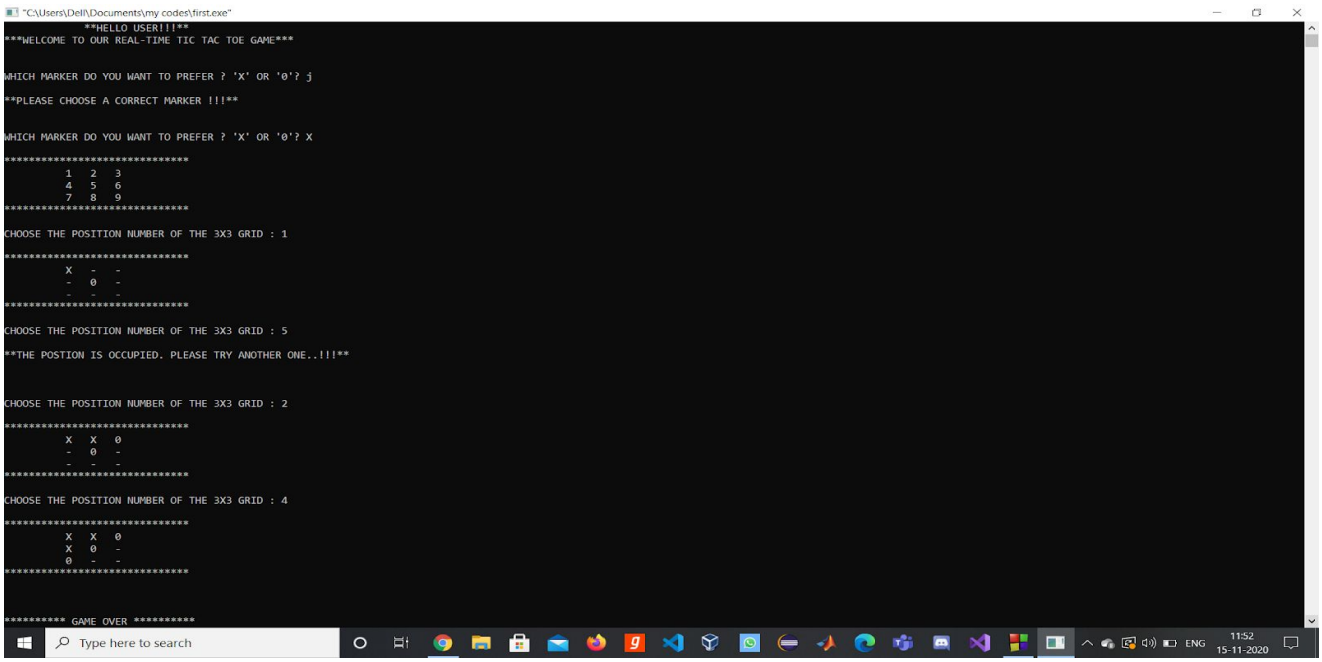
Experiments/Results:

We observed that no matter how well the Player plays the game AI will either win the game or the game draws.If player breaks the last winning position of every winning state it will surely result into DRAW otherwise the AI will WIN as it is using the MINIMAX algorithm with alpha - beta pruning to speed it up, which in turn maximizes the chances of winning of the AI and minimises the chance of the player winning the game. Following are some of the examples to demonstrate this-


"C:\Users\Dell\Documents\my codes\first.exe"
HELLO USER!!!
WELCOME TO OUR REAL-TIME TIC TAC TOE GAME
WHICH MARKER DO YOU WANT TO PREFER ? 'X' OR '0'? X
1 2 3
4 5 6
7 8 9
CHOOSE THE POSITION NUMBER OF THE 3X3 GRID : 5
0 - -
- X -
- - -
CHOOSE THE POSITION NUMBER OF THE 3X3 GRID : 9
0 - 0
- X -
- - X
CHOOSE THE POSITION NUMBER OF THE 3X3 GRID : 6
0 0 0
- X X
- - X
********** GAME OVER **********
SORRY!! YOU LOST THE GAME
DO YOU WANT TO RESTART THE GAME ?[y/n]: n
Process returned 0 (0x0) execution time : 19.251 s
Press any key to continue.
Type here to search
ENG
11:32
15-11-2020



