PROJECT

IMPLEMENTING TIC TAC TOE USING GAMES TREES AND THE MINIMAX ALGORITHM

Developed by:

RHEEYA UPPAAL
2nd Year, Computer Science

TABLE OF CONTENTS

Abstract

This project focuses on creating a game of ‘Tic Tac Toe’. The project is a single player game where a human player plays against the computer. The software has been designed in such a way that the computer is unbeatable. The game can only end in the computer winning or in a draw.

The project makes use of game trees for the computer to predict or “look ahead” further into the game and decipher what moves can be made for the next certain number of turns.

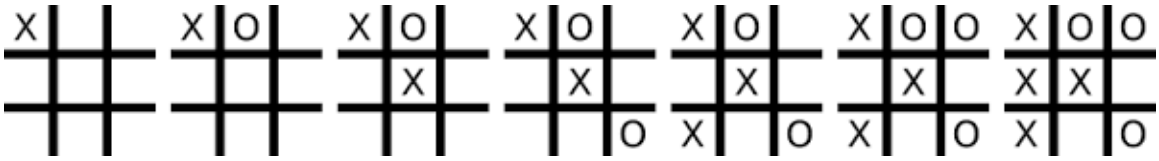
The computer then evaluates the game tree to select the best possible move for the player’s move. The computer re-evaluates all moves and re-selects the best move every time the player makes a move.

In further revisions of the project, alpha-beta pruning (which reduce the number of nodes in the game tree by eliminating repetitions) and partial game trees (where the computer only generates ‘n’ future moves for every move the player makes, instead of making the complete game tree) can be implemented.

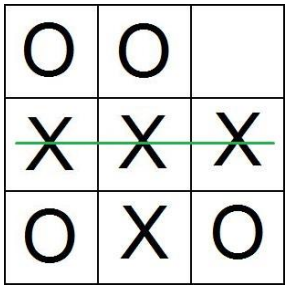
1. Introduction

1.1 Overview

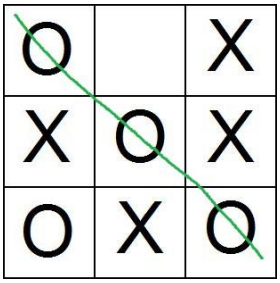
Tic-tac-toe, also called Noughts and crosses (in the Commonwealth Countries), or Xs and Os (in Ireland) is a paper-and-pencil game for two players: *X* and *O*, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three respective marks in a horizontal, vertical, or diagonal row wins the game.



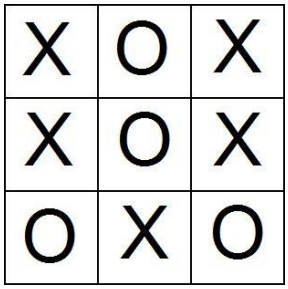
An example of a game of Tic Tac Toe



Game where X wins



Game where O wins



Game resulting in a draw

Players soon discover that best play from both parties leads to a draw (often referred to as cat or cat's game). The game can be generalized to an m,n,k-game in which two players alternate placing stones of their own color on an *m*×*n* board, with the goal of getting *k* of their own color in a row. Tic-tac-toe is the (3,3,3)-game.

1.2 Purpose

Tic Tac Toe closely relates to the branch of artificial intelligence that deals with the searching of game trees. A computer program to play Tic-tac-toe can be implemented perfectly, to enumerate the 765 essentially different positions (the state space complexity), or the 26,830 possible games up to rotations and reflections (the game tree complexity).

Despite its apparent simplicity, Tic-tac-toe requires detailed analysis to determine even some elementary combinatory facts, the most interesting of which are the number of

possible games and the number of possible positions. A position is merely a state of the board, while a game usually refers to the way a terminal position is obtained.

There are 19,683 possible board layouts (3^9 since each of the nine spaces can be X, O or blank), and 362,880 (i.e. 9!) possible games (different sequences for placing the Xs and Os on the board). However, two matters much reduce these numbers:

- The game ends when three-in-a-row is obtained.
- The number of Xs is always either equal to or exactly 1 more than the number of Os (if X starts).

