**AI Project Documentation**

**Tic-Tac-Toe**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **ID** | **Level** | **Dept.** |
| **علي الدين اشرف حسن قطب** | **20220296** | **3** | **AI** |
| **عمر احمد عطية امين** | **20220310** | **3** | **AI** |
| **عمر وجيه محسن احمد** | **20220324** | **3** | **AI** |
| **مريم عمرو صلاح الدين** | **20220468** | **3** | **CS** |
| **ملك علاء محمد** | **20220503** | **3** | **AI** |

**Introduction and overview**

**Project idea and overview**

The project aims to develop an intelligent Tic-Tac-Toe player that plays against humans in the classic 3X3 grid using advanced AI algorithms to make the optimal decisions

The main algorithms used are Minmax algorithm, alpha beta pruning and some heuristic functions

**Similar applications**

Several applications provide Tic-Tac-Toe games with AI opponents:

1.Google’s Tic-Tac-Toe Game: A browser-based game with adjustable difficulty levels.

2.Tic-Tac-Toe Android/iOS Apps: Mobile apps, often featuring user-friendly interfaces and adjustable AI strategies.

3.Command-Line Implementations: Many educational projects provide text-based Tic-Tac-Toe games to demonstrate AI concepts.

These implementations primarily rely on the Minimax algorithm or its variants for decision-making. The proposed project extends upon this with additional optimizations like symmetry reduction and heuristic evaluation.

**Literature review**

1."Artificial Intelligence: A Modern Approach" by Stuart Russell and Peter Norvig

Describes foundational concepts of Minimax, Alpha-Beta Pruning, and heuristic design for AI decision-making.

2."Game Theory and AI Applications" by Shoham and Leyton-Brown

Explores the theoretical basis of decision-making algorithms in games.

3."Heuristics: Intelligent Search Strategies for Computer Problem Solving" by Judea Pearl

Explains heuristic evaluation functions and their role in game-playing algorithms.

4."Algorithms for Tic-Tac-Toe" by Claude Shannon

Focuses on classical approaches to solving Tic-Tac-Toe.

5.Research on Symmetry Reduction in Game Trees (Various articles from Google Scholar)

Discusses how exploiting symmetry can optimize the search space in grid-based games.

. <https://www.geeksforgeeks.org/finding-optimal-move-in-tic-tac-toe-using-minimax-algorithm-in-game-theory/>

. <https://medium.com/@amadi8/tic-tac-toe-agent-using-alpha-beta-pruning-18e8691b61d4>

. <https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/>

<https://www.whitman.edu/documents/academics/mathematics/2019/felstiner-guichard.pdf>

<https://www.researchgate.net/publication/343021763_An_Intuitive_Implementation_Of_Alpha-Beta_Pruning_Using_Tic-Tac-Toe>

**Proposed solution**

**Main functionalities :**

**.** User chooses desired algorithm

**.** AI plays optimal moves using chosen algorithm

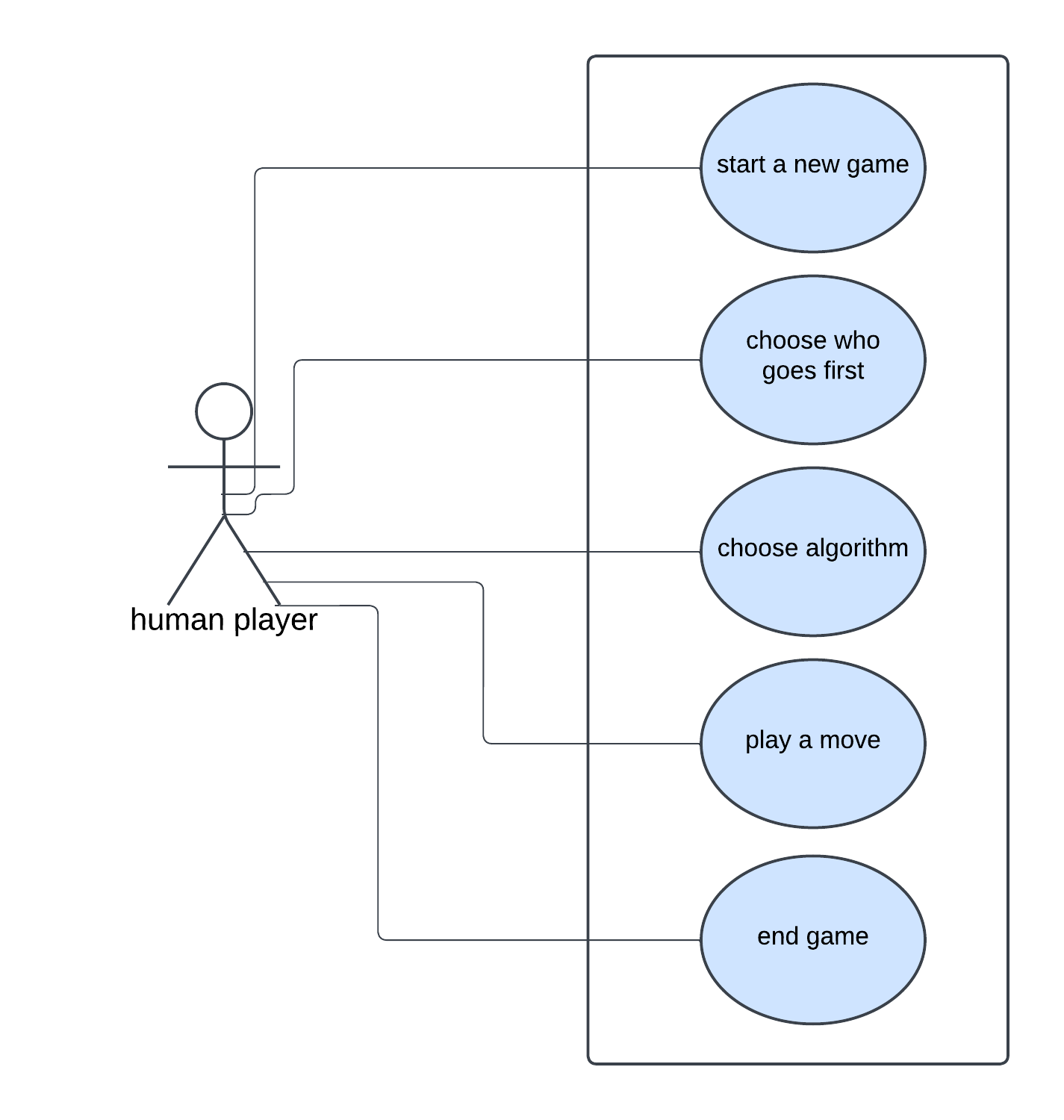
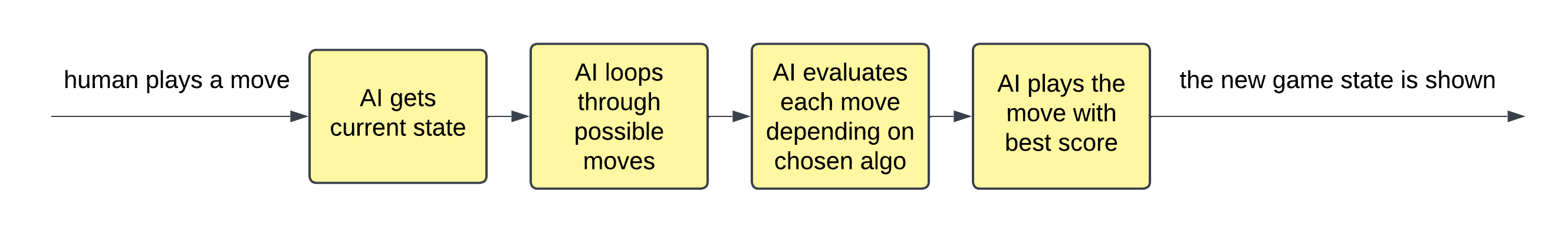
**.** Interactive GUI viewing the game board

