**Introduction**

To solve games using AI, we will introduce the concept of a game tree followed by a minimax algorithm. The different states of the game are represented by nodes in the game tree, very similar to the above planning problems. The idea is just slightly different. In the game tree, the nodes are arranged in levels that correspond to each player's turns in the game so that the “root” node of the tree (usually depicted at the top of the diagram) is the beginning position in the game. In tic-tac-toe, this would be the empty grid with no Xs or Os played yet. Under root, on the second level, there are the possible states that can result from the first player’s moves, be it X or O. We call these nodes the “children” of the root node.

Each node on the second level would further have as its children nodes the states that can be reached from it by the opposing player's moves. This is continued, level by level, until reaching states where the game is over. In tic-tac-toe, this means that either one of the players gets a line of three and wins, or the board is full and the game ends in a tie.

**What is Minimax?**

Minimax is an artificial intelligence applied in two-player games, such as tic-tac-toe, checkers, chess, and go. These games are known as zero-sum games, because in a mathematical representation: one player wins (+1) and the other player loses (-1) or both of anyone not to win (0).

**How does it work?**

The algorithm searches, recursively, the best move that leads the AI player to win or not lose (draw). It considers the current state of the game and the available moves at that state, then for each valid move it plays (alternating \*min\* and \*max\*) until it finds a terminal state (win, draw or lose).

**Members**

1. 200953170, Priyanshu Pranay

2. 200911245, Biprajit Deb





