 **BỘ THÔNG TIN VÀ TRUYỀN THÔNG**

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

------🙟🕮🙝------

Logo

Description automatically generated

**BÁO CÁO: NHẬP MÔN TRÍ TUỆ NHÂN TẠO**

**Tic tac toe (XO) ỨNG DỤNG THUẬT TOÁN MINIMAX**

Môn học: NHẬP MÔN TRÍ TUỆ NHÂN TẠO

Giảng viên: Nguyễn Thị Mai Trang

Nhóm bài tập: 05

Sinh viên nhóm: Phạm Tiến Thuận – B20DCCN678

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

**Hà Nội, tháng 5/2023**

Mục lục

[**I.** **LÝ THUYẾT VỀ THUẬT TOÁN MINIMAX** 2](#_Toc135082659)

[**1.** **Khái niệm** 2](#_Toc135082660)

[**2.** **Lịch sử hình thành** 3](#_Toc135082661)

[**3.** **Cách xây dựng thuật toán** 4](#_Toc135082662)

[**3.1** **Giải thuật Minimax** 4](#_Toc135082663)

[**3.2** **Các bước giải thuật Minimax** 5](#_Toc135082664)

[**3.3** **Ví dụ mô phỏng giải thuật Minimax cho trò chơi Tic Tac Toe** 5](#_Toc135082665)

[**3.4** **Ví dụ** 6](#_Toc135082666)

[**4.** **Ưu điểm và nhược điểm** 6](#_Toc135082667)

[**4.1** **Ưu điểm** 6](#_Toc135082668)

[**4.2** **Nhược điểm** 7](#_Toc135082669)

[ Tóm lại, thuật toán Minimax là một thuật toán tìm kiếm cây trò chơi phổ biến và mạnh mẽ cho các trò chơi hai người với lượt đi xen kẽ. Tuy nhiên, nó có những giới hạn về khả năng áp dụng cho các trò chơi lớn và phức tạp hơn 7](#_Toc135082670)

[**5.** **Các thư viện sử dụng** 7](#_Toc135082671)

[**II.** **Sản phẩm.** 8](#_Toc135082672)

[**1.** **Giới thiệu sản phẩm.** 8](#_Toc135082673)

[**2.** **Cấu trúc project** 9](#_Toc135082674)

[**BoardState**: 9](#_Toc135082675)

[**MainViewModel**: 9](#_Toc135082676)

[Lớp **MainViewModelState**: 9](#_Toc135082677)

[**MainActivity**: 10](#_Toc135082678)

[**TicTacToeScreen**: 10](#_Toc135082679)

[**3.** **Ứng dụng của thuật toán trong sản phẩm.** 11](#_Toc135082680)

[**4.** **Giới thiệu một số hình ảnh về Game.** 14](#_Toc135082681)

[**5.** **Mã nguồn project** 19](#_Toc135082682)

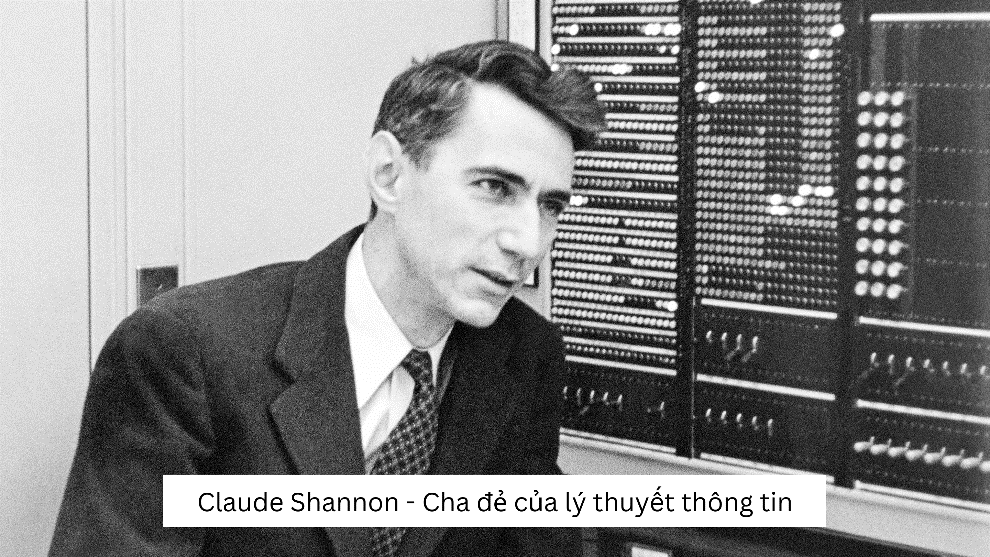
1. **LÝ THUYẾT VỀ THUẬT TOÁN MINIMAX**
2. **Khái niệm**