**.** AI uses heuristics or symmetry reduction if user choose that

**.** Dark mode support

**.** Game restart option to start a new game any time

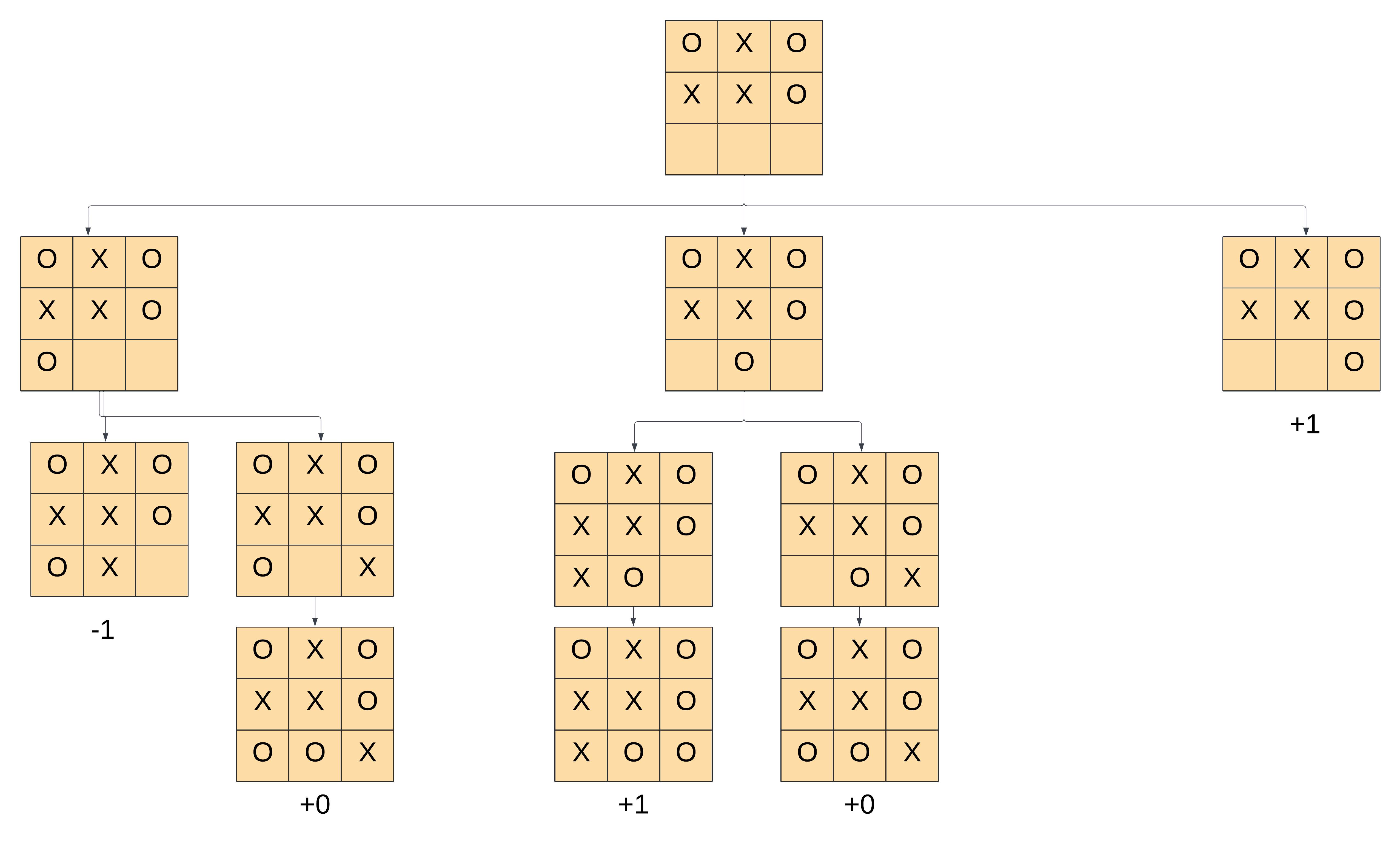
**.** Win/loss/tie detection : game detects these states and provide appropriate messages

