Tic Tac Toe

USING MINIMAX ALGORITHM

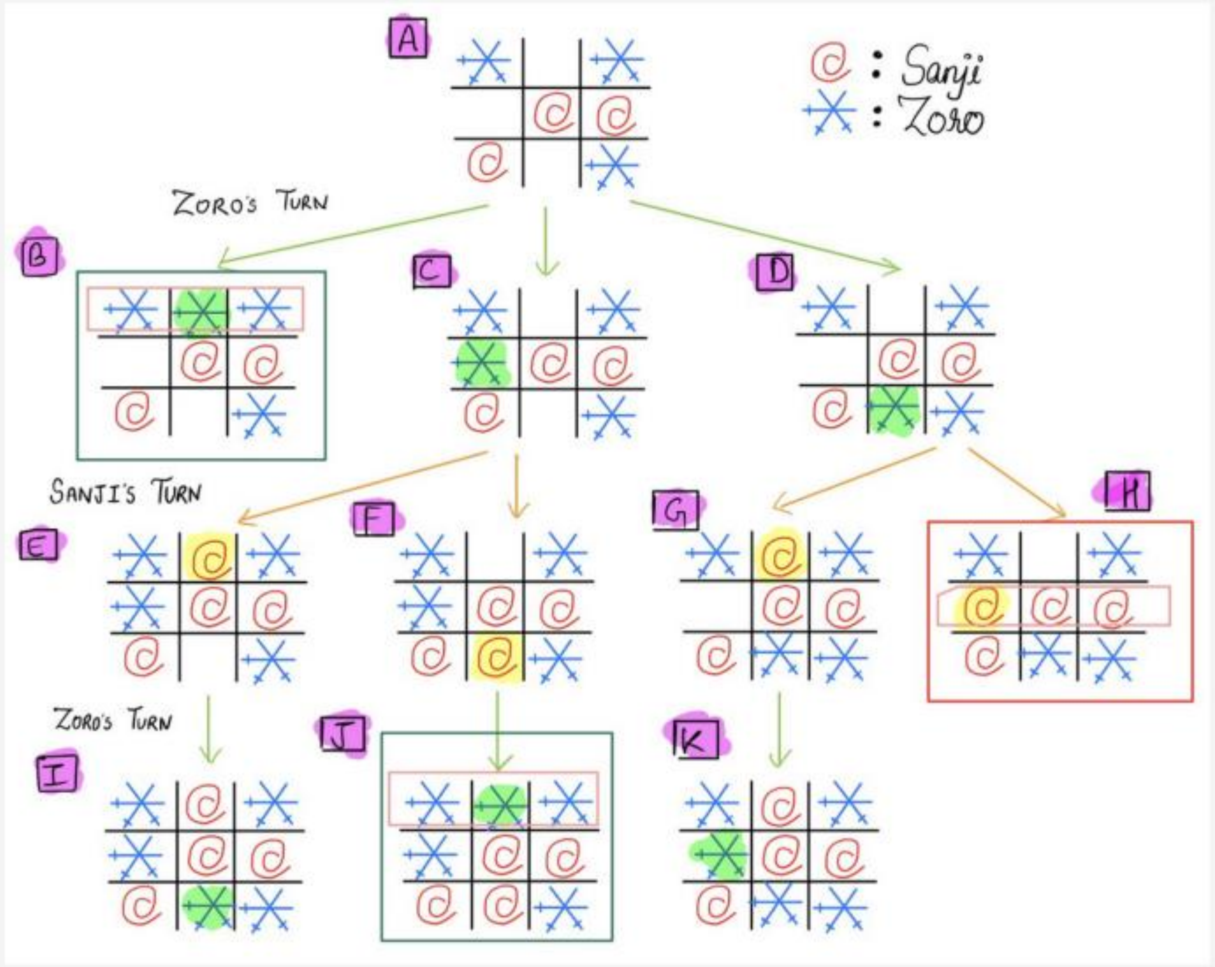
Tic Tac Toe is a simple fun game played with two players. There is a 3x3 grid formed by two vertical and two horizontal lines. This board game is played for fun, meanwhile, it can be the very basis for kids to embrace mathematical knowledge and a strong logical foundation when clearly defined and modelled with mathematics.
A game in which two players alternately put Xs and Os in compartments of a figure formed by two vertical lines crossing two horizontal lines and each tries to get a sequence of three Xs or three Os before the opponent does the same.

