TIC TAC TOE AI MODEL

Using reinforcement learning

NAME: SRI SAI VIJAYA ADITYA NITTALA

ROLL NO.: 177163

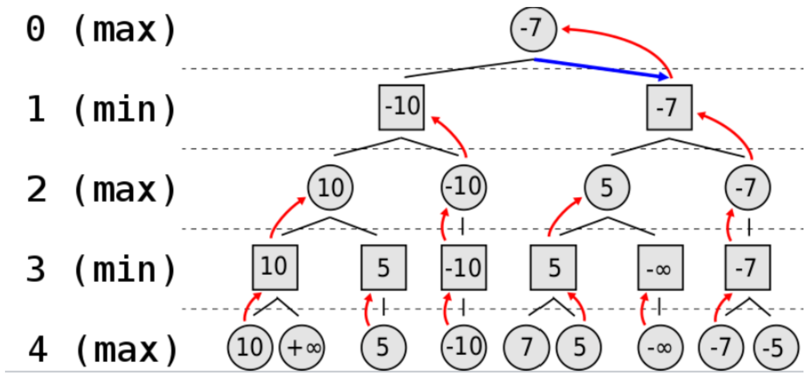
Section: A

The **minimax** algorithm is a recursive algorithm for choosing the next move in an n-player game (usually a 2-player game). A value is associated with each position or state of the game. This value is computed by means of an evaluation function that indicates how good it would be for a player to reach that position.

The pseudocode for the algorithm is given below:

```
function minimax(node, depth, maximizingPlayer) is
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value := −∞
        for each child of node do
            value := max(value, minimax(child, depth − 1, FALSE))
        return value
    else (* minimizing player *)
        value := +∞
        for each child of node do
            value := min(value, minimax(child, depth − 1, TRUE))
        return value
```

With the image below, the algorithm is explained:



- Circles are for the maximizing player, that is the player running the algorithm.
- The squares are for the minimizing player.
- At each step, the maximizing player wants to maximize his/her winning and the minimizing player wants to minimize.
- Upon reaching the greatest depth, that is, final board state, the evaluation function is called to get a score on the board state. Depending on whose turn it was, the minimum or maximum value of two child nodes are returned.
- This continues until a value is returned to the current board state, which then helps the computer pick a move.

minimax() set-up for Tic Tac Toe game:

- Depth for a nxn tic tac toe board is when all nxn cells are evaluated.
- The terminal board state is when all cells of the board are filled.
- The **Human** is given a value of -1.
- The **Computer** is given a value of +1.
- Evaluation:
    - If **Computer** wins: +1
    - If **Human** wins: -1
    - If **draw**: 0

The pseudocode is given below:

```
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
```

NOTE:

The code attached below is for a **3x3** tic tac toe board but the same functions with a few modifications (mentioned as comments in the code) will work with a **4x4** board. Due to insufficient computing power, a game of 4x4 tic tac toe with minimax() on my computer took about 20 minutes.