Thuật toán Minimax (hay còn gọi là Minmax) là một thuật toán đệ quy lựa chọn bước đi kế tiếp. Thuật toán phổ biến trong trò chơi đối kháng trong đó hai người thay phiên đi nước đi của mình như: cờ vua, game xo, tic-tac-toe, cờ vây, .... Khi chơi bạn có thể khai triển hết không gian trạng thái nhưng khó khăn chủ yếu là bạn phải tính toán được phản ứng và nước đi của đối thủ mình như thế nào? Cách xử lý đơn giản là bạn giả sử đối thủ của bạn cũng sử dụng kiến thức về không gian trạng thái giống bạn. Giải thuật Minimax áp dụng giả thuyết này để tìm kiếm không gian trạng thái của trò chơi.

1. **Lịch sử hình thành**

Trong những cột mốc đánh dấu sự hình thành và phát triển của trí tuệ nhân tạo, sự hình thành của thuật toán Minimax luôn được gọi tên như sự đóng góp rất lớn cho trí tuệ nhân tạo nói chung và thuật toán ứng dụng trong trò chơi nói riêng.

Claude Shannon, được coi là cha đẻ của lý thuyết thông tin, đã đóng góp quan trọng cho sự phát triển của thuật toán Minimax. Năm 1950, ông đã viết bài báo "Programming a Computer for Playing Chess" (Lập trình một máy tính chơi cờ vua) - bài báo đầu tiên về phát triển một phần mềm máy tính chơi cờ vua. Bài báo trình bày phương pháp sử dụng thuật toán Minimax để lập kế hoạch cho một chương trình chơi cờ vua trên máy tính. Ông đã đề xuất sử dụng một bảng đánh giá vị trí của các quân cờ để tính toán giá trị của các trạng thái trong trò chơi.



Năm 1956, cụm từ “Artificial Intelligence” (trí tuệ nhân tạo) được đề xuất bởi nhà khoa học máy tính John McCarthy tại hội nghị Dartmouth, đánh dấu mốc chính thức ra đời của trí tuệ nhân tạo. McCarthy vừa là cha đẻ của trí tuệ nhân tạo, còn là cha đẻ của ngôn ngữ lập trình Lisp. Trong thời gian ấy, bài báo “The Dartmouth Conference” (hội nghị Dartmouth) của John McCarthy đã đề cập đến việc sử dụng thuật toán Minimax để tạo ra một chương trình chơi trò chơi tic-tac-toe trên máy tính.



Từ đó đến nay, đã có rất nhiều nghiên cứu và bài báo về thuật toán Minimax, có thể kể đến như: “Dynamic programming and minimax algorithms” (Quy hoạch động và thuật toán Minimax) – Bellman, R. (1957); “Analysis of a simple game using minimax” (Phân tích một trò chơi đơn giản sử dụng Minimax) – Knuth, D.E. (1975); “Artificial intelligence: A modern approach” (Trí tuệ nhân tạo – Cách tiếp cận hiện đại) - Russell, S., & Norvig, P. (2010).

Và kể từ ấy, thuật toán Minimax đã được sử dụng rộng rãi trong các trò chơi hai người với lượt đi xen kẽ, và được coi là một trong những thuật toán cơ bản nhất của trí tuệ nhân tạo. Các biến thể của thuật toán này như **Alpha-Beta pruning** và **Monte Carlo Tree Search** cũng đã được phát triển để tăng tốc độ tính toán và cải thiện hiệu quả của thuật toán.

1. **Cách xây dựng thuật toán**
   1. **Giải thuật Minimax**

Hai đối thủ trong trò chơi được gọi là MIN và MAX luân phiên thay thế nhau đi. MAX đại diện cho người quyết dành thắng lợi và cố gắng tối đa hóa ưu thế của mình, ngược lại người chơi đại diện cho MIN lại cố gắng giảm điểm số của MAX và cố gắng làm cho điểm số của mình càng âm càng tốt. Giả thiết đưa ra MIN và MAX có kiến thức như nhau về không gian trạng thái trò chơi và cả hai đối thủ đều cố gắng như nhau.