Introduction

Minimax is a recursive or backtracking algorithm used in decision-making and game theory to find the optimal move for a player, assuming that your opponent also plays optimally.
Backtracking algorithm is used to find a solution to computational problems in such a way that a candidate is incrementally built towards a solution, one step at a time.
In Minimax the two players are called maximizer and minimizer. The maximizer tries to get the highest score possible while the minimizer tries to do the opposite and get the lowest score possible.

Our Approach

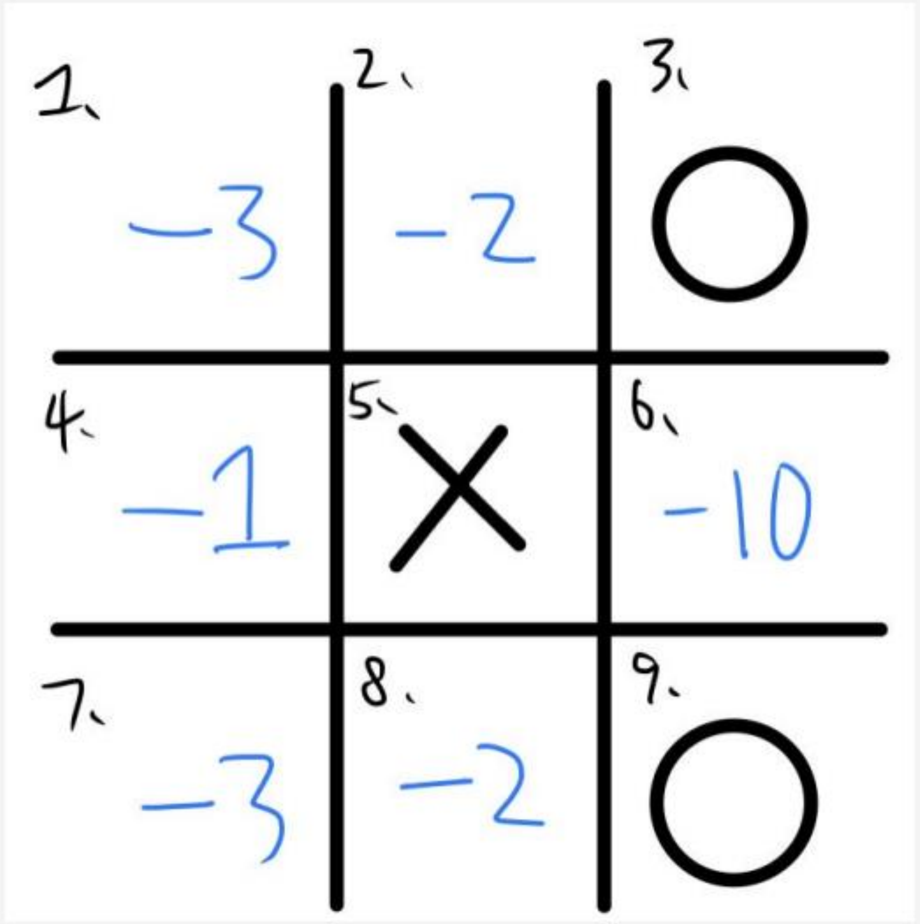
To begin with, we are considering two rivals, Zoro and Sanji. They decided to play the tic-tac-toe. Zoro starts the game by picking ‘X’ and Sanji picks ‘O’ and plays second. In this example, Zoro will try to get the maximum possible score, and Sanji will try to minimize the possible score of Zoro.