****

**Applied algorithms**

**1)Minmax**

Explore all possible moves to find best move assuming the opponent plays optimally

****

Human first tie AI first tie

**A screenshot of a computer

Description automatically generated A screenshot of a computer

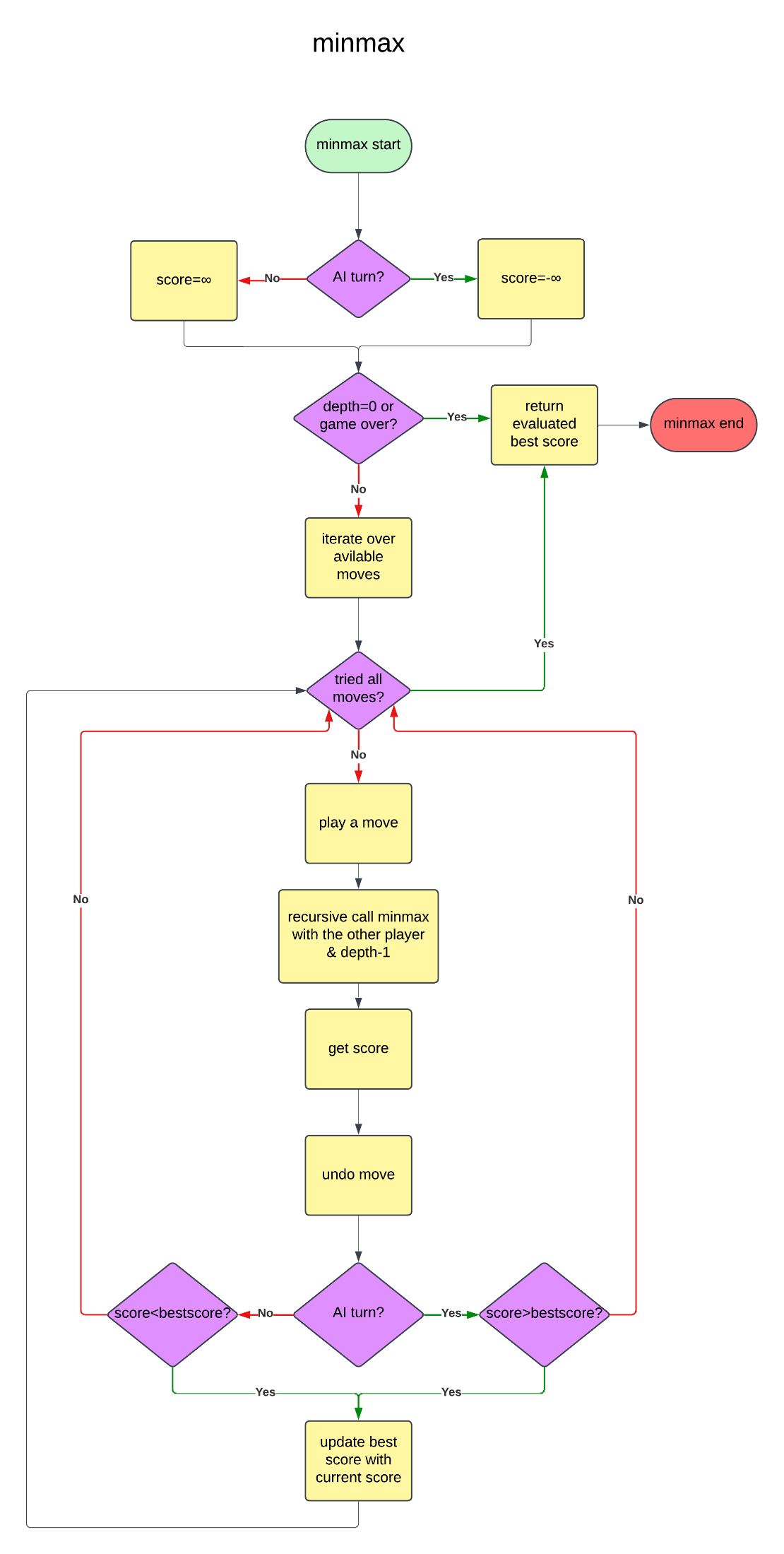
Description automatically generated**

Human first AI wins AI first AI wins

**A number on a black background

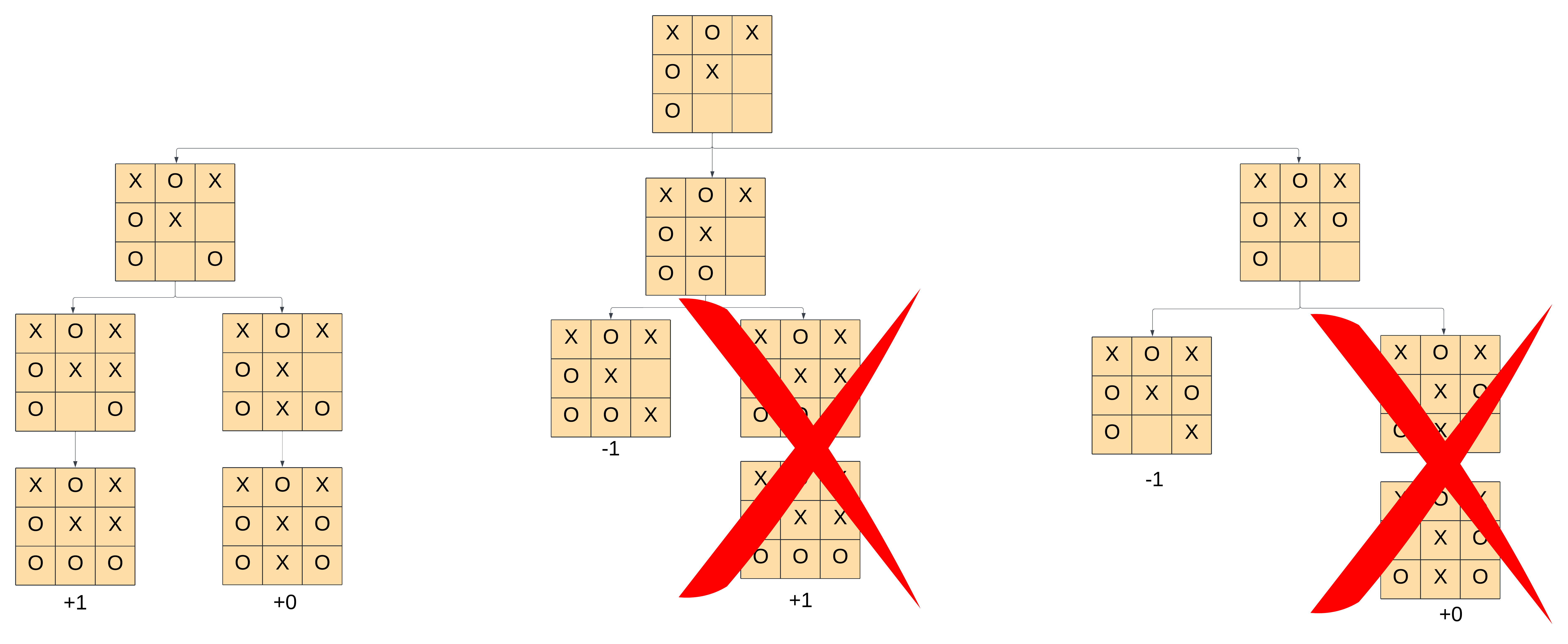
Description automatically generatedA screenshot of a computer screen

Description automatically generated**