Mỗi Node biểu diễn cho một trạng thái trên cây trò chơi. Node lá là Node chứa trạng thái kết thúc của trò chơi.

Giải thuật Minimax thể hiện bằng cách định trị các Node trên cây trò chơi:

* Node thuộc lớp MAX thì gán cho nó giá trị lớn nhất của con Node đó.
* Node thuộc lớp MIN thì gán cho nó giá trị nhỏ nhất của con Node đó.

Từ các giá trị này người chơi sẽ lựa chọn cho mình nước đi tiếp theo hợp lý nhất.

* 1. **Các bước giải thuật Minimax**
* Nếu như đạt đến giới hạn tìm kiếm (đến tầng dưới cùng của cây tìm kiếm tức là trạng thái kết thúc của trò chơi).
* Tính giá trị của thế cờ hiện tại ứng với người chơi ở đó. Ghi nhớ kết quả.
* Nếu như mức đang xét là của người chơi cực tiểu (nút MIN), áp dụng thủ tục Minimax này cho các con của nó. Ghi nhớ kết quả nhỏ nhất.
* Nếu như mức đang xét là của người chơi cực đại (nút MAX), áp dụng thủ tục Minimax này cho các con của nó. Ghi nhớ kết quả lớn nhất.
  1. **Ví dụ mô phỏng giải thuật Minimax cho trò chơi Tic Tac Toe**

Quy ước:

* MAX đại diện quân đi O.
* MIN đại diện quân đi X.

Trạng thái kết thúc là trạng thái có 5 ô liên tiếp ngang, dọc, chéo có cùng một quân cờ X hoặc O, nếu là X tức MIN thắng còn O tức MAX thắng còn nếu tất cả các ô cờ đều được đi và trạng thái chưa kết thúc thì bàn cờ hòa. Điểm thắng của X là -1, của O là 1, và bàn cờ hòa là 0.

Áp dụng giải thuật Minimax: Từ trạng thái bàn cờ hiện tại ta dự đoán nước đi của trạng thái tiếp theo nếu trạng thái tiếp theo ta tiến hành lượng giá cây trò chơi bằng cách ta tiến hành quét cạn tất cả các trạng thái tiếp theo cho đến lúc gặp trạng thái chiến thắng (Node lá) tính điểm cho Node lá bằng cách:

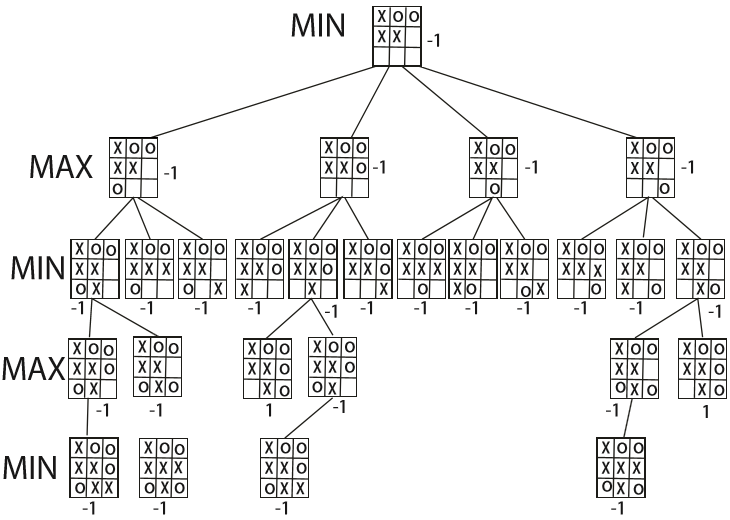
* Nếu ở trạng thái mà ta gặp chiến thắng nếu đó là lượt đi của quân X thì đánh giá điểm trạng thái đó là -1.
* Nếu ở trạng thái ta gặp chiến thắng nếu đó là lượt đi của quân O thì đánh giá điểm trạng thái đó là 1.
* Nếu là hòa thì điểm trạng thái đó là 0.

Sau đó tính ngược lại cây trò chơi theo quy tắc:

* Nút thuộc lớp MAX thì gán cho nó giá trị lớn nhất của các Node con của Node đó.
* Nút thuộc lớp MAX thì gán cho nó giá trị nhỏ nhất của các Node con của Node đó.

