**REPORT FOR TIC-TAC-TOE USING MINMAX ALGORITHIM**

As a project work for Course

**ARTIFICIAL INTELLIGENCE (INT 404)**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

NAME : SHESHANK

REGISTRATION NO. : 11802650

NAME : PURUSHOTTAM SINGH

REGISTRATION NO. : 11802669

NAME : ASHISH SUKRALIA

REGISTRATION NO. : 11802744

PROGRAM : CSE (B.Tech)

SEMESTER : Four

SCHOOL : School of Computer Science

and Engineering

NAME OF UNIVERSITY : Lovely Professional University,

Phagwara, Punjab, India

DATE OF SUBMISSION : 25MAR-31MAR

**Tic Tac Toe**

***Abstract: -***

**Tic-tac-toe** (American), **noughts and crosses** (British), or **Xs and Os** 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 of their marks in a horizontal, vertical, or diagonal row is the winner.

The following example game is won by the first player, X:

[Game of Tic-tac-toe, won by X](https://en.wikipedia.org/wiki/File:Tic-tac-toe-game-1.svg)

***Acknowledgement: -***

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend my sincere thanks to all of them.

We are highly indebted to Sagar Pandey Sir for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

We would like to express our gratitude towards our parents & member of Lovely Professional University for their kind co-operation and encouragement which help us in completion of this project.

**Table of Content**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Topic** | **Page** |
| 1. | Abstract | 2 |
| 2. | Introduction | 5 |
| 3. | Team members with their role | 6 |
| 4. | Functions of Program | 7 |
| 5. | Process of Unbeatable Tic Tac Toe | 9 |
| 6 | About minmax algorithim | 10 |
| 7. | Flowchart of minmax algorithim | 11 |
| 8. | Application | 12 |
| 9. | Conclusion | 13 |
| 10. | References | 13 |

***Introduction: -***

Because of the simplicity of tic-tac-toe, it is often used as a pedagogical tool for teaching the concepts of good sportsmanship and the branch of artificial intelligence that deals with the searching of game trees. It is straightforward to write a computer program to play tic-tac-toe perfectly or 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) on this space.

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. Harary's generalized tic-tac-toe is an even broader generalization of tic-tac-toe. It can also be generalized as a nd game. Tic-tac-toe is the game where n equals 3 and d equals 2.If played properly, the game will end in a draw, making tic-tac-toe a futile game.

***Team Members with their role: -***

SHESHANK:-

***Contributions***:-

1. Coding(joined)
2. MinMax(joined)
3. Reports

PURUSHOTTAM:-

***Contributions***:-

1. Coding(joined)
2. Setup of game
3. MinMax(joined)

ASHISH:-

***Contributions:-***

1. Coding(joined)
2. Setup of game
3. Report(joined)

**Functions*: -***

* ***game\_board:-*** This function returns the setup of board for game with all blank spaces.
* ***Show\_board:-*** This functions uses board as a parameter and returns nothing. But it shows or displays the board with its current position.
* ***Empty\_places:-***  This function uses board as a parameter. Return either True or False. If no place is left to play in the board then return False else True.
* ***Winner\_check:-***  Takes board and win\_game as a parameter and returns either Human or AI wins the game.
* ***Play\_game :-*** No parameter and returns nothing. It runs the whole game inside itself.
* ***Player\_Turn:-*** Takes board as a parameter and returns board and player\_turn variable .
* ***Find\_best\_move:-*** Takes three parameters :- 1.Board, 2. Player\_turn ,3. ***Win\_game*** and return all these three variables also.
* ***Minmax:- takes four variables ad parameters and returns two variable . It is used to find the best step for AI to play.***

***Now about some important variable in the game:***

1. ***Player\_turn :- It is a Boolean variable which is True if it is player or human turn or in case of AI turn it is false.***
2. ***Board:- It is a “3x3” List which is the board for our game.***
3. ***Win\_game:- It is a character variable to check and decide who win the game.***
4. ***Human :- It is a char variable to play on the board for player it may be ‘X’ or ‘O’ depends on Player prefence.***
5. ***AI :- It is a char variable to play on the board for player it may be ‘X’ or ‘O’ depends on Player prefence.***

***Process of Unbeatable Tic Tac Toe: -***

In Tic Tac Toe there are some combinations depends on present situation to lead to a draw. In this process You can’t beat the the computer(AI).