****

**2)Alpha beta pruning**

Optimize minmax by cutting off branches that won’t affect final decision

****

Human first tie AI first tie

**A black background with white text

Description automatically generated A screenshot of a computer

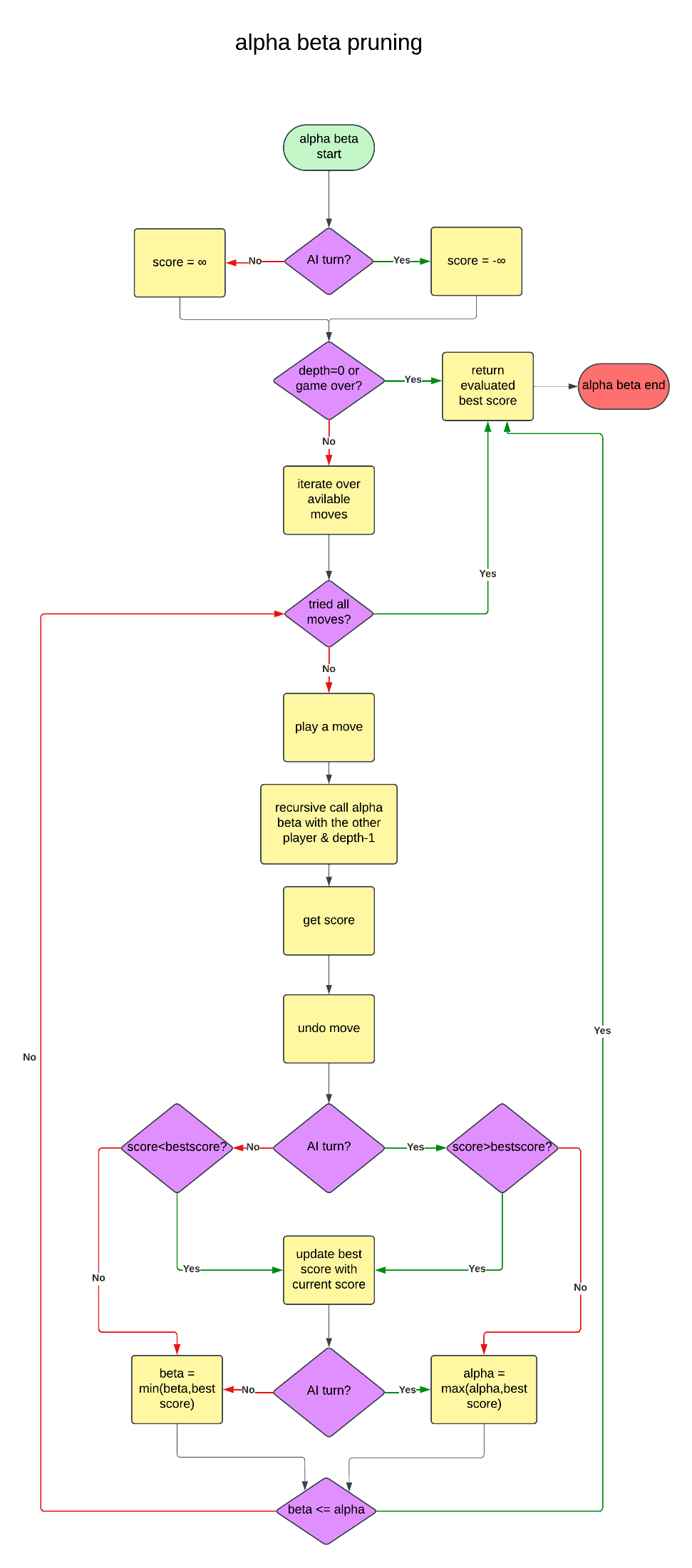
Description automatically generated**

Human first AI wins AI first AI wins

**A screenshot of a computer

Description automatically generated** **A number on a black background

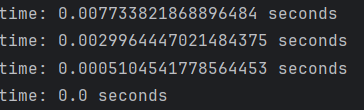
Description automatically generated**

****

**3)depth limitation**

Approximate evaluation of non-terminal states to limit tree depth by using a fixed depth