Sau khi lượng giá hết cây trò chơi ta tiến hành chọn bước đi tiếp theo nguyên tắc:

* Nếu lớp tiếp theo là MAX ta chọn Node con có giá trị lớn nhất.
* Nếu lớp tiếp theo là MIN ta chọn Node con có giá trị nhỏ nhất.
  1. **Ví dụ**



1. **Ưu điểm và nhược điểm**
   1. **Ưu điểm**

* Đảm bảo tìm được chiến lược tối ưu cho một người chơi trong trò chơi hai người với lượt đi xen kẽ.
* Dễ hiểu và áp dụng.
* Tính toán độ chính xác cao khi số lượng các trạng thái trong trò chơi không quá lớn.
* Có thể được mở rộng để giải quyết các trò chơi phức tạp hơn bằng cách sử dụng các biến thể như Alpha-Beta pruning hoặc Monte Carlo Tree Search.
  1. **Nhược điểm**
* Độ phức tạp tính toán của thuật toán tăng rất nhanh khi kích thước của cây trò chơi tăng, làm cho việc tính toán trở nên không thực tế cho các trò chơi lớn.
* Cần phải tính toán toàn bộ cây trò chơi trước khi đưa ra quyết định, làm tốn thời gian và tài nguyên tính toán.
* Chỉ có thể được áp dụng cho các trò chơi với lượt đi xen kẽ và có số lượng trạng thái hữu hạn, không áp dụng cho các trò chơi thời gian thực hoặc trò chơi với số lượng trạng thái vô hạn.
* Tóm lại, thuật toán Minimax là một thuật toán tìm kiếm cây trò chơi phổ biến và mạnh mẽ cho các trò chơi hai người với lượt đi xen kẽ. Tuy nhiên, nó có những giới hạn về khả năng áp dụng cho các trò chơi lớn và phức tạp hơn

1. **Các thư viện sử dụng**

Các thư viện sử dụng để triển khai thuật toán Minimax phụ thuộc vào ngôn ngữ lập trình được sử dụng. Dưới đây là một số thư viện phổ biến được sử dụng trong các ngôn ngữ lập trình khác nhau:

* Python:
* Pygame: một thư viện hỗ trợ viết game với Python, bao gồm các phương thức hỗ trợ cho thuật toán Minimax.
* Numpy: thư viện hỗ trợ xử lý các phép tính toán số học và đại số tuyến tính, được sử dụng trong thuật toán Minimax.
* Java:
* JGameGrid: một thư viện hỗ trợ viết game với Java, bao gồm các phương thức hỗ trợ cho thuật toán Minimax.
* JME: thư viện đồ họa 3D dành cho Java, cũng có thể được sử dụng để triển khai thuật toán Minimax trong các trò chơi.
* C++:
* SFML: một thư viện đồ họa và âm thanh dành cho C++, cũng hỗ trợ thuật toán Minimax.
* SDL: một thư viện đa nền tảng hỗ trợ đồ họa, âm thanh và nhập liệu dành cho C++, có thể được sử dụng để viết các trò chơi và triển khai thuật toán Minimax.
* C#:
* Unity Game Engine: là một trong những công cụ phổ biến nhất để phát triển game trên nhiều nền tảng, bao gồm cả nền tảng di động. Unity cung cấp một số thư viện và phương thức hỗ trợ để triển khai thuật toán Minimax trong các trò chơi.
* SharpDX: là một thư viện dành cho C# cho phép truy cập trực tiếp vào DirectX API. Thư viện này hỗ trợ triển khai các trò chơi 3D và cung cấp một số phương thức hỗ trợ cho thuật toán Minimax.
* XNA Game Studio: là một bộ công cụ phát triển game dành cho Microsoft Windows và Xbox 360. XNA cung cấp một số thư viện và phương thức hỗ trợ cho thuật toán Minimax.
* AForge.NET: là một thư viện hỗ trợ xử lý hình ảnh và thị giác máy tính, cũng có thể được sử dụng để triển khai thuật toán Minimax trong các trò chơi.

1. **Sản phẩm.**
2. **Giới thiệu sản phẩm.**

**Tic-Tac-Toe** là sản phầm game làm bằng Kotlin được xây dựng trên nền tảng android kết hợp với xây dựng bot AI làm bằng thuật toán Minimax.