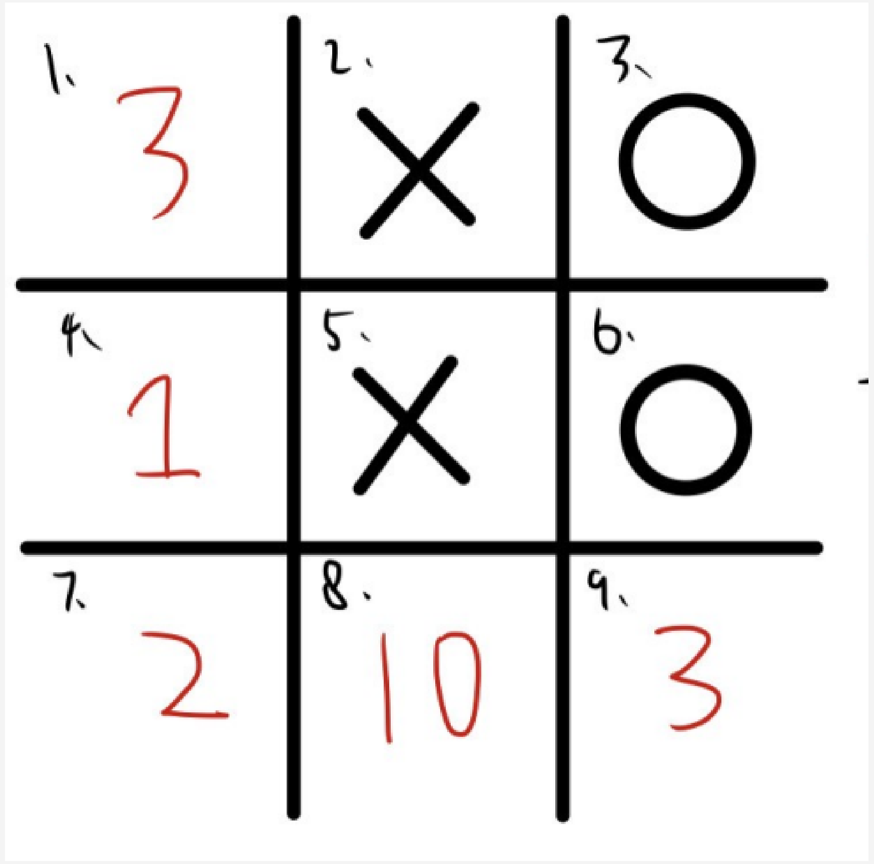
Sample I/O

Min Value

This is the value where the A.I. seeks to minimize the possible loss for a worst-case scenario. For every horizontal, vertical or diagonal lane on the board position, the min value is -1 per lane if the 3 positions have none of your moves. Then, for every additional opponent move on a lane, you -1 (Only if you didn't make a move on that lane). However, if there are 2 opponent moves, your min value becomes -10(Your enemy will win if you don't block that position!).



This is the value where the A.I. seeks to maximize the possible gain for a best-case scenario. For every horizontal, vertical or diagonal lane on the board position, you +1 per lane if the 3 positions have none of your opponent's moves. Then, for every additional move you made on the lane, you +1 again(Only if your enemy didn't make any move on that lane). However, if there are 2 of your moves, your max value becomes +10(You win by making a move there!).



Max Value

```
function minimax(board, depth, isMaximizingPlayer):

    if current board state is a terminal state :
        return value of the board

    if isMaximizingPlayer :
        bestVal = -INFINITY
        for each move in board :
            value = minimax(board, depth+1, false)
            bestVal = max( bestVal, value)
        return bestVal
function minimax(board, depth, isMaximizingPlayer):
```

```
    if current board state is a terminal state :
        return value of the board

    if isMaximizingPlayer :
        bestVal = -INFINITY
        for each move in board :
            value = minimax(board, depth+1, false)
            bestVal = max( bestVal, value)
        return bestVal

    else :
        bestVal = +INFINITY
        for each move in board :
            value = minimax(board, depth+1, true)
            bestVal = min( bestVal, value)
        return bestVal
```

Pseudocode

Predicting Video's Sentiment using Text Extraction

Classifying Memes, texts, tweets etc into the severity of Fake or Real

Giving a detailed synopsis about the content present on the websites

Future Roadmap

About Us

Thank You!