Human first tie AI first tie

** A screenshot of a computer

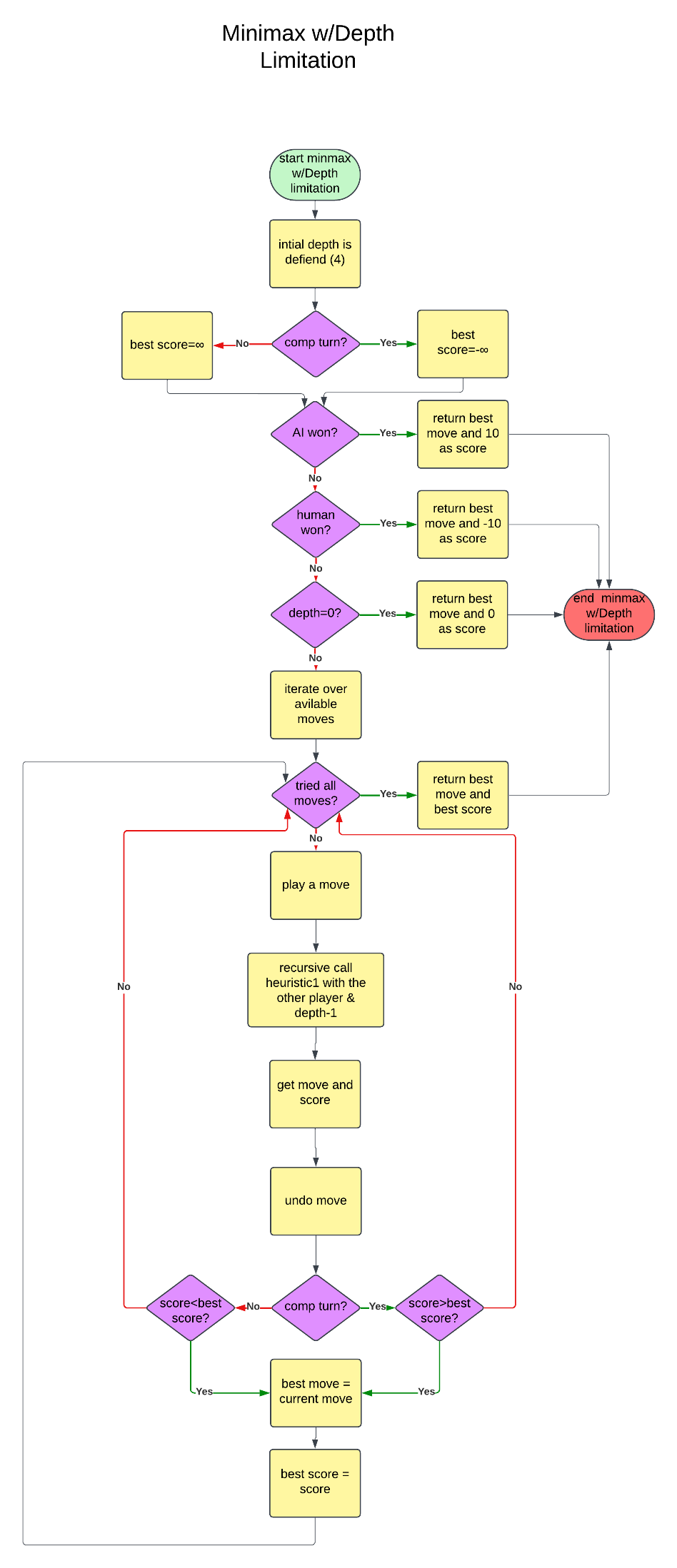
Description automatically generated**

Human first AI wins AI first AI wins

A screenshot of a computer

Description automatically generated A number on a black background

Description automatically generated

****

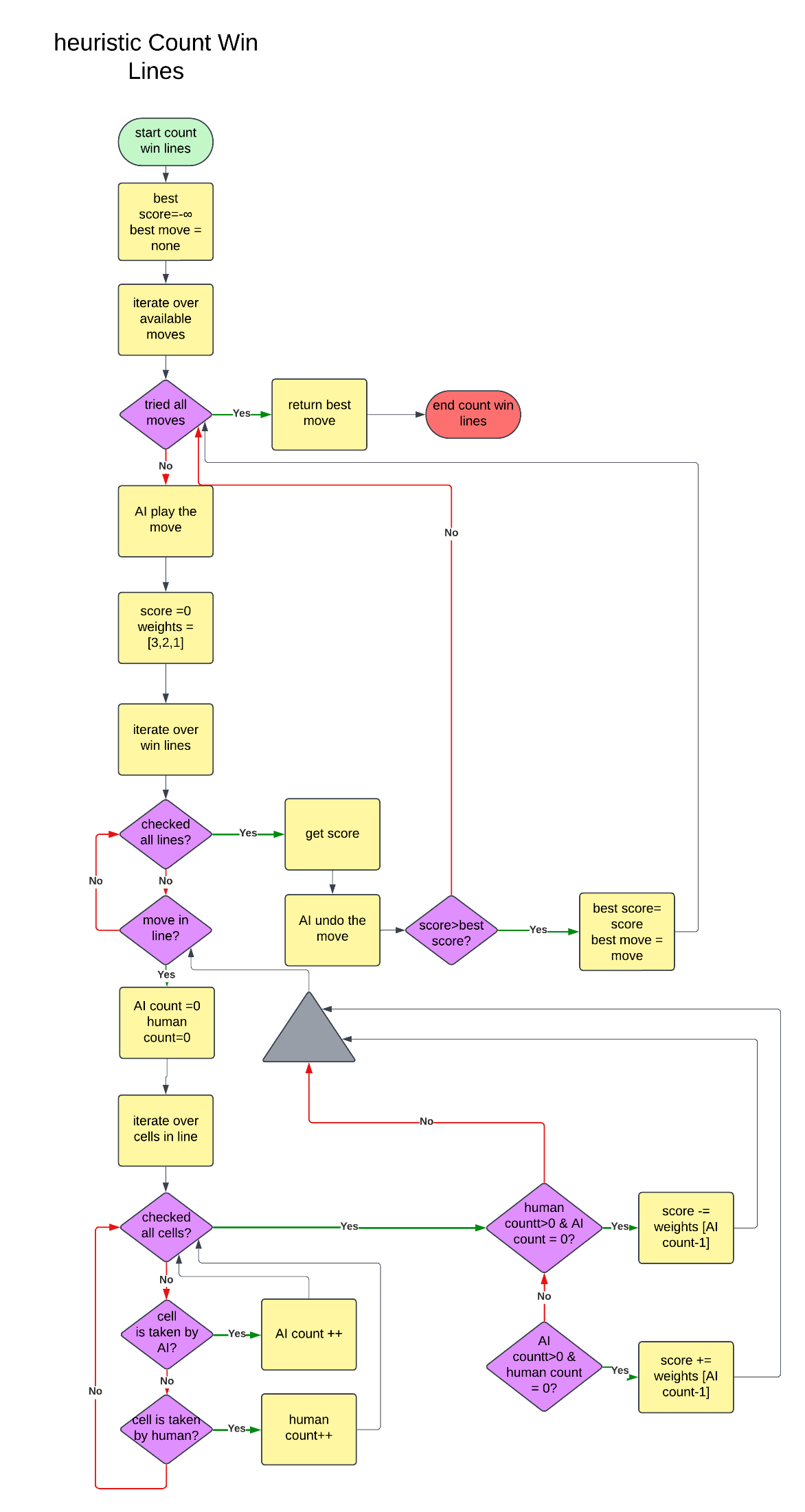
