-AI Project Discussion Report:

-Software Engineering, Level 4:

-TeamID: 6

20211818 **-ايريني إيهاب حلمي**

20211833 **-رضوي مصطفي محمد**

20211824 **-تسنيم عادل محمد**

20211821 **-بسمة عماد محمد**

……

- An Intelligent Tic-Tac-Toe Player using the Minimax Algorithm, Alpha-Beta Pruning, and Heuristic Functions:

- Introduction and Overview:

- developing an intelligent Tic-Tac-Toe player:

- The project aims to develop an intelligent Tic-Tac-Toe player that employs advanced algorithms and heuristic functions to make strategic decisions during gameplay.
- Tic-Tac-Toe is a classic two player game where opponents take turns marking X or O in a 3x3 grid, aiming to form a line of three of their symbols horizontally, vertically, or diagonally.

- the functionalities/features of the Game:

- Basic Interface: The game has a straightforward user interface, consisting of a 3x3 grid representing the Tic Tac Toe board.

- Two-Player Mode: The primary mode of the game is for (Player vs AI) to take turns playing against each other. One player uses "X" as their symbol, and the other uses "O."

- Turn-Based Gameplay: Players take turns placing their symbols on empty spaces in the grid. The first player to align three of their symbols horizontally, vertically, or diagonally wins the game.

- Winning Announcement: When a player wins, the game announces the winner and highlights the winning combination on the board.

- Restart or Play Again Option: After a game is completed, players typically have the option to restart the game or play again without reopening the application.

- the algorithms used for the Game :

1- Minimax Algorithm:

-The time complexity:

-the branching factor as b and the depth of the tree as d. The time complexity of the Minimax algorithm can be expressed as O(b^d).

-The Minimax algorithm is :

a decision-making algorithm used in two-player game.

Its primary objective is to find the optimal move for a player, assuming that the opponent is also playing optimally.

The term "Minimax" is derived from the algorithm's goal of minimizing the possible loss for a worst-case scenario (loss for the player)

and maximizing the potential gain.

2-Minimax with Alpha-Beta pruning Algorithm:

-The time complexity: of the minimax algorithm with alpha-beta pruning is often expressed as: O(b^d/2).

-Alpha-beta pruning: significantly reduces the number of nodes explored, especially in scenarios where large parts of the tree can be pruned.

It avoids unnecessary computations by disregarding branches that cannot affect the final decision.

3-Minimax with Distance_heuristic:

-The time complexity: the distance heuristic in the evaluation function, it will add a constant time factor to the evaluation of each node. Let's denote h

h as the time complexity of calculating the distance heuristic for a node. In this case, the modified time complexity becomes:

O(b^d.h) -> h = O(n).

- **Minimax with Distance_heuristic:** the algorithm aims to improve its efficiency in searching through the game tree. The distance heuristic provides additional information about the desirability of a move by considering the spatial relationship of the move on the game board. This help the algorithm prioritize moves that are closer to a certain position, possibly leading to a more informed decision.

4- Minimax with blocking_opponent_evaluation:

-Time complexity is: O(max(m,b^d)).

-Minimax with blocking_opponent_evaluation: When describing the Minimax algorithm with blocking_opponent_evaluation, it typically means incorporating an evaluation function that takes into account the opponent's potential to win or block winning moves. This enhancement aims to make the

algorithm more strategic by considering not only its own chances of winning but also preventing the opponent from winning.

5- Symmetry Reduction Algorithm:

-Time complexity: O(N^2).

- Symmetry Reduction Algorithm: The goal of this algorithm is to exploit the symmetrical nature of certain game states to reduce the effective size of the search space, leading to more efficient and faster computations.

6- Heuristic Reduction Algorithm:

-Time complexity: O(N log N) + O(N) = O(N log N).

- Heuristic Reduction Algorithm: is a method of finding a good move for a player by evaluating the board state and assigning scores to different positions. The algorithm tries to reduce the complexity of the problem by using a heuristic function that estimates the value of a board for a player, without considering all the possible outcomes. The heuristic function can be based on various criteria, such as the number of marks in a row, the number of open spaces, the potential threats and opportunities, etc.