The complete analysis is further complicated by the definitions used when setting the conditions, like board symmetries.

Number of terminal positions: When considering only the state of the board, and after taking into account board symmetries (i.e. rotations and reflections), there are only 138 terminal board positions. Assuming that X makes the first move every time:

- 91 unique positions are won by (X)
- 44 unique positions are won by (O)
- 3 unique positions are drawn

1.3 Scope of the Project

The field of software engineering and development is fast growing, especially the sector of game development which is growing even more rapidly.

In a game project such as this, the product is the game. However, a game is much more than software. It has to provide content to become enjoyable; and the quality for that cannot be measured. The software part of the project is not the only one; it must be considered in connection to all other parts: The environment of the game and so on.

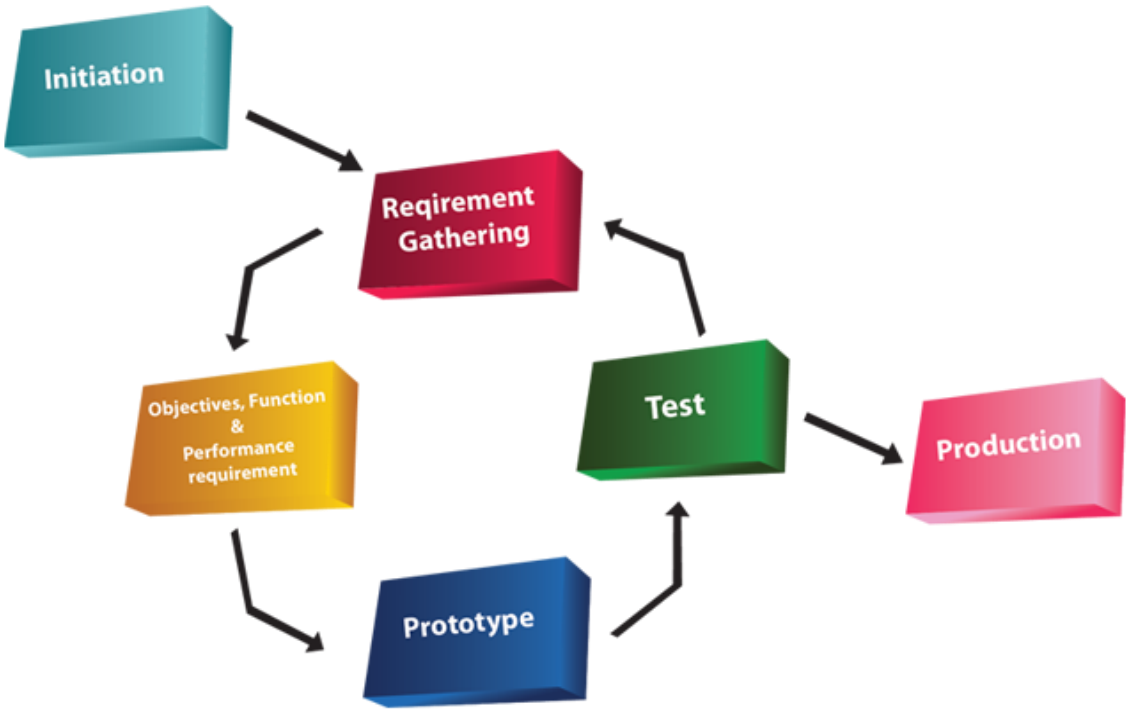
Since a large majority of the world indulges in playing games, the scope of games in general is huge. Tic Tac Toe, being a simple paper game, is even more widespread and well known, increasing the scope of this project dramatically.

2. Overall Description

2.1 Process Model

The “Prototyping” software process model has been adopted for this project. The model is a systems development method (SDM) in which a prototype (an early approximation of a final system or product) is built, tested, and then reworked as necessary until an acceptable prototype is finally achieved.

