Term Project

Design and Analysis of Computer Algorithms
(CS 575-20) - Summer 2022

Tic-Tac-Toe (Puzzle Game)
Minimax Algorithm

Name: Nagalakshmi Prasanna Pujita Bodapati
B-Number: B00929285
Language used: C Language

Submitted By

Pujita Bodapati
B00929285

Problem Statement

The game is implemented in a 3x3 grid where two players are playing against each other to win by making their optimal moves. In this puzzle one opponent is a computer (considered as a bot) and the other opponent is a player (a person who gives manual inputs). Computer moves are shown as ‘X’ and player moves are ‘O’. Each player can make their move against each other alternatively. The game ends when the complete puzzle (3x3 grid) is filled with the player moves or when one of the players wins. There can be 3 possible outcomes - ‘Draw’, ‘Computer Wins’ or ‘Player Wins’.

The outcome can be decided by the position of the ‘X’ or ‘O’ in the puzzle. There will be a winner when the alignment with the same character (‘X’ or ‘O’) matches either - horizontally, vertically or diagonally. There are many conditions in the puzzle where moves made by one player cannot be overridden by the other or once a move is made, one cannot undo it.

This puzzle is implemented using the Minimax algorithm where computer moves are coded in such a way that it will check all possible moves made by the opponent and will decide the best move (optimal solution) against the opponent where the best case scenario is when the ‘computer Wins’ and worst case is ‘Draw’.

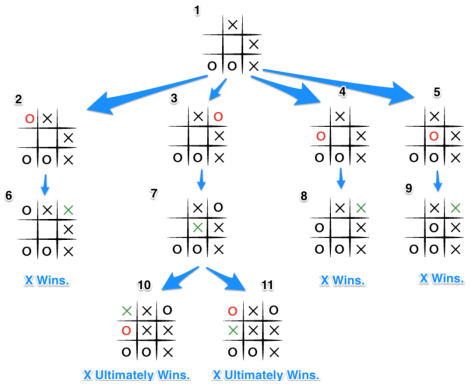
Objective

The main objective of this puzzle is to design and implement a minimax algorithm in such a way that Computer will make its best move by checking all the optimal solutions possible and win against a human player. This is done by following all the rules for the Tic-Tac-Toe puzzle and giving the best outcome which includes a win for the computer.

Algorithm Description

Minimax Algorithm - Tic-Tac-Toe Puzzle (3x3 Grid)

A minimax algorithm is a recursive algorithm which can be used mainly in game-theory as well as decision making. When using this algorithm it provides an optimal solution (moves) considering the opponent is always playing their best (optimal) moves. This algorithm can be mainly used for playing games which includes backtracking or it also can be used in AI. Such games may include - Chess, Tic-Tac-Toe (mainly two player games).



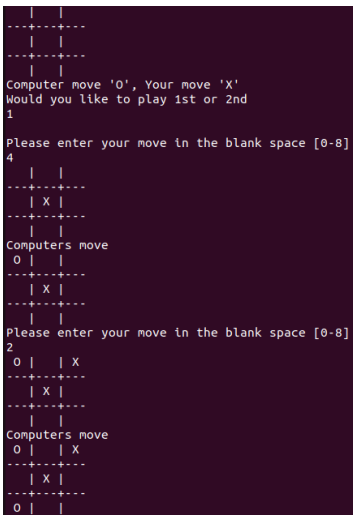
The minimax algorithm assumes that there will be two players, one is min and the other is max (in this case computer vs a human player). Initially, there will be an option for the main player to select if he wants to play 1st or 2nd. Based on the option chosen by the player the computer will make its move against the player using a recursive minimax algorithm. The puzzle board will be of 3x3 grid. Initially the values in the array (board) will be 0. Once the player has made his/her move by selecting the position in the grid where the range of values can be from 0-8, the corresponding value in the array will be updated to -1 (since the min score will be -1). Later, the algorithm will check all the moves that are available and will update the value to 1 in the array board in such a way that either this move will block the player from winning or it will make its best move to get an optimal solution.