The algorithm then chooses the move that maximizes the heuristic score for the player, or minimizes it for the opponent.

- Main functionalities/features in our proposed Game (explained using a use-case diagram and a flowchart):

- Start Game:
  -Description: Initiates a new game.
  -Actor: Player(s)

-Trigger: User selects "Start Game" option.
-Outcome: The game board is initialized, and the first player is ready to make a move.

- Make Move:
  -Description: Represents a player making a move (placing "X" or "O" on the board).
  -Actor: Player
  -Trigger: User selects an empty cell on the game board.
  -Outcome: The selected cell is filled with the player's symbol, and the turn switches to the next player.

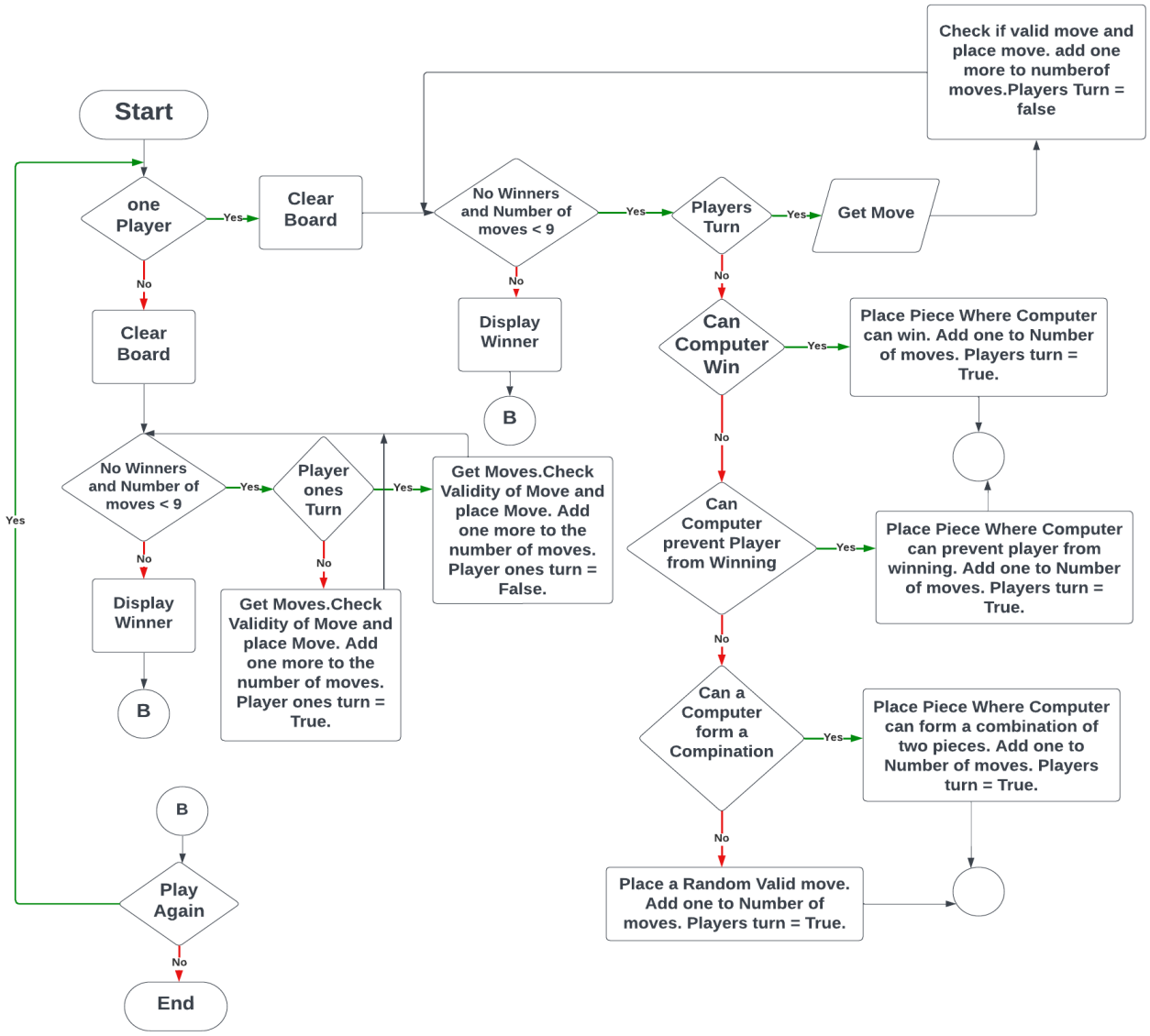
- Check Win:
  -Description: Checks if a player has won the game.
  -Actor: System
  -Trigger: After each move.
  -Outcome: If a player has won, the game announces the winner, highlighting the winning combination.

