# Project updates

Team: Malhar Bhise, Suhani Jain and Yuvna Jain

---

## Week 1: 27th October 2023

| Week number | Concept |
|---|---|
| 1 | Tic tac toe bot using the minimax algorithm |

### Description

To begin, we built tic tac toe, a simple enough version to be played between a user (player) and the CPU (bot). It had all the basics, a game board, win checker, player and CPU move functions, etc. With this, we moved on to building the bot to play the game.

We understood the minimax algorithm in order to implement the AI bot for tic tac toe. The minimax algorithm optimizes decision-making in two-player games like Tic-Tac-Toe by exploring all possible moves and ensuring the selection of the best move for the bot while assuming optimal play from the opponent. This guarantees that the bot will either win or force a draw, making it a powerful strategy for games that have a finite number of possible states. The maximizing and minimising concept is a fundamental part of the minimax algorithm used in two-player games. In the context of games like Tic-Tac-Toe, it represents the two opposing players, typically referred to as Max (the player trying to maximize their score, AI bot in this context) and Min (the opponent, the player in this context). In a game like tic tac toe, there were very few possible moves for the bot to play making it easy for us to understand and implement. It set a good base for bot simulation and made us understand the algorithm in a game setting.

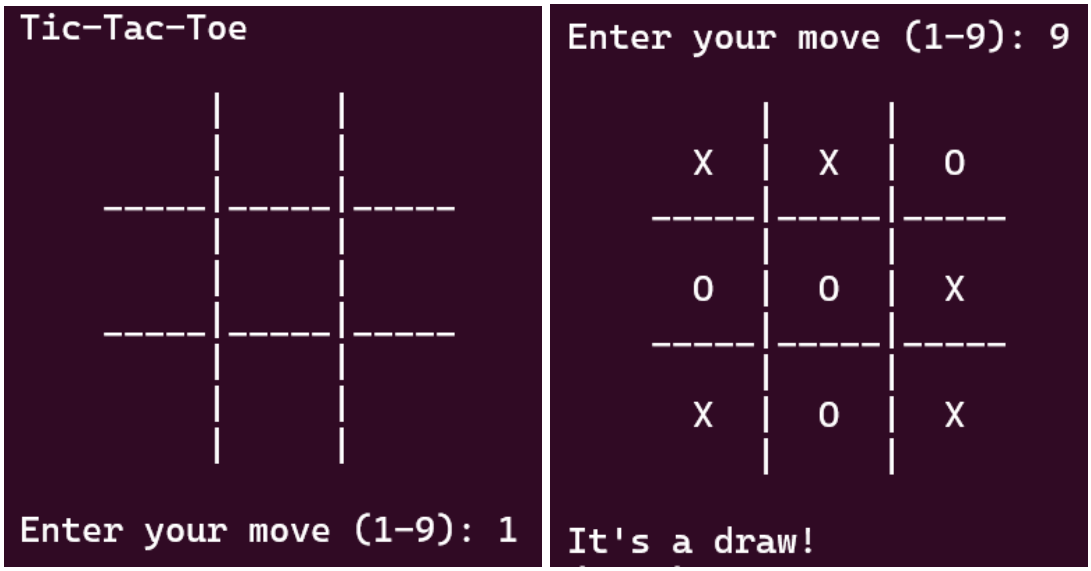
## Python CPU code

Attached below is the code for the AI bot simulation and the user input.

The code has several functions including initialization of the board, player input, AI bot input using the minimax algorithm, a function checking for winning conditions, and the main game loop.

https://drive.google.com/file/d/1Rwo2IeunHDPseHoluM5fiCd7CZoeW6hj/view?usp=sharing

## Simulation




## Takeaway

We developed our understanding of the basics of creating AI for games with a sort of “Hello world” in game bot development. We understood a simple recursive algorithm and were able to implement it successfully. In the process, we learnt the difficulties faced in a problem like this, such as falling in the loop of infinite recursion, missing arguments, getting parameters wrong, etc, and we will be better equipped to face our next task.

# Week 2 plan

- In the next week, we aim to complete the AI simulation for a slightly trickier game - connect 4. This will be using the minimax algorithm along with alpha-beta pruning and transition tables.
- Additionally, we want to understand and divide tasks for our chess AI simulation which is the main task of this project.

# Week 2: 10th November 2023

| Week number | Concept |
|---|---|
| 1 | Tic tac toe bot using the minimax algorithm |
| 2 | Connect 4 bot using minimax + alpha-beta pruning |

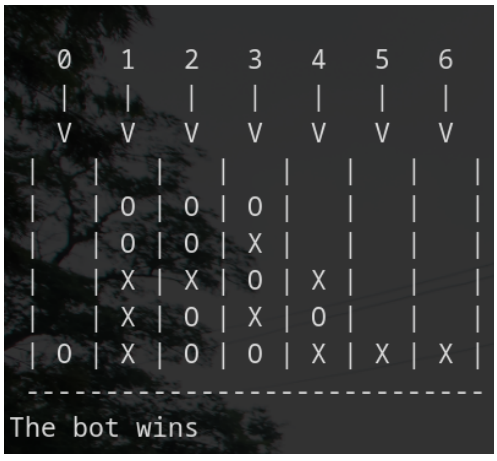
## Description

With a simple yet solid foundation in building games and game bots, we decided to tackle our second hurdle, Connect4. While building the game was easy, implementing the MiniMax algorithm for the bot proved to be a challenge. With the number of game states being significantly larger, traversing the entire game tree for connect4 was not feasible. Hence, we implemented alpha-beta pruning to neglect unnecessary game states, and traverse only the viable ones. Although we had planned to start with building the chess game, we were not able to as the Connect4 bot itself took longer than we anticipated