The basic idea is that throwaway prototype is built to understand the requirements. This prototype is developed based on the currently known requirements. Development of the prototype undergoes design, coding and testing. These phases are not performed very formally or thoroughly. By using this prototype, the client can get an "actual feel" of the system, since the interactions with prototype can enable the client to better understand the requirements of the desired system.



The Prototyping Process Model

This model works best in scenarios where not all of the project requirements are known in detail ahead of time. It is an iterative, trial-and-error process that takes place between the developers and the users.

There are several steps in the Prototyping Model:

1. The new system requirements are defined in as much detail as possible
2. A preliminary design is created for the new system.
3. A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
4. The users thoroughly evaluate the first prototype, noting its strengths and weaknesses, what needs to be added, and what should to be removed. The developer collects and analyzes the remarks from the users.
5. The first prototype is modified, based on the comments supplied by the users, and a second prototype of the new system is constructed.
6. The second prototype is evaluated in the same manner as was the first prototype.
7. The preceding steps are iterated as many times as necessary, until the users are satisfied that the prototype represents the final product desired.
8. The final system is constructed, based on the final prototype.
9. The final system is thoroughly evaluated and tested. Routine maintenance is carried out on a continuing basis to prevent large-scale failures and to minimize downtime.

Advantages of Prototyping:

1. Users are actively involved in the development
2. It provides a better system to users, as users have natural tendency to change their mind in specifying requirements and this method of developing systems supports this user tendency.
3. Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed.
4. Errors can be detected much earlier as the system is made side by side.
5. Quicker user feedback is available leading to better solutions.

Disadvantages:

1. Leads to implementing and then repairing way of building systems.
2. Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.

2.2 Functional Requirements

Normal requirements consist of the objectives and goals that are stated during the meeting with the relevant customers. Normal requirements of our project are:

1. User friendly efficient and lucrative system
2. Minimum maintenance cost
3. Availability of expected requirements within the required configuration
4. Ease of operation
5. Measured coding

Expected requirements are implicit to the system and may be so fundamental that the customer does not state them. Their absence is a cause for dissatisfaction:

1. Development system with limited cost
2. Minimum hardware requirements relevant to the game
3. Efficient design of the entire system

The game also has the following, important requirements:

1. User Interfaces – The presence of a menu and other elements to make a game user friendly and easy to understand are important.
2. Hardware Interfaces – A software should ideally be platform independent and should not require any specific software to run.

2.3 Non-functional Requirements

Nonfunctional requirements deal with the characteristics of the system which cannot be expressed as functions of the system, portability of the system, usability of the system, etc.

Nonfunctional requirements may include:

- Reliability issues
- Accuracy of results
- Human –Computer interface issues
- Constraints on the system implementation, etc.

2.4 Assumptions and Dependencies

The project makes use of game trees. The game tree consists of multiple nodes, each of which contains a certain state of the board.

In this project, identical board positions have not been eliminated. This has increased the number of nodes in the tree to over 300,000. Alpha-Beta pruning have also not been used to simplify the tree structure.

3 Project Design

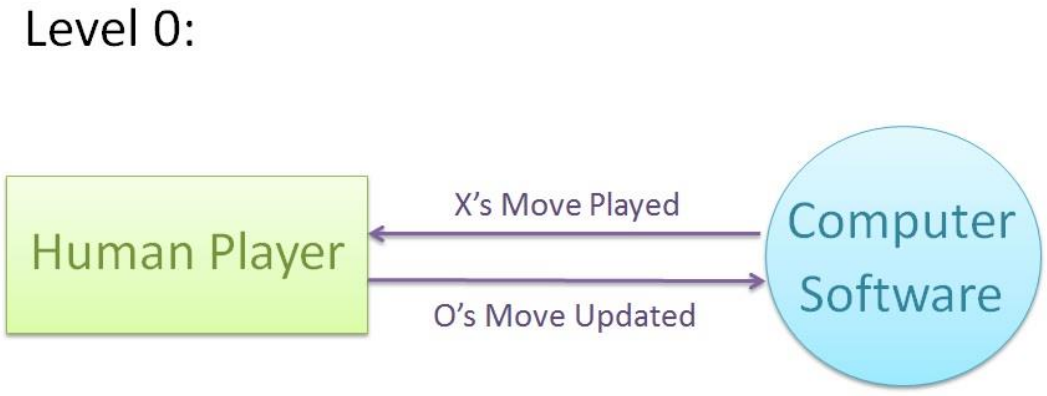
3.1 Analysis Models

3.1.1 Functional Requirements (DFD)

A data flow diagram is a graphical representation that depicts information flow and the transforms that are applied as data moves from input to output. Process specification (PSPEC) is used to specify the procedure details.

