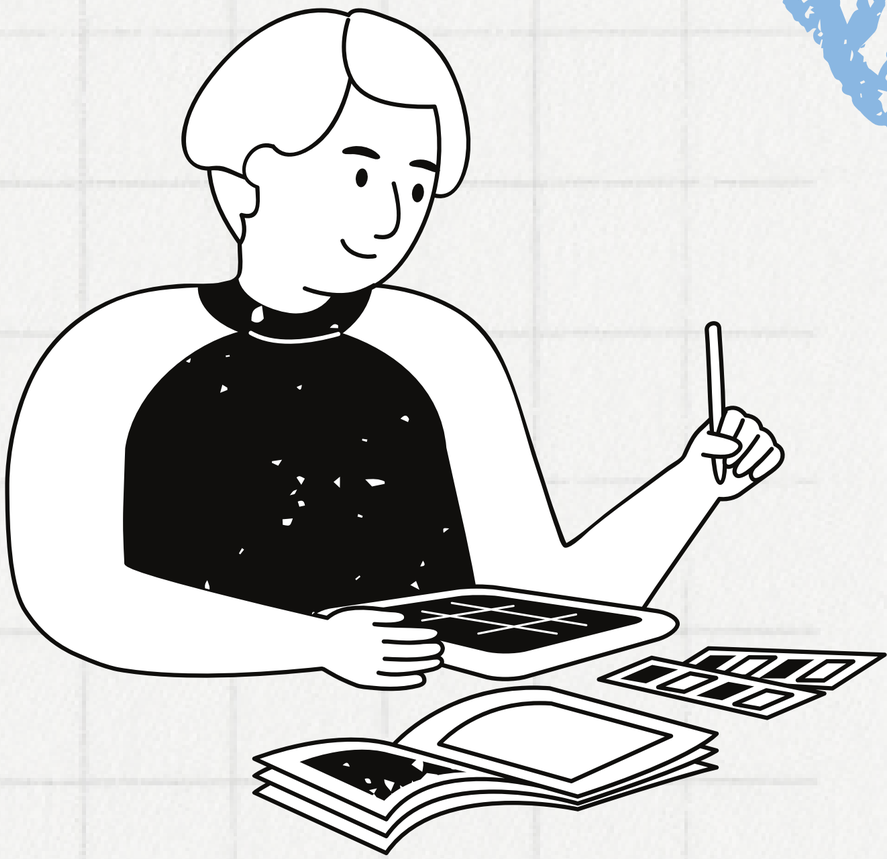
Tic Tac Toe Using Bruteforce and Heuristic Apprcoch

Presented by Joel Pawar

Problem Statement

1. Tic Tac Toe Using Bruteforce approch
2. Tic Tac Toe Using Heuristic Approch



What is Bruteforce Technique

the bruteforce approach is like trying every possible option until you find the right one, without using any clever shortcuts or strategies.
Example:reallife example of a bruteforce approach is searching for a lost item in your house. You might start by checking every room, opening every drawer, and looking under every piece of furniture until you find the item. This method doesn't involve any special strategy or shortcuts; you're simply systematically checking every possible location until you find what you're looking for. While this approach may eventually lead you to the lost item, it can be timeconsuming, especially if you have a large house with many rooms and hiding spots.

TIC TAC TOE USING BRUTEFORCE APPROCH

- **Bruteforce Usage:**
- **Minimax Algorithm: Bruteforce is used in the minimax algorithm to recursively evaluate all possible future game states.**
- **Generate Moves: Bruteforce generates all possible next moves for a given board position to exploe all potential game outcomes.**
- **Move Evaluation: Bruteforce evaluates each move by simulating future game states to determine the best move for the computer.**
- **Move Table Usage:**
- **Optimization: The move table stores evaluated scores for each board position to avoid redundant calculations.**
- **Caching Scores: Previously computed scores are stored in the move table to prevent recalculating them during subsequent moves.**
- **Efficiency: By storing and retrieving scores from the move table, the algorithm improves efficiency and reduces computation time.**
-

What is Heuristic Technique

Heuristic approach is a problemsolving strategy that uses practical experience and rules of thumb to find solutions quickly, even if they may not be optimal. It's like using shortcuts based on common sense rather than exhaustive search. Example: Imagine you're trying to find the fastest route to work. Instead of examining every possible road and calculating the exact travel time, you might use a heuristic approach by taking the main highway because it's usually faster during rush hour based on past experience.

TIC TAC TOE USING HEURISTIC APPROACH

- This heuristic_move() function implements strategies for the computer's move:
- Winning Move: Checks for an immediate win.
- Blocking Opponent: Prevents the opponent from winning.
- Forking Strategy: Creates opportunities for multiple winning paths.
- Center Strategy: Occupies the center or corners strategically.
- Random Move: Chooses a random empty cell when no strategic moves are available.

Results

main.py

Save Run

```python
import random

# Function to print the Tic Tac Toe board
def print_board(board):
    for row in board:
        print(' '.join(row))

# Function to check if any player has won the game
def check_winner(board, player):
    # Check rows and columns
    for i in range(3):
        if all(cell == player for cell in board[i]) or all
            (board[j][i] == player for j in range(3)):
            return True
    # Check diagonals
    if all(board[i][i] == player for i in range(3)) or all
        (board[i][2 - i] == player for i in range(3)):
        return True
```

Shell

```
O X O
- X -
O - -
Computer's move:
O X O
X X -
O - -
Enter row (0, 1, 2): 2
Enter column (0, 1, 2): 1
O X O
X X -
O O -
Computer's move:
O X O
X X X
O O -
Computer wins!
>
```

Results

main.py

Save Run

```python
import random

def print_board(board):
    for row in board:
        print(' '.join(row))

def check_winner(board):
    # Check rows, columns, and diagonals for a winner
    for i in range(3):
        if board[i][0] == board[i][1] == board[i][2] != '-':  # Rows
            return board[i][0]
        if board[0][i] == board[1][i] == board[2][i] != '-':  # Columns
            return board[0][i]
    if board[0][0] == board[1][1] == board[2][2] != '-':  # Diagonal 1
        return board[0][0]
```

Shell

```
User's turn (O)
Enter row (0, 1, 2): 1
Enter column (0, 1, 2): 2
O O X
X X O
O - -
Computer's turn (X)
O O X
X X O
O - X
User's turn (O)
Enter row (0, 1, 2): 2
Enter column (0, 1, 2): 1
O O X
X X O
O O X
It's a draw!
>
```

https://replit.com/@joelpawarwork/AI-Presentations#bruteforce.py

https://replit.com/@joelpawarwork/AI-Presentations#heuristic.py