Game có các chức năng là:

* Chơi người với người
* Chơi người với bot
  + Bot có 2 chế độ:
    - Easy
    - Hard

1. **Cấu trúc project**

**A screenshot of a computer

Description automatically generated**

**Hình 1. Cấu trúc Project**

Cấu trúc project gồm có 5 thành phần chính:

**BoardState**:

Đây là lớp đại diện cho trạng thái hiện tại của bảng chơi. Nó chứa vị trí hiện tại (position) và trạng thái của bảng chơi (board).

Hàm getStateIndex(i: Int): Trả về trạng thái của ô thứ i trên bảng chơi.

Hàm changeState(i: Int, player: String): Thay đổi trạng thái của ô thứ i trên bảng chơi dựa vào người chơi ('X' hoặc 'O').

**MainViewModel**:

Lớp này là một ViewModel, nó chứa trạng thái của ứng dụng và cung cấp các phương thức để cập nhật trạng thái này.

Lớp **MainViewModelState**:

Chứa các thông tin về trạng thái hiện tại của game như trạng thái bảng chơi, người chơi hiện tại, người chiến thắng, ...

* Hàm updateState(state: State): Cập nhật trạng thái của game.
* Hàm play(move: Int = -1): Thực hiện một nước đi. Nếu là lượt người chơi, nó sẽ cập nhật trạng thái của bảng và chuyển lượt cho máy. Nếu là lượt máy, nó sẽ tính toán nước đi tốt nhất bằng thuật toán Minimax và cập nhật trạng thái bảng.
* Hàm resetGame(): Đặt lại trạng thái của game về ban đầu.

**MainActivity**:

Đây là hoạt động chính của ứng dụng, nơi mà giao diện người dùng được tạo và quản lý.

Trong hàm onCreate(), giao diện người dụng được thiết lập thông qua hàm setContent(). Trạng thái của game được lấy từ ViewModel và cập nhật trên giao diện người dụng.

Giao diện bao gồm một Scaffold với một TopAppBar, một ButtonGrid để hiển thị bảng chơi và một nút ResetButton để người chơi có thể đặt lại trạng thái của game.

**TicTacToeScreen**:

Đây là nơi vẽ UI ứng dụng.

**Minimax:**

Đây sẽ chứa logic thuật toán Minimax sẽ được ứng dụng trong sản phẩm.Gồm có thành phần sau:

* minMaxDecisionModeEase và minMaxDecisionModeHard đều là các phương thức để quyết định nước đi tiếp theo dựa trên trạng thái hiện tại của bảng. Sự khác biệt giữa hai chế độ là cách chúng tạo ra danh sách các nước đi tiếp theo. minMaxDecisionModeEasy chỉ cần gọi hàm minValue cho mỗi trạng thái tiếp theo có thể có, trong khi minMaxDecisionModeHard sẽ gọi hàm minimax để xác định giá trị tốt nhất có thể đạt được từ trạng thái đó.
* Phương thức minimax là trái tim của thuật toán Minimax. Nó sẽ duyệt qua tất cả các trạng thái tiếp theo có thể có từ trạng thái hiện tại và tính toán giá trị tối ưu nhất có thể đạt được từ mỗi trạng thái. Nếu đang ở chế độ tối đa hóa, nó sẽ chọn giá trị lớn nhất từ các trạng thái tiếp theo; nếu đang ở chế độ tối thiểu hóa, nó sẽ chọn giá trị nhỏ nhất.
* maxValue và minValue được sử dụng để tính toán giá trị tối đa và tối thiểu mà một người chơi có thể đạt được từ một trạng thái nhất định, dựa trên trạng thái tiếp theo có thể có.
* isTerminal kiểm tra xem một trạng thái có phải là trạng thái kết thúc hay không - nghĩa là, nếu một người chơi đã thắng hoặc tất cả các ô đều đã được điền.
* utilityOf trả về điểm số cho một trạng thái nhất định. Trong trò chơi Tic-Tac-Toe, điểm số là +10 nếu X thắng, -10 nếu O thắng, và 0 nếu không ai thắng.
* checkState kiểm tra xem một dòng, cột hoặc đường chéo có tạo thành một chuỗi thắng hay không.
* successorsOf trả về tất cả các trạng thái tiếp theo có thể có từ trạng thái hiện tại. Điều này được thực hiện bằng cách kiểm tra từng ô trên bảng và nếu ô đó chưa được điền (không phải là "X" hoặc "O"), nó sẽ tạo ra một trạng thái mới với người chơi tiếp theo điền