## Python CPU code - Connect 4

Attached below is the Python code for our Connect4 bot.
https://drive.google.com/file/d/1eFFreQy8CQ0s0p0tw_GfTw8UF03ecX5Y/view?usp=sharing

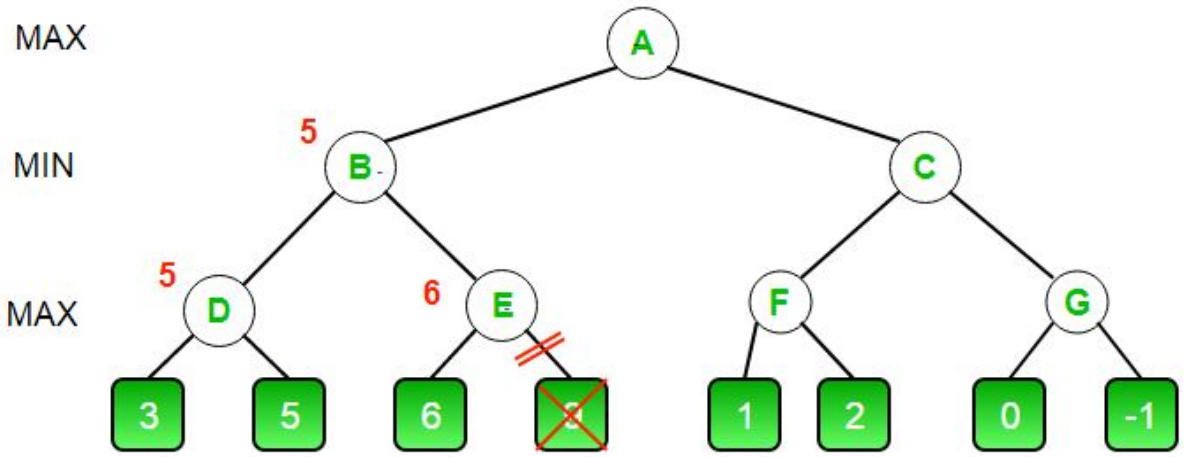
## Simulation



## Mathematics

After having built the game itself, we started out by just using the same minimax algorithm to decide our bot moves, this time however with a more sophisticated evaluation function to evaluate different game results. We soon realized that this strategy did not work as we weren't able to traverse through all game states (4,531,985,219,092 according to Wikipedia). After some fine-tuning and optimization, we developed our algorithm such that our bot would choose his move by executing the minimax algorithm up to a depth of 5 (looking 5 moves ahead), and also implementing alpha-beta pruning, which involves skipping evaluation of redundant game states to save computational power and time.

As seen from the image below, while evaluating all game states in the game tree, the evaluation of the state which returns "9" becomes redundant as the maximising player (E) will not choose anything less than 6 and hence the minimising player (B) will not choose anything other than 5 (as 6 > 5). Hence, "pruning" the node that evaluates to 9 or skipping its evaluation will lead to the same result as if you had evaluated it. For larger trees (such as in our game), this pruning becomes more and more valuable in terms of saving computational and time savings.




*source: geeksforgeeks*

## Results

Although none of our team members are by any means experts at connect4, the bot was able to defeat each one of us every time consistently. To compensate for our lack of skill in the game, we tested our bot against a few bots online, and to our surprise, it performed significantly well. Against each online bot, we played 10 games, 5 as the first player to make a move and 5 as the second. Here were the results:

Against this bot, we won all the times we started first, and all but one match playing second, with the last game ending in a draw. Our final score was 9/10
https://replit.com/@JackIngram2/impossible-connect-4-bot

Against this bot, we lost all our games but 2, which ended in draws. However, even in the games we lost, our bot managed to reach the end game and lost on the final few moves. We assume that this is because this solver has precoded many of its moves (called transition tables) as it was able to compute its moves way too fast according to us for a web-based platform.
https://connect4.gamesolver.org/

We are aware of ways we can improve our bot, by creating transition tables for frequently accessed game states, but since we already have a pretty good bot and this would take a lot of time to implement, we decided to move on to our next step of the project.
From this stage, we were able to learn about the challenges of tackling a game large enough that it cannot be covered by a full tree search. We had to tune changes and face a lot of bugs and challenges along the way. At the end of the day, however, we were able to create a formidable opponent in connect4 against any opponent (including perfect bots) and were able to learn and implement many techniques used to traverse large game trees and evaluate them.

## Week 3 plan

Over the next week, we have a significant amount of work for us.

- We plan to start building our chess game engine
- We need to start planning out the strategies we will need to employ to build our bot.

We plan to split the work evenly and work as much as we can.

## References


- https://www.mygreatlearning.com/blog/alpha-beta-pruning-in-ai/
- https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/
- https://www.hackerearth.com/blog/developers/minimax-algorithm-alpha-beta-pruning/