For making the Unbeatable Tic Tac Toe we uses a game theory algorithim

Name –“MINMAX” .\

Consider a board with the nine positions numbered as follows:

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

When X plays 1 as their opening move, then O should take 5. Then X takes 9 (in this situation, O should not take 3 or 7, O should take 2, 4, 6 or 8):

* X1 → O5 → X9 → O2 → X8 → O7 → X3 → O6 → X4, this game will be a draw.

or 6 (in this situation, O should not take 4 or 7, O should take 2, 3, 8 or 9. In fact, taking 9 is the best move, since a non-perfect player X may take 4, then O can take 7 to win).

* X1 → O5 → X6 → O2 → X8, then O should not take 3, or X can take 7 to win, and O should not take 4, or X can take 9 to win, O should take 7 or 9.
  + X1 → O5 → X6 → O2 → X8 → O7 → X3 → O9 → X4, this game will be a draw.
  + X1 → O5 → X6 → O2 → X8 → O9 → X4 (7) → O7 (4) → X3, this game will be a draw.
* X1 → O5 → X6 → O3 → X7 → O4 → X8 (9) → O9 (8) → X2, this game will be a draw.
* X1 → O5 → X6 → O8 → X2 → O3 → X7 → O4 → X9, this game will be a draw.
* X1 → O5 → X6 → O9, then X should not take 4, or O can take 7 to win, X should take 2, 3, 7 or 8.
  + X1 → O5 → X6 → O9 → X2 → O3 → X7 → O4 → X8, this game will be a draw.
  + X1 → O5 → X6 → O9 → X3 → O2 → X8 → O4 (7) → X7 (4), this game will be a draw.
  + X1 → O5 → X6 → O9 → X7 → O4 → X2 (3) → O3 (2) → X8, this game will be a draw.
  + X1 → O5 → X6 → O9 → X8 → O2 (3, 4, 7) → X4/7 (4/7, 2/3, 2/3) → O7/4 (7/4, 3/2, 3/2) → X3 (2, 7, 4), this game will be a draw.

In both of these situations (X takes 9 or 6 as second move), X has a 1/3 property to win.

If X is not a perfect player, X may take 2 or 3 as second move. Then this game will be a draw, X cannot win.

* X1 → O5 → X2 → O3 → X7 → O4 → X6 → O8 (9) → X9 (8), this game will be a draw.
* X1 → O5 → X3 → O2 → X8 → O4 (6) → X6 (4) → O9 (7) → X7 (9), this game will be a draw.

If X plays 1 opening move, and O is not a perfect player, the following may happen:

Although O takes the only good position (5) as first move, but O takes a bad position as second move:

* X1 → O5 → X9 → O3 → X7, then X can take 4 or 8 to win.
* X1 → O5 → X6 → O4 → X3, then X can take 2 or 9 to win.
* X1 → O5 → X6 → O7 → X3, then X can take 2 or 9 to win.

Although O takes good positions as the first two moves, but O takes a bad position as third move:

* X1 → O5 → X6 → O2 → X8 → O3 → X7, then X can take 4 or 9 to win.
* X1 → O5 → X6 → O2 → X8 → O4 → X9, then X can take 3 or 7 to win.

O takes a bad position as first move (except of 5, all other positions are bad):

* X1 → O3 → X7 → O4 → X9, then X can take 5 or 8 to win.
* X1 → O9 → X3 → O2 → X7, then X can take 4 or 5 to win.
* X1 → O2 → X5 → O9 → X7, then X can take 3 or 4 to win.
* X1 → O6 → X5 → O9 → X3, then X can take 2 or 7 to win.