1. **Ứng dụng của thuật toán trong sản phẩm.**

Trên đây là đoạn code để kiểm tra mode chơi hiện tại và quyết định nước đi tiếp theo cho AI:

#Minimax với sử dụng depth (Được sử dụng trong chế độ bot Hard)

-----------------------------------------------------------------------------------------------------

private fun minimax(boardState: BoardState, depth: Int, isMaximizingPlayer: Boolean): Int {  
 if (isTerminal(boardState)) {  
 val score = utilityOf(boardState)  
 if (score == 10) {  
 Log.i("X winn", "minimax: $score - $depth $isMaximizingPlayer")  
 return score - depth  
 } else if (score == -10) {  
 Log.i("O win", "minimax: $score + $depth $isMaximizingPlayer")  
 return score + depth  
 }  
 return score  
 }  
 if (isMaximizingPlayer) {  
 var bestValue = -1000  
 for (possibleMove in successorsOf(boardState)) {  
 val value = minimax(possibleMove, depth + 1, false)  
 bestValue = Math.max(bestValue, value)  
 Log.i("Max value", "minimax: $value $bestValue + $depth $isMaximizingPlayer")  
 }  
 return bestValue  
 } else {  
 var bestValue = 1000  
 for (possibleMove in successorsOf(boardState)) {  
 val value = minimax(possibleMove, depth + 1, true)  
 bestValue = Math.min(bestValue, value)  
 Log.i("Min value", "minimax: $value $bestValue + $depth $isMaximizingPlayer")  
  
 }  
 return bestValue  
 }  
}

-----------------------------------------------------------------------------------------------------

Giải thích: Hàm này thực hiện thuật toán Minimax để tìm nước đi tốt nhất cho máy.

* Tham số boardState biểu diễn trạng thái hiện tại của bàn cờ.
* Tham số depth là độ sâu của cây trò chơi, giúp đánh giá độ tốt của nước đi. Độ sâu càng thấp thì nước đi càng tốt.
* Tham số isMaximizingPlayer xác định xem người chơi hiện tại có đang cố gắng tối đa hóa điểm số hay không. Nếu là true, nghĩa là AI (X) đang chơi và ngược lại.
* Trong hàm này, nếu trạng thái hiện tại của bàn cờ là trạng thái kết thúc trò chơi (isTerminal(boardState) trả về true), hàm sẽ trả về điểm của trạng thái đó (thắng, thua, hòa). Nếu không, hàm sẽ duyệt qua tất cả các nước đi có thể (successorsOf(boardState)) và gọi đệ quy hàm minimax cho mỗi nước đi.

#Với minimax không sử dụng depth (Được sử dụng trong chế độ bot Easy)

-----------------------------------------------------------------------------------------------------

*//Picks best option for the X-player*private fun maxValue(boardState: BoardState): Int {  
 if (isTerminal(boardState)) {  
 return utilityOf(boardState)  
 }  
 var v = (-Double.POSITIVE\_INFINITY).toInt()  
 for (possibleMove in successorsOf(boardState)) {  
 v = Math.max(v, minValue(possibleMove))  
 }  
 Log.i("Minimax Max", v.toString())  
 return v  
}  
  
*//Picks best option for the O-player*private fun minValue(boardState: BoardState): Int {  
 if (isTerminal(boardState)) {  
 return utilityOf(boardState)  
 }  
 var v = Double.POSITIVE\_INFINITY.toInt()  
 for (possibleMove in successorsOf(boardState)) {  
 v = Math.min(v, maxValue(possibleMove))  
 }  
 Log.i("Minimax Min", v.toString())  
 return v  
}

-----------------------------------------------------------------------------------------------------

Giải thích: Tương tự hàm minimax trên nhưng không sử dụng depth để tối ưu điểm số

## **Giới thiệu một số hình ảnh về Game.**

A screenshot of a cell phone

Description automatically generated with medium confidence

Hình 2: Hình ảnh gamePlay

A screenshot of a cell phone

Description automatically generated with medium confidence

Hình 3: Hình ảnh game single Play chế độ dễ

A screenshot of a cell phone

Description automatically generated with medium confidence

Hình 4: Hình ảnh game Multiplayer

A screenshot of a game

Description automatically generated

Hình 5: Hình ảnh phe X chiến thắng

1. **Mã nguồn project**

<https://github.com/thuanpham582002/Tic-Tac-Toe-Jetpack-Compose>

A qr code with black squares

Description automatically generated with low confidence