**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TP.HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**Logo

Description automatically generated**-----o0o----

**TIỂU LUẬN HỌC PHẦN:**

**THỰC HÀNH TRÍ TUỆ NHÂN TẠO**

**TÊN ĐỀ TÀI: ÁP DỤNG THUẬT TOÁN MINIMAX CHO**

**TRÒ CHƠI TIC-TAC-TOE**

**Thành phố Hồ Chí Minh, tháng 12 năm 2022**

**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TP.HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**Logo

Description automatically generated**-----o0o----

**TIỂU LUẬN HỌC PHẦN:**

**THỰC HÀNH TRÍ TUỆ NHÂN TẠO**

**TÊN ĐỀ TÀI: ÁP DỤNG THUẬT TOÁN MINIMAX CHO**

**TRÒ CHƠI TIC-TAC-TOE**

|  |  |
| --- | --- |
| **Nhóm 1** | **Giảng viên hướng dẫn** |
| 1. Nguyễn Phương Việt – 2001207016   (Nhóm trưởng)   1. Đặng Ngọc Thịnh - 2001206912 | **Ngô Dương Hà** |

**Thành phố Hồ Chí Minh, tháng 12 năm 2022**

# **LỜI CAM ĐOAN**

Chúng em xin cam đoan đề tài tiểu luận: “ ÁP DỤNG THUẬT TOÁN MINIMAX CHO

TRÒ CHƠI TIC-TAC-TOE ” do nhóm chúng em nghiên cứu và thực hiện.

Kết quả bài làm của đề tài là trung thực và không sao chép từ bất kỳ bài tập của nhóm khác hay mua bán bất kỳ nơi nào.

Các tài liệu được sử dụng trong tiểu luận có nguồn gốc, xuất xứ rõ ràng và có đường link tham khảo chi tiết.

**(Ký và ghi rõ họ tên)**

**Nguyễn Phương Việt**

# **MỤC LỤC**

[LỜI CAM ĐOAN 1](#_Toc123261740)

[CHƯƠNG 1: THUẬT TOÁN MINIMAX 3](#_Toc123261741)

[1.1 Thuật toán minimax 3](#_Toc123261742)

[1.2 Hoạt động của thuật toán minimax 3](#_Toc123261743)

[CHƯƠNG 2: TRÒ CHƠI TIC-TAC-TOE 4](#_Toc123261744)

[2.1 Giới thiệu trò chơi Tic-Tac-Toe 4](#_Toc123261745)

[2.2 Giải thuật Minimax cho trò chơi Tic-Tac-toe 4](#_Toc123261746)

[2.3 Áp dụng giải thuật 5](#_Toc123261747)

[CHƯƠNG 3: GIAO DIỆN 12](#_Toc123261748)

[3.1 Giao diện chính 12](#_Toc123261749)

[3.2 Kết quả 13](#_Toc123261750)

[Tài liệu tham khảo 14](#_Toc123261751)

# **CHƯƠNG 1: THUẬT TOÁN MINIMAX**

## **Thuật toán minimax**

Thuật toán minimax là một thuật toán recursive hoặc backtrack được sử dụng trong việc ra quyết định và lý thuyết trò chơi. Nó cung cấp một nước đi tối ưu cho người chơi giả sử rằng đối thủ cũng đang chơi một cách tối ưu.

Thuật toán minimax sử dụng recursive để tìm kiếm thông qua Game Tree.

Thuật toán minimax chủ yếu được sử dụng để chơi trò chơi trong AI. Chẳng hạn như Cờ vua, Cờ caro, tic-tac-toe, cờ vây và các trò chơi kéo xe khác nhau. Thuật toán này tính toán quyết định minimax cho trạng thái hiện tại.

Trong thuật toán này, hai người chơi trò chơi, một người được gọi là MAX và người kia được gọi là MIN.

Cả hai người chơi chiến đấu với nó vì người chơi đối phương nhận được lợi ích tối thiểu trong khi họ nhận được lợi ích tối đa.

Cả hai Người chơi của trò chơi là đối thủ của nhau, trong đó MAX sẽ chọn giá trị tối đa và MIN sẽ chọn giá trị nhỏ nhất.

Thuật toán minimax thực hiện thuật toán tìm kiếm theo chiều sâu để khám phá Game Tree hoàn chỉnh.

Thuật toán minimax tiến hành tất cả các cách xuống nút đầu cuối của cây, sau đó quay ngược lại cây dưới dạng recursive.

## **Hoạt động của thuật toán 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.

# **CHƯƠNG 2: TRÒ CHƠI TIC-TAC-TOE**

## **Giới thiệu trò chơi Tic-Tac-Toe**

**Tic-tac-toe** là một trò chơi phổ biến dùng viết trên bàn cờ giấy có chín ô, 3x3. Hai người chơi, người dùng ký hiệu **O**, người kia dùng ký hiệu **X**, lần lượt điền ký hiệu của mình vào các ô. Người thắng là người thể tạo được đầu tiên một dãy ba ký hiệu của mình, ngang dọc hay chéo đều được.