A DFD will look at from where data comes, where it goes and where it is stored. However it does not store information about processing timing and sequence.

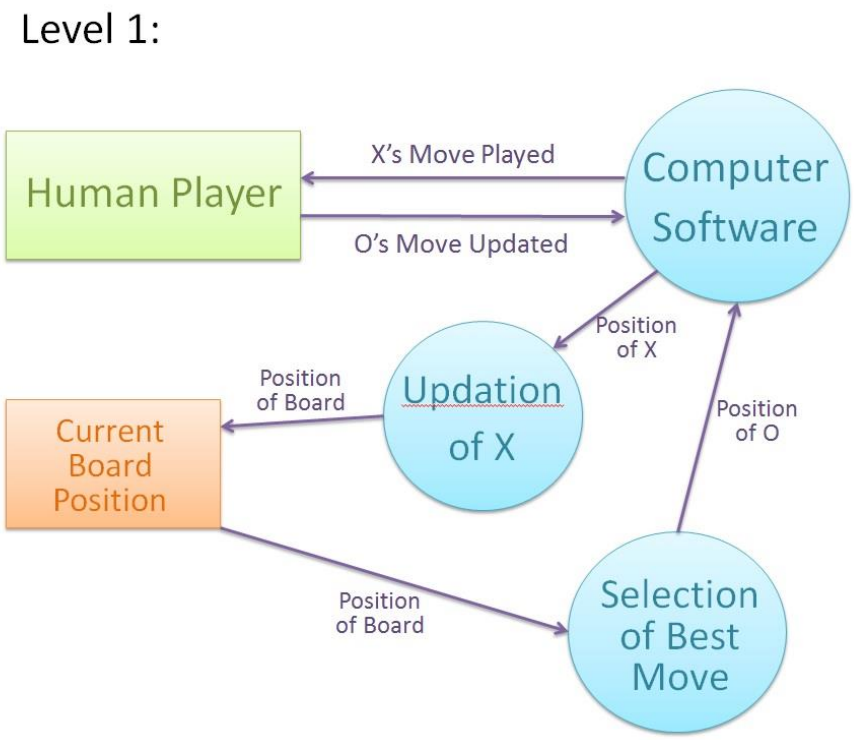
DFDs should not be confused with flowcharts. Flowcharts show the flow of control while DFDs show the flow of data. DFDs do not store control elements.



Data Flow Diagram – Level 0

The data flow diagram for this project consists of two major components: The human player and the Computer Software. The player inputs a position where X is to be played. In return, the computer software plays the O move and returns the updated state of the game to the player.

The functions have been described in more detail in the level one diagram.



Data Flow Diagram – Level 1

The computer software first accepts the position where X is to be updated. Then, the 'Updation of X' process checks whether the move is valid and updates accordingly. The updated state of the game is stored the 'Current Board Position' database.

The 'Selection of Best Move' process retrieves the current state from the game and processes the best move that can be made by O, i.e. the computer software. The process then returns the position for O and sends the updated state of the game to the user.