Below example shows error handling i.e when user selects wrong marker or wrong/occupied grid position, it throws an error:



Insights/Discussion/Analysis : Minimax Algorithm traverse a tree according to the depth first search,for a game play tree, minimax algorithm executes recursively until we don't get final outcome minimax gives us optimal decision in game play in adversarial search and contingent strategy is used for optimal solutions. Contingent strategy specifies maximum in the initial state of a particular node and then max will be moving in further hierarchical possible states.So time complexity for a tree with the height h and m valid moves then the time complexity is O(h*m).

If both the player play optimally from a point to the the end the maximum value of a node is utility for being a max in the corresponding state and the minimax is just its usefulness .Maximum value will be stored in max and the minimum value will be stored as min if given a choice. If the size of the board is 3 the total possible moves is 9! and for board size 4 the total possible moves is 16! Increment of board size exponential growth the computational time and that we experience an explosion of computational time but it is the idea for the analysis of games which are done mathematically and for the algorithm which are more likely to be practical.

To implement the minimax algorithm for 4X4 board we've applied the concept of alpha-beta pruning for eliminating the solutions which might not make an impact on the ultimate decision.The Alpha-beta pruning does not change the standard Algorithm.Alpha-beta Pruning removes nodes which is not useful so after implementation of alpha-beta pruning in minimax algorithm we can reduce time complexity up tp O(h*m/2).

$$\text{MINIMAX}(S) = \begin{cases} \text{UTILITY}(S) \text{ , if TERMINAL\_TEST}(S) \\ \max_{a \in \text{ ACTION}(s)} \text{MINIMAX}(\text{RESULT}(S,a)) \text{ , if PLAYER}(s)=\text{MAX} \\ \min_{a \in \text{ ACTION}(s)} \text{MINIMAX}(\text{RESULT}(S,a)) \text{ , if PLAYER}(s)=\text{MIN} \end{cases}$$

Conclusion:

We learnt that by using a minimax algorithm for mathematical aspects in addition with alpha beta pruning concept we can make the process much faster ,and by optimizing the utility function using heuristic function, the 3x3 tic tac toe game can be developed. We concluded that a 3x3 tic tac toe game in the perspective of artificial intelligence, can be developed using these techniques. If we Increase the size of the game to 4x4 or even more, then it would create a huge time complexity issue with the same algorithm and techniques, for which other logics must be further researched.

Acknowledgement:

In this project, we learned to build a tic-tac-toe game between two players where the first player will always have the winning strategy irrespective of the opponent's move, using minimax and alpha-beta pruning algorithms , under the guidance of our respected teachers, Mr. Nitin Kumar, Dr. Poulami Dalapati and Dr. Roshni Chakraborty. It helped us in exploring the field of artificial intelligence in game development very well and also helped us to work collaboratively in a group.

References:

- Peter Norvig & Stuart Russell - Artificial Intelligence : A Modern Approach (3rd edition)
- Artificial Intelligence in Game Development: Aarthi Ellumalai
- George T. Heineman; Gary Pollice; Stanley Selkow. Algorithms in a nutshell. O'Reilly, 2009.
- P. G. P. S. P. Sunil Karamchandani, "A Simple Algorithm For Designing An Artificial Intelligence Based Tic Tac Toe Game".
- https://www.academia.edu: tic tac toe insights
- https://web.stanford.edu: tic tac toe insights and examples.

Demo link (G-drive):

https://drive.google.com/file/d/1SVED4voGqSGqdLMQ2aS7koABgDX2Ge9A/view?usp=sharing

Code link (G-drive):

https://drive.google.com/file/d/1wv3jWT7wykrI53DOlEJy_VH7ufN_9WpW/view?usp=sharing