A picture containing shape

Description automatically generated

## **Giải thuật Minimax cho trò chơi Tic-Tac-toe**

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

Trạng thái kết thúc là trạng thái có 3 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:

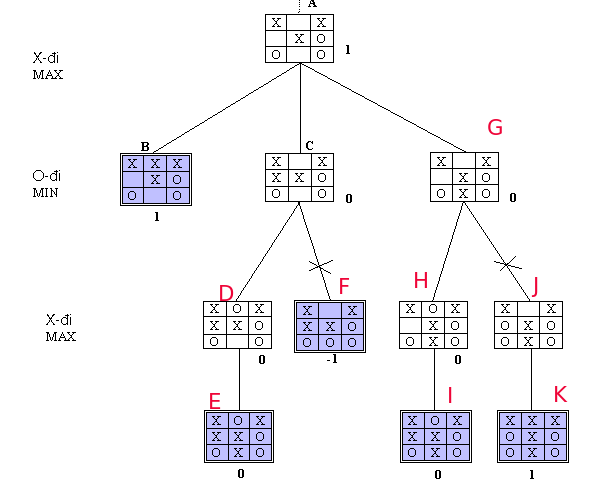
* Nếu ở trạng thái mà ta gặp chiến thắng nếu đó là lượt đi của quân X  thì đánh giá điểm trạng thái đó là -1.
* Nếu ở trạng thái ta gặp chiến thắng nếu đó là lượt đi của  quân O thì đánh giá điểm trạng thái đó là 1.
* Nếu là hòa thì điểm trạng thái đó là 0.

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.

Ví dụ

## **Áp dụng giải thuật**

"""

Tic Tac Toe Player

"""

import math

import copy

X = "X"

O = "O"

EMPTY = None

def initial\_state():

    """

    Returns starting state of the board.

    """

    return [[EMPTY, EMPTY, EMPTY],

            [EMPTY, EMPTY, EMPTY],

            [EMPTY, EMPTY, EMPTY]]

def player(board):

    """

    Returns player who has the next turn on a board.

    """

    xCounter = 0

    oCounter = 0

    for i in range(0, len(board)):

        for j in range(0, len(board[0])):

            if board[i][j] == X:

                xCounter += 1

            elif board[i][j] == O:

                oCounter += 1

    if xCounter > oCounter:

        return O

    else:

        return X

def actions(board):

    """

    Returns set of all possible actions (i, j) available on the board.

    """

    possibleActions = set()

    for i in range(0, len(board)):

        for j in range(0, len(board[0])):

            if board[i][j] == EMPTY:

                possibleActions.add((i, j))

    return possibleActions

def result(board, action):

    """

    Returns the board that results from making move (i, j) on the board.

    """

    # Create new board, without modifying the original board received as input

    result = copy.deepcopy(board)

    result[action[0]][action[1]] = player(board)

    return result

def winner(board):

    """

    Returns the winner of the game, if there is one.

    """

    # Check rows

    if all(i == board[0][0] for i in board[0]):

        return board[0][0]

    elif all(i == board[1][0] for i in board[1]):

        return board[1][0]

    elif all(i == board[2][0] for i in board[2]):

        return board[2][0]

    # Check columns

    elif board[0][0] == board[1][0] and board[1][0] == board[2][0]:

        return board[0][0]

    elif board[0][1] == board[1][1] and board[1][1] == board[2][1]:

        return board[0][1]

    elif board[0][2] == board[1][2] and board[1][2] == board[2][2]:

        return board[0][2]

    # Check diagonals

    elif board[0][0] == board[1][1] and board[1][1] == board[2][2]:

        return board[0][0]

    elif board[0][2] == board[1][1] and board[1][1] == board[2][0]:

        return board[0][2]

    else:

        return None

def terminal(board):

    """

    Returns True if game is over, False otherwise.

    """

    if winner(board) is not None or (not any(EMPTY in sublist for sublist in board) and winner(board) is None):

        return True

    else:

        return False

    #return True if winner(board) is not None or (not any(EMPTY in sublist for sublist in board) and winner(board) is None) else False # noqa E501

def utility(board):

    """

    Returns 1 if X has won the game, -1 if O has won, 0 otherwise.

    """

    if terminal(board):

        if winner(board) == X:

            return 1

        elif winner(board) == O:

            return -1

        else:

            return 0

    # Check how to handle exception when a non terminal board is received.

def minimax(board):

    """

    Returns the optimal action for the current player on the board.

    """

    if terminal(board):

        return None

    else:

        if player(board) == X:

            value, move = max\_value(board)

            return move

        else:

            value, move = min\_value(board)

            return move

def max\_value(board):

    if terminal(board):

        return utility(board), None

    v = float('-inf')

    move = None

    for action in actions(board):

        # v = max(v, min\_value(result(board, action)))

        aux, act = min\_value(result(board, action))

        if aux > v:

            v = aux

            move = action

            if v == 1:

                return v, move

    return v, move

def min\_value(board):

    if terminal(board):

        return utility(board), None

    v = float('inf')

    move = None

    for action in actions(board):

        # v = max(v, min\_value(result(board, action)))

        aux, act = max\_value(result(board, action))

        if aux < v:

            v = aux

            move = action

            if v == -1:

                return v, move

    return v, move

# **CHƯƠNG 3: GIAO DIỆN**

## **Giao diện chính**



A picture containing graphical user interface

Description automatically generated

## **Kết quả**

A picture containing shape

Description automatically generated

# **Tài liệu tham khảo**

[1]. Slide bài giảng trí tuệ nhân tạo năm 2022 – trường Đại học Công nghiệp Thực phẩm Thành phố Hồ Chí Minh.

[2]. <https://doc.edu.vn/tai-lieu/chuyen-de-thuat-toan-minimax-7459/>

[3]. https://www.iostream.vn/giai-thuat-lap-trinh/giai-thuat-tim-kiem-minimax-s1EVnH