3.1.2 Entity Relationship Diagram

The data model of a project consists of three interrelated pieces of information:

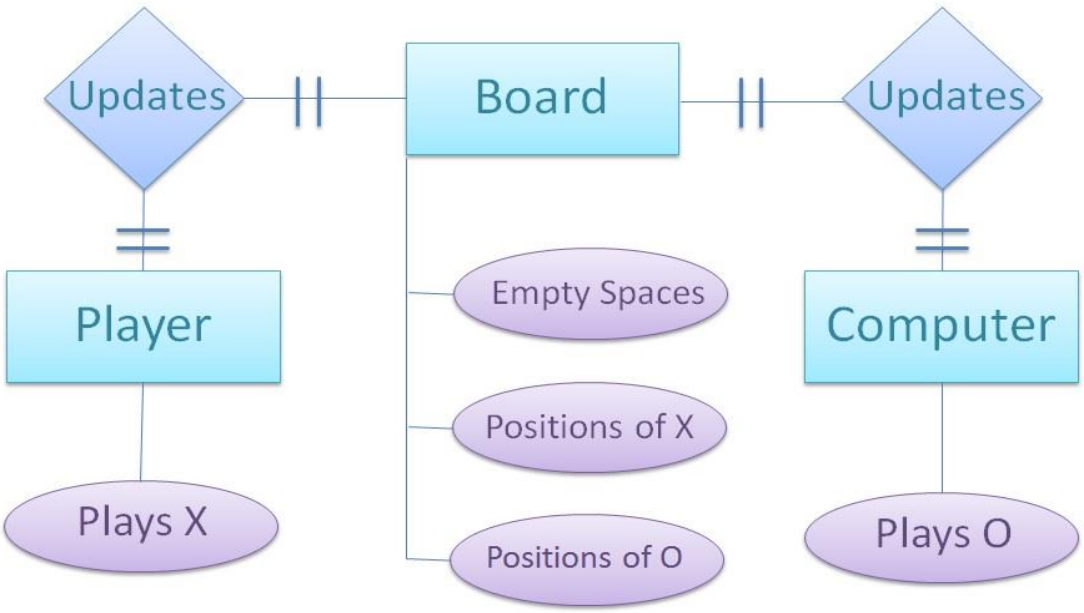
- The data object (Entity)
- The attributes that describe the data object
- The relationships that connect data objects to one another

The data object description (DOD) incorporates the data objects and all of their attributes. The Entiry Relationship Diagram (ERD) enables a software engineer to identify data objects and their relationships using a graphical notation.

A data object can be an external entity (E.g. anything that produces or consumes information), a thing (e.g., a report or a display), an occurrence (e.g., a telephone call) or event (e.g., an alarm), a role (e.g., salesperson), an organizational unit (e.g., accounting department), a place (e.g., a warehouse), or a structure (e.g., a file). Data objects are related to one another.

Attributes define the properties of a data object. They can be used to name an instance of the data object, describe the instance or make reference to another instance in another table. For example, a car is defined in terms of make, model, ID number, body type, color and owner. The body of the table represents specific instances of the data object.

The relationships are always defined by the context of the problem that is being analyzed. A relationship denotes a specific connection between the entities. Relationship pairs are bidirectional.

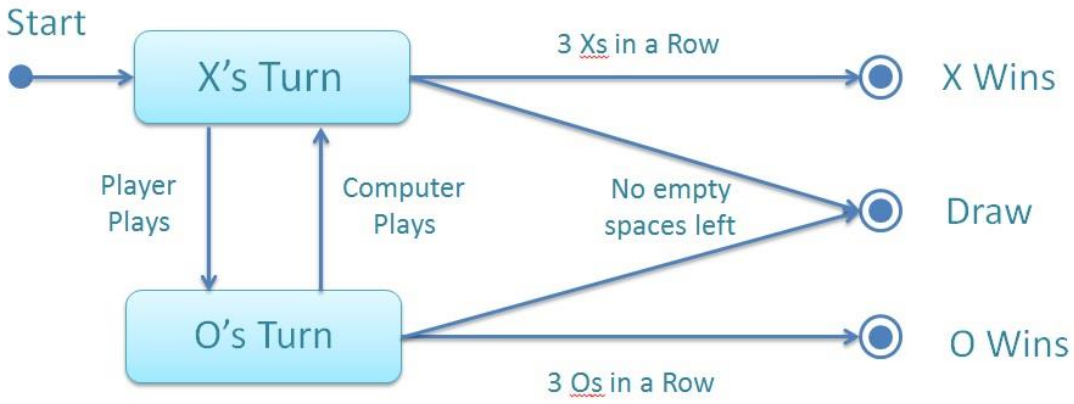


Entity – Relationship Diagram

3.1.3 State Transition Diagram

Behavioral modeling is an operational principle for all requirement analysis methods. Control information is contained in control specification or CSPEC.