**4)heuristic count win lines**

Quick evaluation based on the potential winning lines

Time is always zero in all moves

A screen shot of a computer

Description automatically generated

****

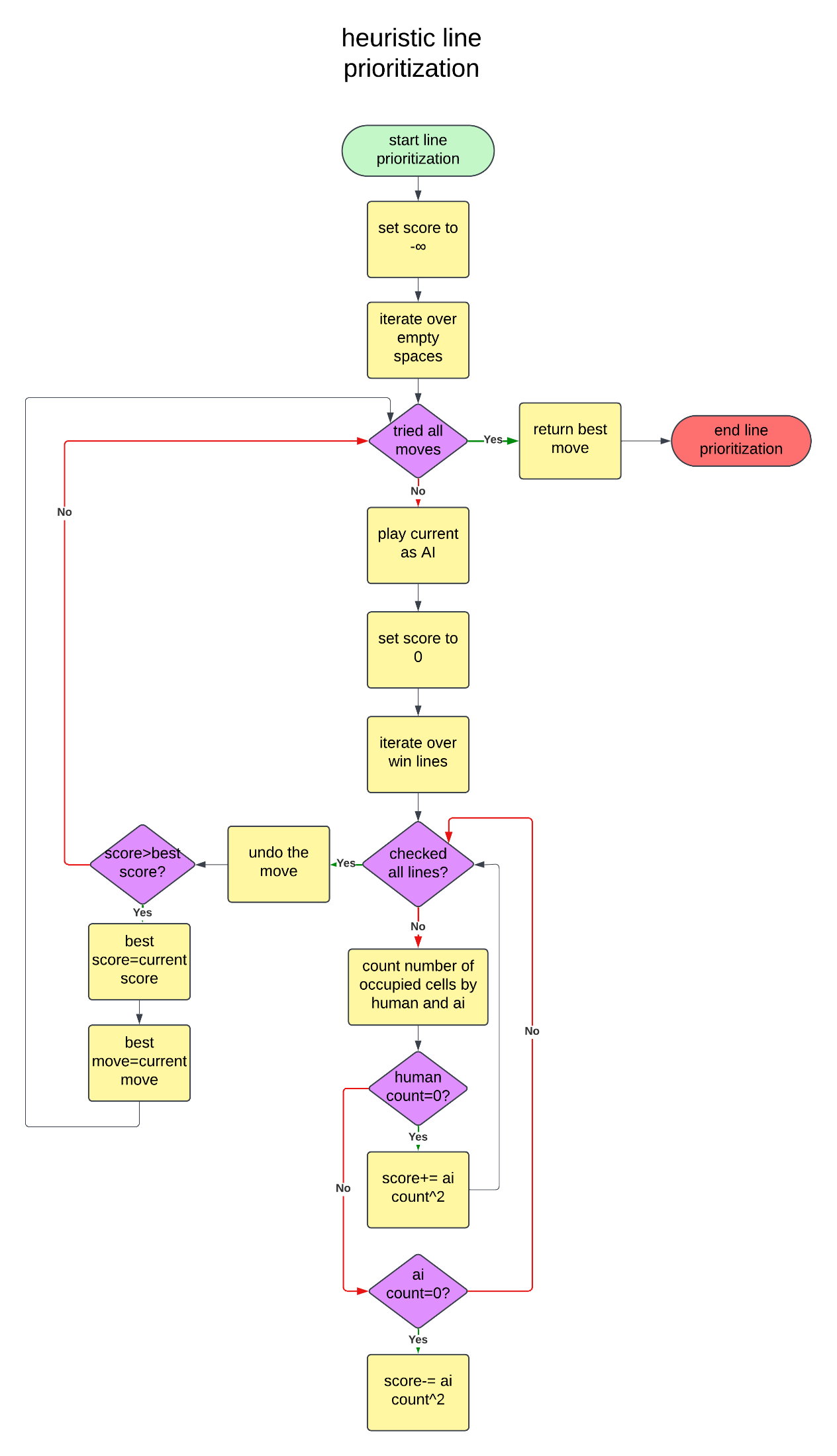
**5)heuristic line prioritization**

Prioritizes moves based on the state of win line by favoring lines with more ai cells and penalize lines with more human cells