**Process of Unbeatable Tic TAc Toe**

***GamePlay:-***

Ultimate tic-tac-toe is significantly more complex than most other variations of tic-tac-toe, as there is no clear strategy to playing. This is because of the complicated [game branching](https://en.wikipedia.org/wiki/Branch_(computer_science)" \o "Branch (computer science)) in this game. Even though every move must be played in a local board, equivalent to a normal tic-tac-toe board, each move must take into account the global board in several ways:

1. **Anticipating the next move:** Each move played in a local board determines where the opponent's next move may be played. This might make moves that may be considered bad in normal tic-tac-toe viable, since the opponent is sent to another local board, and may be unable to immediately respond to them. Therefore, players are forced to consider the larger game board instead of simply focusing on the local board.
2. **Visualizing the game tree:** Visualizing future branches of the [game tree](https://en.wikipedia.org/wiki/Game_tree" \o "Game tree) is more difficult than single board tic-tac-toe. Each move determines the next move, and therefore reading ahead—predicting future moves—follows a much less linear path. Future board positions are no longer interchangeable, each move leading to starkly different possible future positions. This makes the game tree difficult to visualize, possibly leaving many possible paths overlooked.
3. **Winning the game:** Due to the rules of ultimate tic-tac-toe, the global board is never directly affected. It is only governed by actions that occur in local boards. This means that each local move played is not intended to win the local board, but to win the global board. Local wins are not valuable if they cannot be used to win the global board—in fact, it may be strategic to sacrifice a local board to your opponent in order to win a more important local board yourself. This added layer of complexity makes it harder for humans to analyze the relative importance and significance of moves, and consequently harder to play well.

***About MinMax:-***

In two-player [zero-sum games](https://en.wikipedia.org/wiki/Zero-sum_game" \o "Zero-sum game), the minimax solution is the same as the [Nash equilibrium](https://en.wikipedia.org/wiki/Nash_equilibrium" \o "Nash equilibrium).

In the context of zero-sum games, the [minimax theorem](https://en.wikipedia.org/wiki/Minimax_theorem" \o "Minimax theorem) is equivalent to:

For every two-person, [zero-sum](https://en.wikipedia.org/wiki/Zero-sum" \o "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 them a payoff of V regardless of Player 2's strategy, and similarly Player 2 can guarantee themselves 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, they also minimize their own maximum loss (i.e. maximize their minimum payoff). See also [example of a game without a value](https://en.wikipedia.org/wiki/Example_of_a_game_without_a_value" \o "Example of a game without a value).

**Pseudo Code of MINMAX :-**

**function** minimax(node, depth, maximizingPlayer) **is**

**if** depth = 0 **or** node is a terminal node **then**

**return** the heuristic value of node

**if** maximizingPlayer **then**

value := −∞

**for each** child of node **do**

value := max(value, minimax(child, depth − 1, FALSE))

**return** value

**else** *(\* minimizing player \*)*

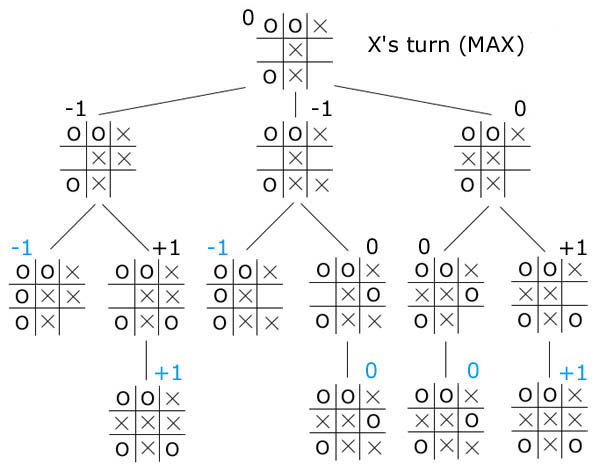
value := +∞

**for each** child of node **do**

value := min(value, minimax(child, depth − 1, TRUE))

**return** value

**FlowChart of MinMax:-**

****

***Applications: -***

We don’t know about more of the applications of tic tac toe.But there are some which are in the following image:-



***Conclusion: -***

In the end we would like to conclude that my aim

To make this project is to research in the field of

Artificial Intelligence and implementing by developing

The logic for the game.

Some scope of improvements are also there in the

Project which will rectified in the future advancements

of the project.We would like to thank all those who have

helped us and contributed in making of this project.

***References: -***

1. <https://en.wikipedia.org/wiki/Tic-tac-toe>
2. <https://en.wikipedia.org/wiki/Minimax>
3. <https://www.youtube.com/watch?v=GTWrWM1UsnA&t=279s> (This video is in java script)
4. <https://www.youtube.com/watch?v=trKjYdBASyQ&t=1157s> (This video is in java script )
5. <https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/>