The state transition diagram (STD) represents the behavior of a system by depicting its states and the events that cause the system to change state. The STD indicates what actions are taken as a consequence of a particular event. A state is any observable mode of behavior.



State Transition Diagram

The game begins with X's turn. After X (or the player) plays, the state is shifted such that it is O's turn. Now, O plays and the state is switched to X's turn again.

This process continues until the game ends. The game can end in three ways:

- If three Xs are placed in a straight line, X wins
- If three Os are places in a straight line, O wins
- If there are no empty spaces and no lines of either X or O have been established, the game is a draw.

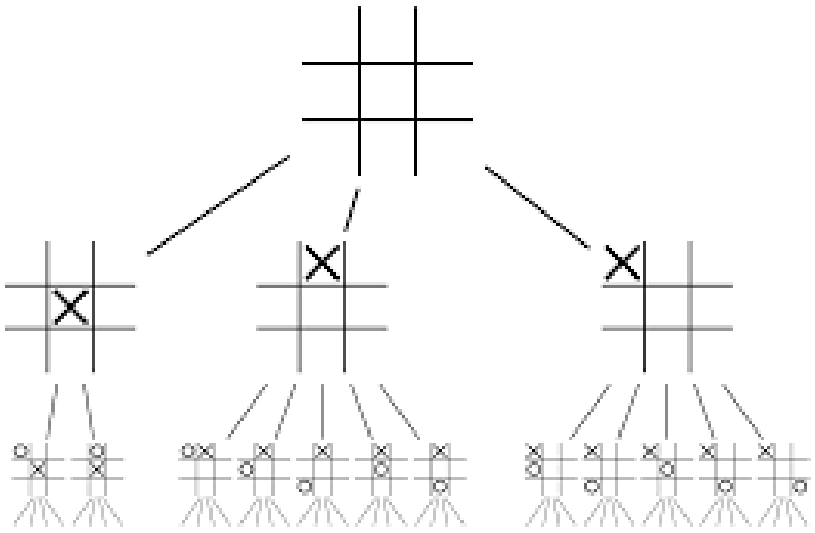
4. Application of Data Structures to solve the given problem

This project makes use of game trees. In game theory, a game tree is a directed graph whose nodes are positions in a game and whose edges are moves. The complete game tree for a game is the game tree starting at the initial position and containing all possible moves from each position.

Game trees are important in artificial intelligence because one way to pick the best move in a game is to search the game tree using the minimax algorithm or its variants. The game tree for tic-tac-toe is easily searchable, but the complete game trees for larger games like chess are much too large to search. Instead, a chess-playing program searches a partial game tree: typically as many plies from the current position as it can search in the time available. Except for the case of "pathological" game trees (which seem to be quite rare in practice), increasing the search depth (i.e., the number of plies searched) generally improves the chance of picking the best move.

Two-person games can also be represented as ‘and-or trees’. For the first player to win a game, there must exist a winning move for all moves of the second player. This is represented in the and-or tree by using disjunction to represent the first player's alternative moves and using conjunction to represent all of the second player's moves.

Each node in the game tree represents a state of the game. Every level, or “ply”, represents a change in state through a move by either player.



The diagram shows the first two levels, or plies, in the game tree for tic-tac-toe. The rotations and reflections of positions are equivalent, so the first player has three choices of move: in the

center, at the edge, or in the corner. The second player has two choices for the reply if the first player played in the center, otherwise five choices. And so on.

The number of leaf nodes in the complete game tree is the number of possible different ways the game can be played. The game tree for tic-tac-toe has 255,168 leaf nodes.

