CHAPTER 1

Problem Definition

- ## Introduction

Tic-tac-toe is a fun game that you can play anytime and anywhere as long as you have a piece of paper, a pencil, and an opponent. Tic-tac-toe is a zero-sum game, which means that if both players are playing their best, then neither player will win. However, if you learn how to play tic-tac-toe and master some simple strategies, then you'll be able to not only play but win the majority of the time.

Basic Rules-:

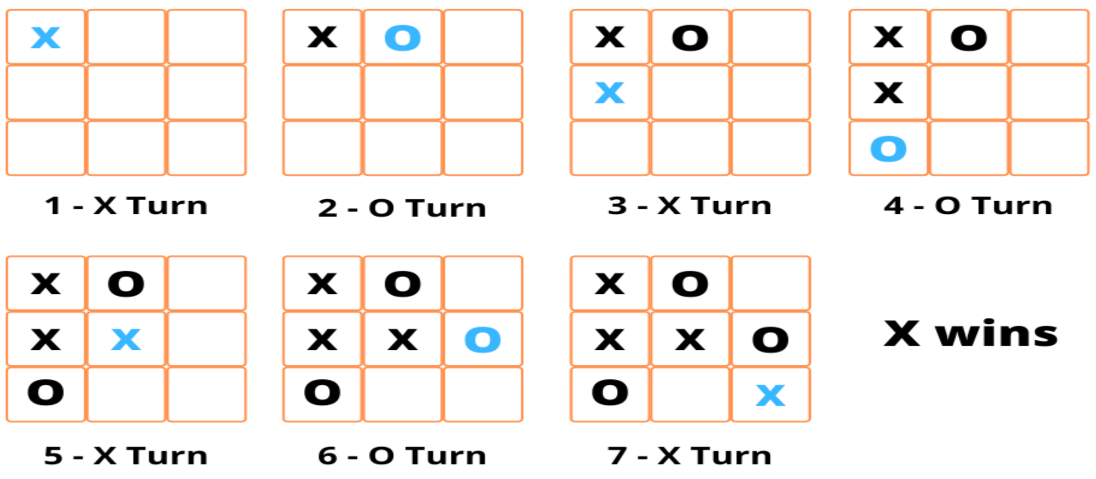
1. The game is played on a grid that's 3 squares by 3 squares.
2. You are X, your friend (or the computer in this case) is O. Players take turns putting their marks in empty squares.
3. The first player to get 3 of her marks in a row (up, down, across, or diagonally) is the winner.
4. When all 9 squares are full, the game is over.

- ### Problem Statement

The Tic –Tac-Toe game is to be played between two people (in this program between HUMAN and COMPUTER). One of the players chooses ‘O’ and the other ‘X’ to mark their respective cells. The game starts with one of the players and the game ends when one of the players has one whole row/ column/ diagonal filled with his/her respective character (‘O’ or ‘X’). If no one wins, then the game is said to be draw.

CHAPTER 2
Problem Explanation

Problem Explanation



The game started with “X” of computer then “O” is played by the Human. For the next turn computer will check for his winning opportunities and place the “X” on grid. After that computer will ask Human for the next move and this cycle will go on until game has some result.

In the game computer will try to find the best move (optimal move) for win.

Computer will ask the Human for the next move and check the cases that where computer can place the “X”. It will check all the cases for his winning and also chances of victory for Human. Computer will check rows, columns and diagonals for win. Computer will check corner space, centre space and sides for move and then it will place his move. If it is not able for any move then it will check the diagonals. If the diagonals are filled with “X” then system wins. If Computer is not able to have some move, then it will check whether the game is ends as a draw or won by Human.

CHAPTER 3

Design Techniques

- ## Backtracking

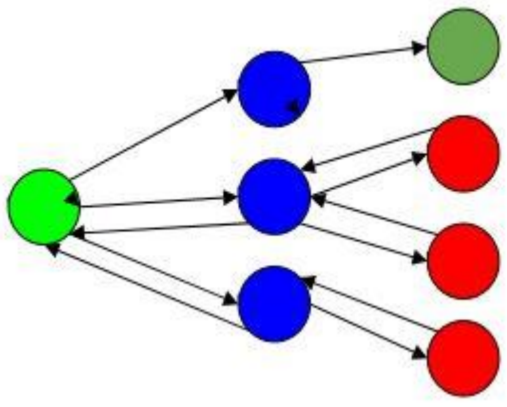
Backtracking is a technique based on algorithm to solve problem. It uses recursive calling to find the solution by building a solution step by step increasing values with time. It removes the solutions that doesn't give rise to the solution of the problem based on the constraints given to solve the problem.

Backtracking algorithm is applied to some specific types of problems,

- Decision problem used to find a feasible solution of the problem.
- Optimisation problem used to find the best solution that can be applied.
- Enumeration problem used to find the set of all feasible solutions of the problem.

In backtracking problem, the algorithm tries to find a sequence path to the solution which has some small checkpoints from where the problem can backtrack if no feasible solution is found for the problem. Its traverse tree by DFS (Depth First Search). The Depth First Search algorithm is a recursive algorithm that uses the idea of backtracking. It involves exhaustive searches of all the nodes by going ahead, if possible, else by backtracking.

Example,



Here,

Green is the start point, blue is the intermediate point, red are points with no feasible solution, dark green is end solution. Here, when the algorithm propagates to an end to check if it is a solution or not, if it is then returning the solution otherwise backtracks to the point one step behind it to find track to the next point to find solution.

- ## Minimax-Algorithm

The MiniMax algorithm is a recursive algorithm used in decision-making and game theory. It delivers an optimal move for the player, considering that the competitor is also playing optimally. This algorithm is widely used for game playing in Artificial Intelligence, such as chess, tic-tac-toe, and myriad double players games.
In this algorithm, two players play the game; one is called **‘*MAX*’**, and the other is **‘*MIN*.**’ The goal of players is to minimize the opponent’s benefit and maximize self-benefit. The MiniMax algorithm conducts a depth-first search to explore the complete game tree and then proceeds down to the leaf node of the tree, then backtracks the tree using recursive calls.

Each state is denoted by the positions occupied by the letters of the two players in a 3x3 matrix and other empty places.
1. ***Initial State:*** An empty 3x3 matrix.
2. ***Intermediate State:*** Any arrangement of a 3x3 matrix obtained after applying valid rules on the current state.
3. ***Final State:*** Same letters in a complete row or complete column, or complete diagonal wins the game.
4. ***Rules:*** The players will get turns one after the other; they can mark their letters on the empty cell.