Most of the moves time is 0 but sometimes in the opening moves it could take some time

A screen shot of a computer

Description automatically generated



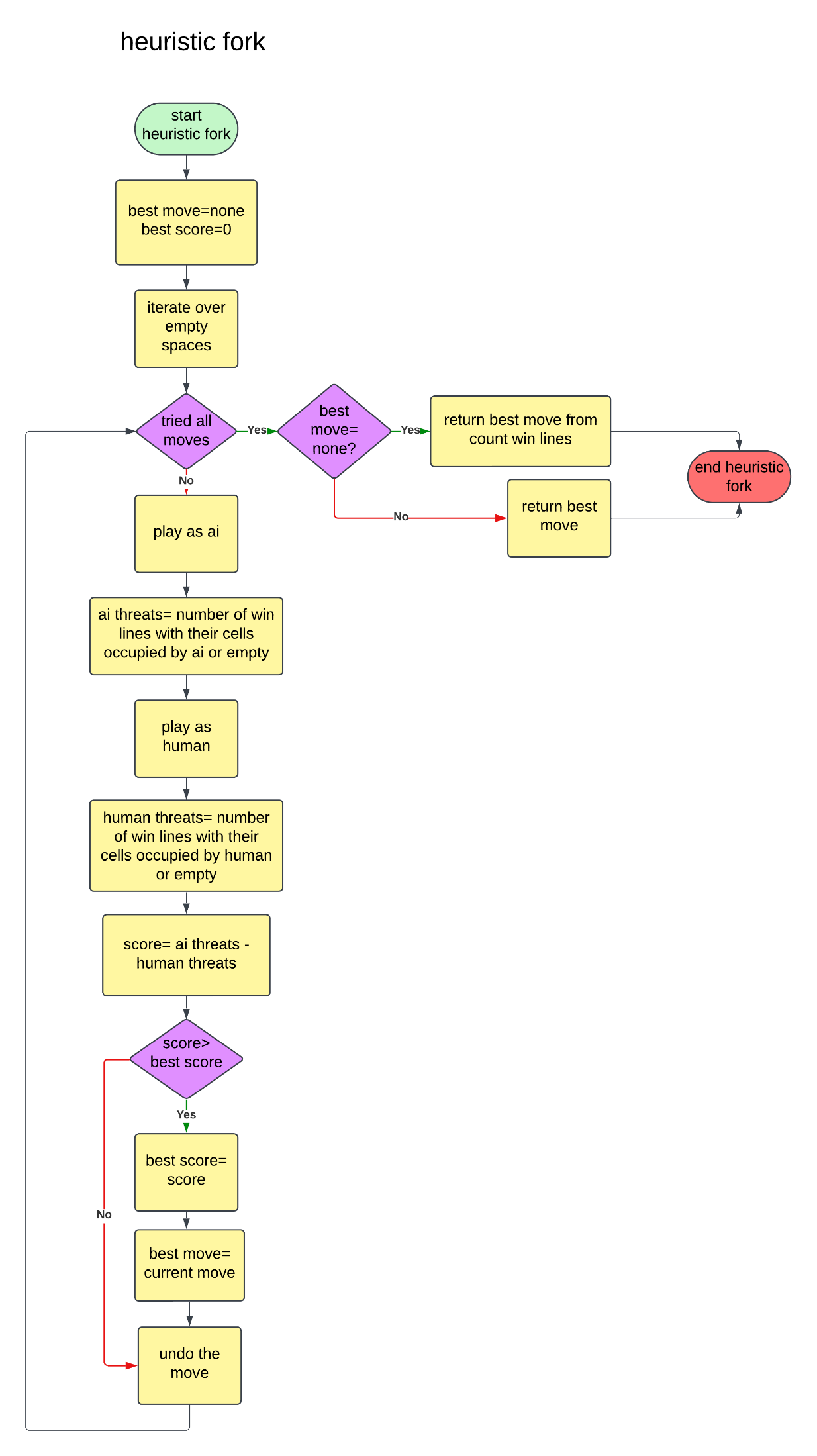
**6)heuristic fork**

Prioritizes moves based on the ability to have 2 or more winning opportunities simultaneously

Most of the moves time is 0 but sometimes in the opening moves it could take some time

A screen shot of a computer

Description automatically generated



**7)Symmetry reduction**

Reduces redundant evaluation by considering only unique board states using rotates and flips it generates all symmetric variations and evaluates the canonical form