Thus, game trees are used to store and access all the possible moves and states of the game. Thus, the computer can "look ahead" and "predict" all the future moves of the game.

However, how does the computer select the best move out of all the possible moves and states it can 'see'? The answer lies in the Minimax Algorithm.

Minimax is a decision rule used in decision theory, game theory, statistics and philosophy for *mini*mizing the possible loss for a worst case (*max*imum loss) scenario. Originally formulated for two-player zero-sum game theory, covering both the cases where players take alternate moves and those where they make simultaneous moves, it has also been extended to more complex games and to general decision making in the presence of uncertainty.

With respect to game theory, the Minimax Theorem states that, for every two-person, zero-sum game with finitely many strategies, there exists a value V and a mixed strategy for each player, such that

(a) Given player 2's strategy, the best payoff possible for player 1 is V, and
(b) Given player 1's strategy, the best payoff possible for player 2 is −V.

Equivalently, Player 1's strategy guarantees him a payoff of V regardless of Player 2's strategy, and similarly Player 2 can guarantee himself a payoff of −V. The name minimax arises because each player minimizes the maximum payoff possible for the other—since the game is zero-sum, he also minimizes his own maximum loss (i.e. maximize his minimum payoff).

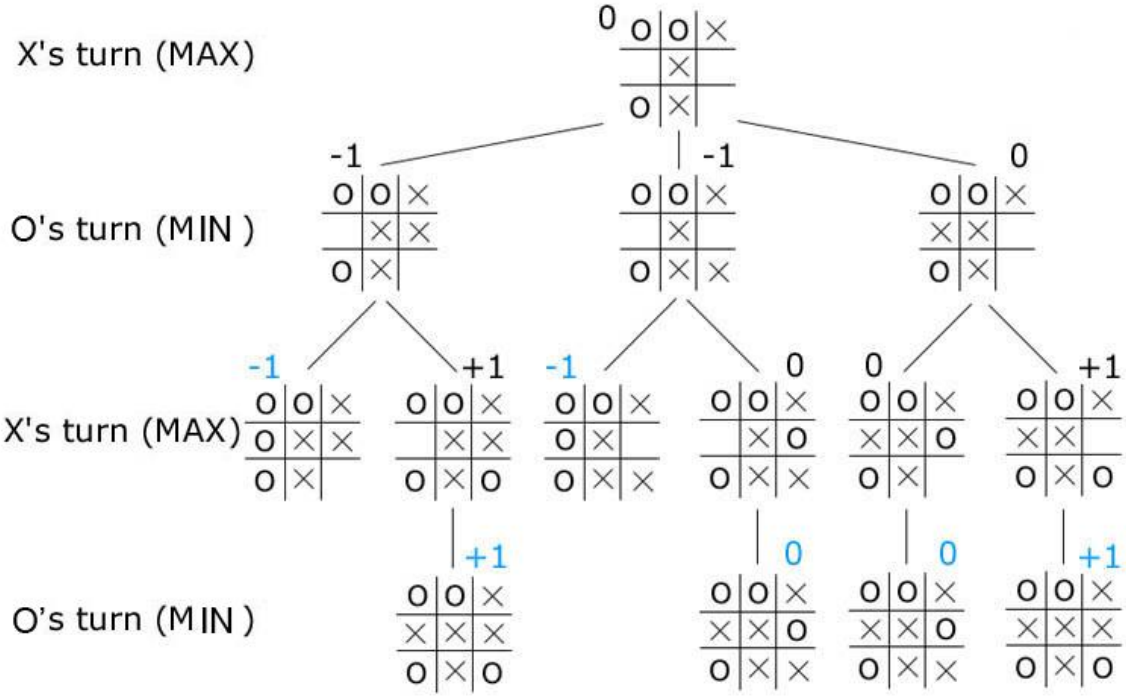
This theorem was first published in 1928 by John von Neumann, who is quoted as saying "As far as I can see, there could be no theory of games … without that theorem … I thought there was nothing worth publishing until the *Minimax Theorem* was proved".