- Restart Game:
  -Description: Restarts the game.
  -Actor: Player(s)
  -Trigger: User selects "Restart" option.
  -Outcome: The game is reset to the initial state, ready for a new round.

- Quit Game:
  -Description: Exits the game.
  -Actor: Player(s)
  -Trigger: User selects "Quit" option.
  -Outcome: The game application is closed.

- **Use-Case Diagram:**

- **Flowchart Diagram:**

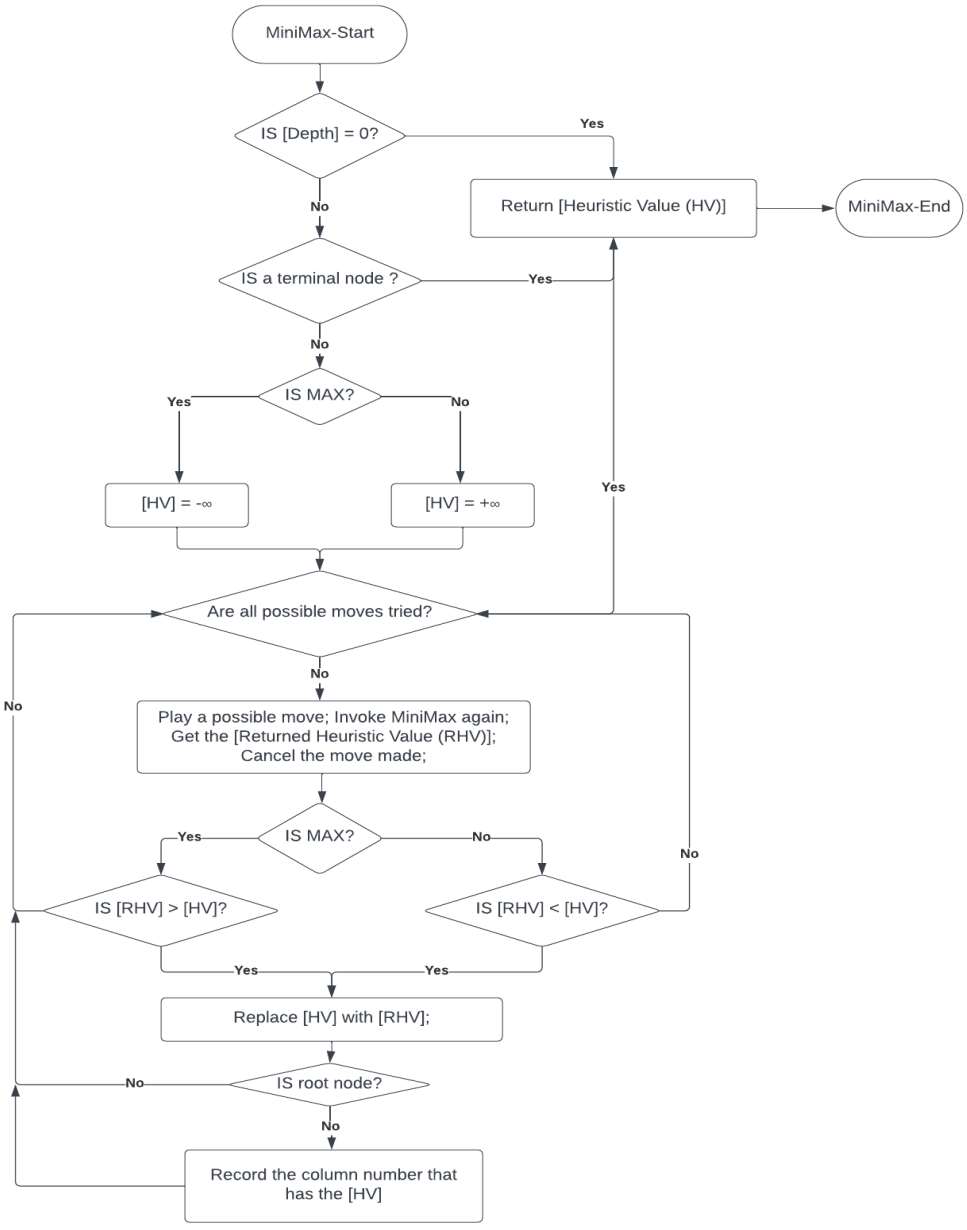


-Applied Algorithms:

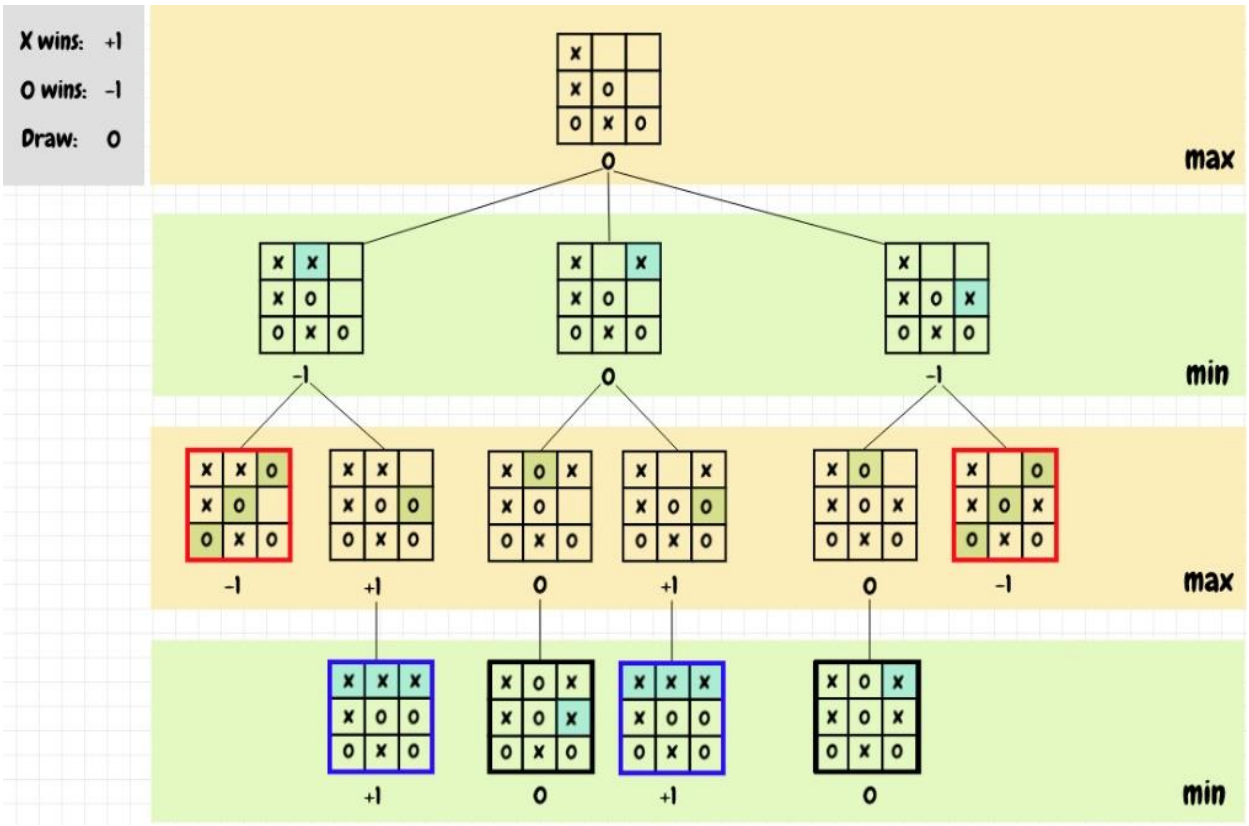
- All the details of some of the AI algorithms used to develop our Game (explained using block diagrams and the flowchart diagram):