Human first tie AI first tie

**A number on a black background

Description automatically generated A screenshot of a computer screen

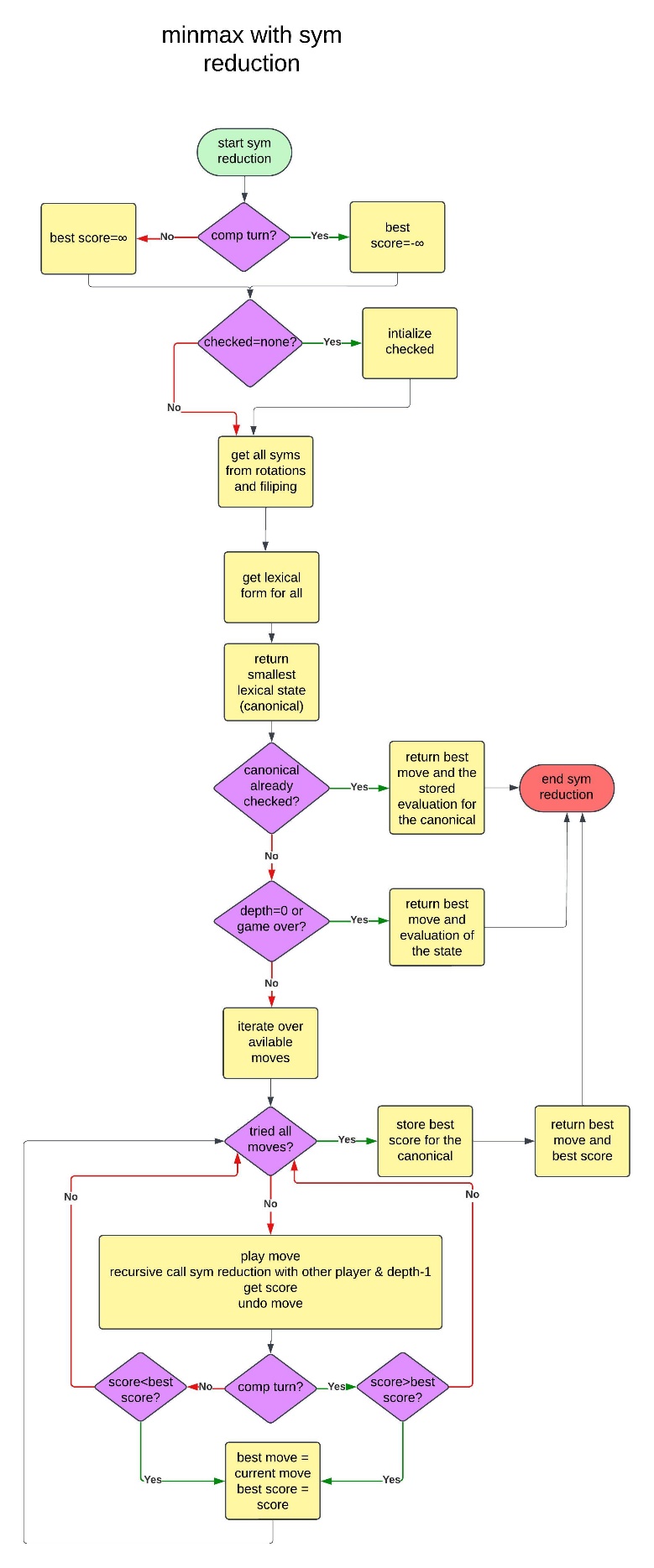
Description automatically generated**

Human first AI wins AI first AI wins

A number on a black background

Description automatically generated A number on a black background

Description automatically generated

****

**Experiments and results**

**Testing**

Verified each algorithm’s correctness by playing multiple games

**Results**

**Minmax:** guaranteed optimal results but expensive for large depth

**Alpha beta pruning:** faster than minmax without loss of optimality

**Depth limitation:** faster with acceptable decision quality

**Count win lines:** best performance for casual play but occasionally suboptimal

**Line prioritization:** better than count win lines

**Fork:** moderate speed but good in late game

**Symmetry reduction:** reduced computational time in symmetry states

**Analysis, discussion and future work**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Time complexity** | **Space complexity** | **Optimality** |
| **Minmax** | **O(b^d)** | **O(d)** | **Always optimal** |
| **Alpha beta** | **O(b^(d/2))** | **O(d)** | **Always optimal** |
| **Depth limitation** | **O(b^dmax)** | **O(dmax)** | **Approximate** |
| **Count win lines** | **O(1)** | **O(1)** | **Approximate** |
| **Line prioritization** | **O(1)** | **O(1)** | **Better approximation** |
| **fork** | **O(1)** | **O(1)** | **Strategic** |
| **Symmetry reduction** | **O(bs^d)** | **O(d)** | **Always optimal** |

Where b= branching factor

bs= branching factor with symmetry reduction

d= depth

dmax= max depth that was decided in the function

**Advantages and disadvantages**

Minmax

Advantages

Guarantees optimal play

Simple to implement

Provides deterministic outcomes

Disadvantages

Expensive for large games

Requires large memory

Alpha beta pruning

Advantages

Guarantees optimal play like minmax

Reduces computational load by pruning unnecessary branches

Speed up decision making

Makes deeper search feasible

Disadvantages

More complex to implement compared to minmax

Efficiency depends on moves order

Depth reduction

Advantages

Reduces computational load by setting a maximum depth

Speed up decision making

Easy to implement

Disadvantages

Doesn’t provide optimal solutions especially in edge cases

Count win lines

Advantages

Very low computational load

Speed up decision making

Easy to implement

intuitive

Disadvantages

Doesn’t provide optimal solutions

Doesn’t account for opponent’s moves very well

Line prioritization

Advantages

Better than count win lines

Balanced offence and defense

Disadvantages

Doesn’t provide optimal solutions

More expensive than similar algorithms

Fork

Advantages

Explores strategic opportunities

Combines both offence and defense

Disadvantages

May overlook simple winning moves for complex strategies

More expensive than basic heuristics

Application

Advantages

Demonstrates AI techniques effectively

Provides a user friendly interface

Disadvantages

Limited scalability beyond 3x3 boards

Heuristics may not perform optimally in edge cases

Future work

Extend game to larger board size

Incorporate machine learning for adaptive difficulty

Add a feature for players to analyze AI's decision-making process

Optimize heuristics especially in edge states

Conclusion

This project combines classical AI techniques and user-friendly design to create a practical Tic-Tac-Toe application. It serves as a valuable learning resource for understanding game AI fundamentals