In simpler words, leaf node of the game tree is assigned a 'Minimax' value. If X wins, the node is assigned the value 1. If O wins, the node is assigned the value -1. If the node results in a draw, it is assigned with the value 0. These values can be arbitrary.

Thus, it is apparent that 'lower' or 'min' values are good for player 'O' while 'higher' or 'max' values are good for player 'X'. Thus, the algorithm has been named MiniMax.

Depending on whose turn it is at the level of the parent, the parent is assigned either the highest (Max, i.e. X's turn) or lowest (Min, i.e. O's turn) value of its children.

This process is continued till a Minimax value has been assigned to all nodes in the tree, upto the root of the tree (in case of full trees) or the current state of the board (in case of partial trees).



An example of a game tree for the game Tic Tac Toe, using the Minimax Algorithm

5. Output of the Program

```
                        Welcome to Tic Tac Toe!

Rules: Attempt to place 3Xs or Os in a stright line vetrtically, horizontally or
diagonally to win!

Good luck!

Enter a number from 1 to 9 to select your position.


                                  |   |
                                ----------
                                  |   |
                                ----------
                                  |   |

X's turn. 5
```

```
                        Welcome to Tic Tac Toe!

Rules: Attempt to place 3Xs or Os in a stright line vetrtically, horizontally or
diagonally to win!

Good luck!

Enter a number from 1 to 9 to select your position.


                                  |   | O
                                ----------
                                  | X |
                                ----------
                                  |   |

O's turn.
```

```
                        Welcome to Tic Tac Toe!

Rules: Attempt to place 3Xs or Os in a stright line vetrtically, horizontally or
diagonally to win!

Good luck!

Enter a number from 1 to 9 to select your position.


                                  |   | O
                                ----------
                                  | X |
                                ----------
                                  |   |

X's turn. 4
```

```
                      Welcome to Tic Tac Toe!

Rules: Attempt to place 3Xs or Os in a stright line vetrtically, horizontally or
diagonally to win!

Good luck!

Enter a number from 1 to 9 to select your position.


                                |   | O
                               ----------
                               X | X | O
                               ----------
                                |   |

O's turn.
```

```
                      Welcome to Tic Tac Toe!

Rules: Attempt to place 3Xs or Os in a stright line vetrtically, horizontally or
diagonally to win!

Good luck!

Enter a number from 1 to 9 to select your position.


                                |   | O
                               ----------
                               X | X | O
                               ----------
                                |   |

X's turn. 9
```

```
                      Welcome to Tic Tac Toe!

Rules: Attempt to place 3Xs or Os in a stright line vetrtically, horizontally or
diagonally to win!

Good luck!

Enter a number from 1 to 9 to select your position.


                              | O | O
                               ----------
                               X | X | O
                               ----------
                                |   | X

O's turn.
```

```
                      Welcome to Tic Tac Toe!

Rules: Attempt to place 3Xs or Os in a stright line vetrtically, horizontally or
diagonally to win!

Good luck!

Enter a number from 1 to 9 to select your position.


                               |   | O | O
                              ----------
                              X | X | O
                              ----------
                                |   | X

X's turn. 1
```

```
                      Welcome to Tic Tac Toe!

Rules: Attempt to place 3Xs or Os in a stright line vetrtically, horizontally or
diagonally to win!

Good luck!

Enter a number from 1 to 9 to select your position.


                              X | O | O
                              ----------
                              X | X | O
                              ----------
                              O |   | X

O's turn.
```

```
                      Welcome to Tic Tac Toe!

Rules: Attempt to place 3Xs or Os in a stright line vetrtically, horizontally or
diagonally to win!

Good luck!

Enter a number from 1 to 9 to select your position.


                              X | O | O
                              ----------
                              X | X | O
                              ----------
                              O |   | X

X's turn. 8
```


Welcome to Tic Tac Toe!
Rules: Attempt to place 3Xs or Os in a stright line vetrtically, horizontally or diagonally to win!
Good luck!
Enter a number from 1 to 9 to select your position.
X | O | O
X | X | O
O | X | X
Draw!