1- Minimax Algorithm:

- Mini-Max Algorithm Flow Chart:

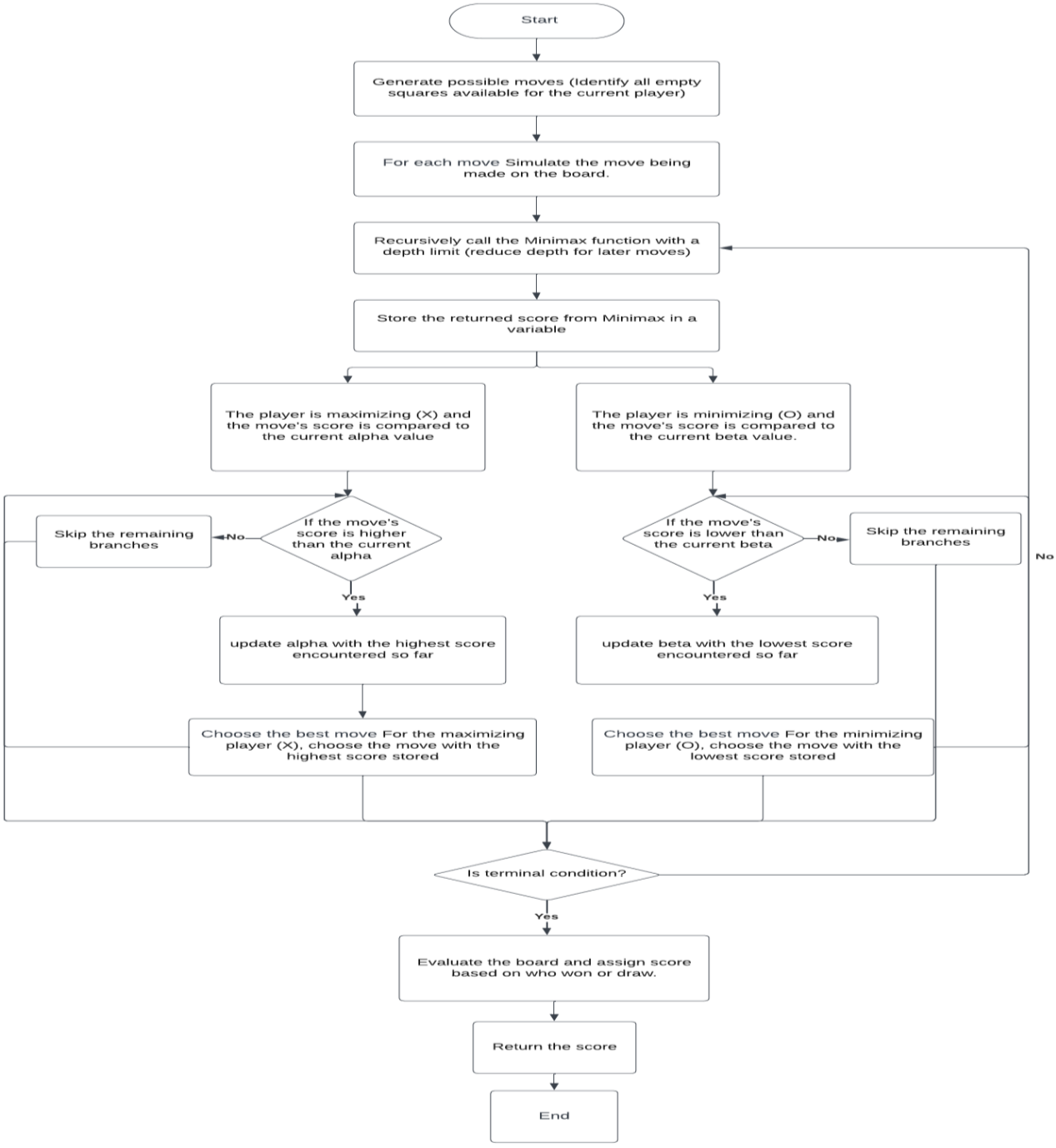


- Mini-Max Algorithm Game Tree:

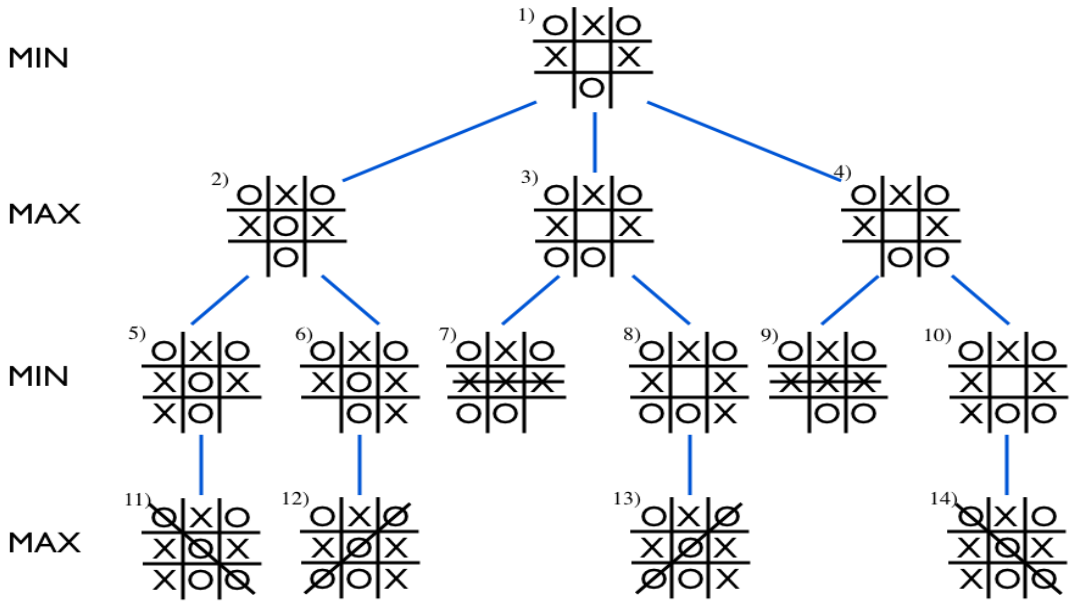


2- **Alpha-Beta Pruning:**

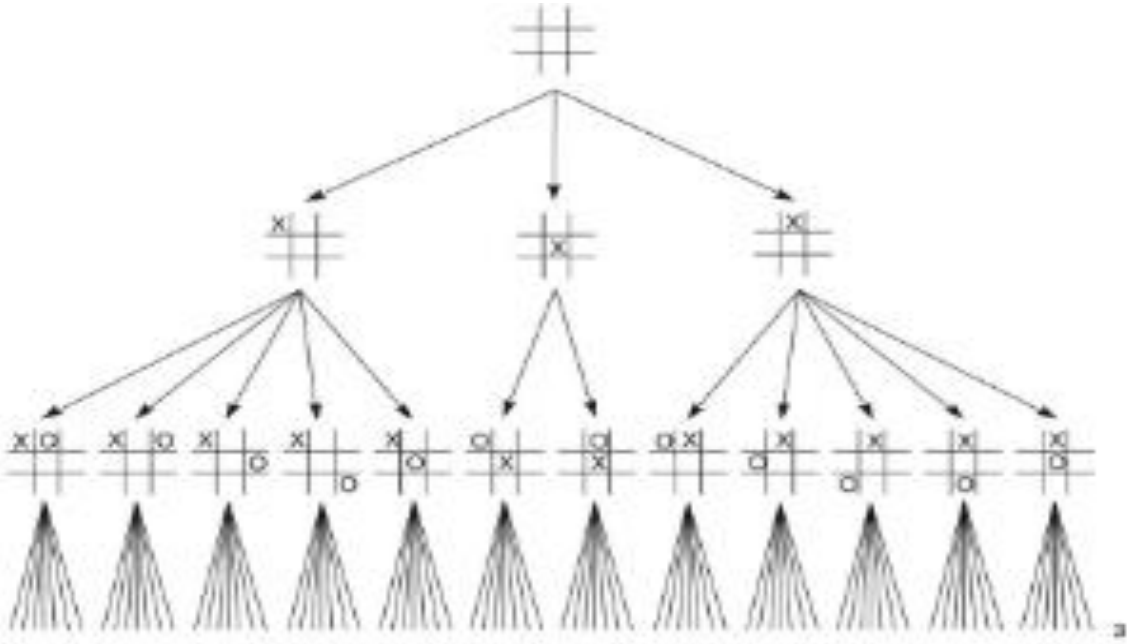
- Alpha-Beta Pruning Flow Chart:



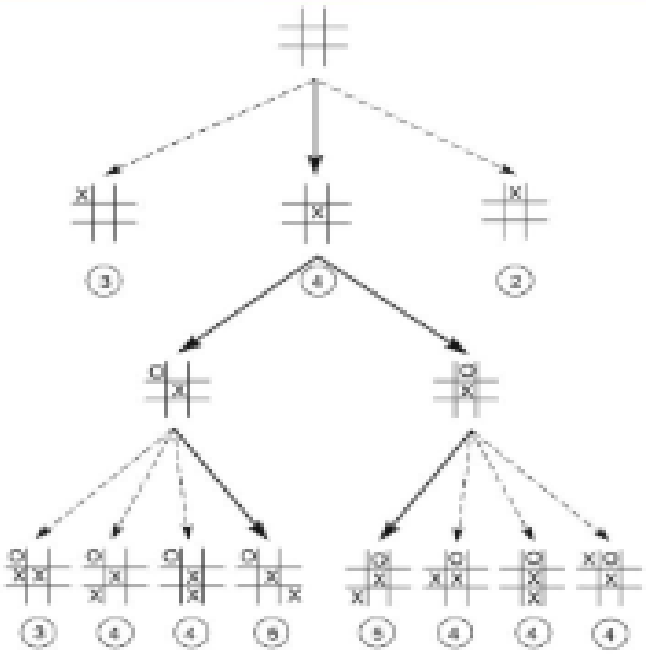
- Alpha-Beta Pruning Game Tree:



3- Symmetry Reduction Algorithm:



4- Heuristic Reduction:



-Statistical Analysis and Visualization:

- The algorithm collects statistics on the number of expanded_nodes explored for each algorithm.
- The mean average of number of nodes explored is visualizing using bar chart.
- And there are a lot of another ways to compare them by the time and the space complexity in the terminal.

Some Results:

```
Minimax: [55504, 56436, 56486, 56490, 59704, 60638, 60684, 60688, 5550
4, 56436, 56486, 56490]
Alpha-beta: [1705, 1829, 1851, 1854, 2706, 2808, 2830, 2832, 2549, 264
8, 2670, 2673, 3817, 4275, 4311, 4314]
```

```
Distance heuristic: [2303, 2575, 2607, 2611, 2303, 2575, 2607, 2611, 4
018, 4631, 4673, 4677]
Blocking opponent heuristic: [2315, 2544, 2576, 2580, 3956, 4531, 4576
, 4579, 4980, 5386, 5419, 5423]
Symmetry reduction: [554, 647, 669, 672, 534, 700, 711, 713, 595, 789,
806, 761, 915, 924]
Heuristic reduction: [2, 5, 8, 11, 2, 5, 8, 11, 2, 5, 8, 11, 2]
```