The game will be completed if either one of the players wins or it's a draw. The final outcome can be decided by the position the values are placed in the puzzle board. Based on the original rules of the Tic-Tac-Toe puzzle one can win the game if either the characters are aligned horizontally, vertically or diagonally. So, based on the moves made by both computer and player the outcome can be decided in the 3x3 puzzle grid by considering the positions from the board. The values will be checked by taking the horizontal, vertical and diagonal positions which are {0,4,8}, {2,4,6}, {0,1,2}, {3,4,5}, {6,7,8}, {0,3,6}, {1,4,7}, {2,5,8}.

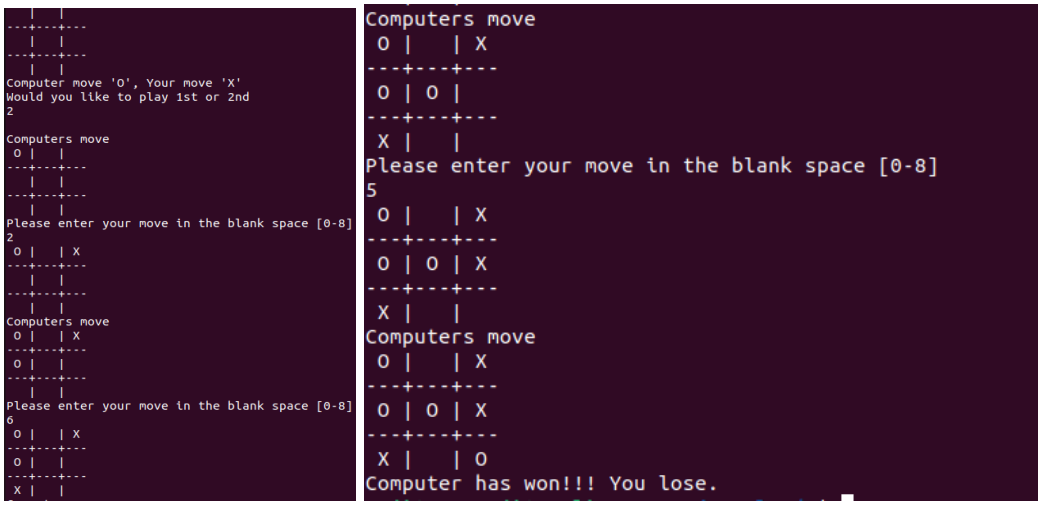
So, if any one of the combinations will match with the same character then the game will end and if the values are -1 then the player will win or if the values are 1 then the computer wins. If both computer or player moves are not aligned and there is no move that can be made further in such a case it's a draw. The complete algorithm is implemented in the terminal and moves are taken as input in the terminal by the player.

Program Output

Case 1 - Draw



Case 2 - Computer Wins



Time Complexity - Minimax Algorithm

As the Minimax algorithm is mainly used to search game trees, it automatically assumes that the players involved will make alternate moves. This algorithm will make use of a utility function where the values with max score are considered for player 1 and minimum score values are considered for player 2. So, the main purpose of Player 1 (Max's) is to select a move such that it uses the maximum of the utility function where the Player 2 (Min's) purpose is to select a move that uses the minimum of the utility function.

The time complexity of this algorithm: Minimax - O(b^m)
The space complexity of this algorithm: Minimax - O(bm)

Where b - No. of moves made at each point, m - Maximum depth of the tree.

References

https://towardsdatascience.com/lets-beat-games-using-a-bunch-of-code-part-1-tic-tac-toe-1543e981fec1

https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/

https://www.javatpoint.com/mini-max-algorithm-in-ai

https://levelup.gitconnected.com/minimax-algorithm-explanation-using-tic-tac-toe-game-22668694aa13

https://www.youtube.com/watch?v=RYUBVSjpli8

Actual Online Game

https://playtictactoe.org/