```
import copy
import math
import numpy as np

# Các hằng số đại diện
X = "X"
O = "O"
EMPTY = None

def initial_state(n=3):
    """
    Tạo bàn cờ kích thước n x n dùng vòng lặp for lồng nhau.
    """
    board = [] # Khởi tạo danh sách bàn cờ rỗng
    k = 1;
    for i in range(n): # Duyệt qua từng hàng (i)
        row = [] # Tạo một hàng rỗng mới
        for j in range(n): # Duyệt qua từng cột (j)
            row.append(EMPTY) # Thêm ô trống vào hàng hiện tại
        board.append(row) # Thêm hàng đã hoàn thiện vào bàn cờ

    return board


def player(board):
    """
    Hàm xác định lượt chơi hiện tại là của ai (X hay O).
    Logic: Đếm số lượng X và O đã đánh trên bàn cờ.
    """
    count_x = 0
    count_o = 0
    for i in board:
        for j in i:
            if j == X:
                count_x += 1
            elif j == O:
                count_o += 1

    # Nếu số lượng bằng nhau thì X đi (X luôn đi trước), ngược lại O đi
    if count_x > count_o:
        return O
    else:
        return X

def actions(board):
    """
    Hàm trả về tập hợp các vị trí có thể đánh tiếp theo (các ô còn trống).
    """
    res = set() # Tập hợp chứa các toạ độ (hàng, cột)
    board_len = len(board) # Kích thước bàn cờ (3)
    for i in range(board_len):
        for j in range(board_len):
            if board[i][j] == EMPTY:
                res.add((i, j)) # Thêm toạ độ ô trống vào danh sách
    return res

def result(board, action):
    """
    Hàm trả về trạng thái bàn cờ mới sau khi thực hiện một nước đi.
    QUAN TRỌNG: Phải dùng deepcopy để không làm thay đổi bàn cờ gốc khi máy tính đang "thử" nước đi.
    """
    curr_player = player(board)
    re = copy.deepcopy(board) # Tạo bản sao độc lập của bàn cờ
    (i, j) = action
    re[i][j] = curr_player    # Gán nước đi vào bản sao
    return re

def get_horizontal_winner(board):
    """
    Kiểm tra chiến thắng theo hàng ngang.
    """
    board_len = len(board)
    for i in range(board_len):
        win = board[i][0] # Lấy giá trị ô đầu tiên của hàng
        if win is None: # Nếu ô đầu trống thì hàng này không thể thắng -> bỏ qua
```

```
            continue
        match = True
        for j in range(board_len):
            if board[i][j] != win: # Nếu có 1 ô khác màu thì không thắng
                match = False
                break
        if match: # Nếu duyệt hết hàng mà vẫn khớp -> Trả về người thắng
            return win
    return None

def get_vertical_winner(board):
    """
    Kiểm tra chiến thắng theo hàng dọc (cột).
    """
    board_len = len(board)
    for i in range(board_len):
        winner_val = board[0][i] # Lấy giá trị ô đầu tiên của cột
        if winner_val is None:
            continue
        match = True
        for j in range(board_len):
            if board[j][i] != winner_val: # Kiểm tra các ô trong cột dọc
                match = False
                break
        if match:
            return winner_val
    return None

def get_diagonal_winner(board):
    """
    Kiểm tra chiến thắng theo 2 đường chéo.
    (Đã sửa lỗi logic ở đoạn này để code chạy được)
    """
    board_len = len(board)

    # 1. Đường chéo chính (từ góc trên trái xuống dưới phải: [0][0], [1][1], [2][2])
    win = board[0][0]
    if win is not None:
        match = True
        for i in range(board_len):
            if board[i][i] != win: # So sánh board[i][i]
                match = False
                break
        if match:
            return win

    # 2. Đường chéo phụ (từ góc trên phải xuống dưới trái: [0][2], [1][1], [2][0])
    win = board[0][board_len - 1]
    if win is not None:
        match = True
        for i in range(board_len):
            # Công thức đường chéo phụ: cột = độ dài - 1 - hàng
            if board[i][board_len - 1 - i] != win:
                match = False
                break
        if match:
            return win

    return None

def winner(board):
    """
    Hàm tổng hợp kiểm tra người chiến thắng (Ngang, Dọc, Chéo).
    """
    return (get_horizontal_winner(board) or
            get_vertical_winner(board) or
            get_diagonal_winner(board) or None)

def terminal(board):
    """
    Hàm kiểm tra game đã kết thúc chưa (Có người thắng hoặc Hòa).
    """
    if winner(board) is not None:
        return True

    # Nếu chưa ai thắng, kiểm tra xem còn ô trống không
    for row in board:
```

```
            if EMPTY in row:
                return False # Còn ô trống -> Chưa kết thúc

        return True # Không còn ô trống và không ai thắng -> Hòa

    def utility(board):
        """
        Hàm trả về điểm số của trạng thái kết thúc (Dùng cho Minimax).
        X thắng: +1
        O thắng: -1
        Hòa: 0
        """
        val = winner(board)
        if val == X:
            return 1
        elif val == O:
            return -1
        return 0

    def maxValue(state):
        """
        Đại diện cho lượt của X (người muốn Maximize điểm số).
        Mục tiêu: Tìm nước đi có điểm cao nhất (gần 1 nhất).
        """
        if terminal(state):  # Nếu game kết thúc, trả về điểm số
            return utility(state)

        v = -math.inf  # Khởi tạo giá trị ban đầu là âm vô cùng

        # Duyệt qua tất cả các nước đi có thể
        for action in actions(state):
            # Gọi minValue vì sau lượt này sẽ đến lượt O (O muốn Min)
            v = max(v, minValue(result(state, action)))

        return v

    def minValue(state):
        """
        Đại diện cho lượt của O (người muốn Minimize điểm số).
        Mục tiêu: Tìm nước đi có điểm thấp nhất (gần -1 nhất).
        """
        if terminal(state):
            return utility(state)

        v = math.inf # Khởi tạo giá trị ban đầu là dương vô cùng

        # Duyệt qua tất cả các nước đi có thể
        for action in actions(state):
            # Gọi maxValue vì sau lượt này sẽ đến lượt X (X muốn Max)
            v = min(v, maxValue(result(state, action)))

        return v

    def minimax(board):
        """
        Hàm chính của AI để quyết định nước đi tốt nhất.
        """
        current_player = player(board)
        best_move = None

        if current_player == X: # Nếu AI là X (Muốn Max điểm)
            best_score = -math.inf
            for action in actions(board):
                # Giả lập đi thử, sau đó xem kết quả trả về từ minValue
                val = minValue(result(board, action))
                if val > best_score:
                    best_score = val
                    best_move = action
        else: # Nếu AI là O (Muốn Min điểm)
            best_score = math.inf
            for action in actions(board):
                # Giả lập đi thử, sau đó xem kết quả trả về từ maxValue
                val = maxValue(result(board, action))
                if val < best_score:
                    best_score = val
                    best_move = action
```

```
    return best_move

# --- KHỐI CHẠY CHƯƠNG TRÌNH CHÍNH (MAIN) ---
if __name__ == "__main__":
    print("--- TRÒ CHƠI TIC TAC TOE (AI MINIMAX) ---")

    # Thêm đoạn này để chọn kích thước
    while True:
        try:
            sz = int(input("Nhập kích thước bàn cờ (3, 4, 5...): "))
            if sz >= 3:
                break
            print("Kích thước phải từ 3 trở lên.")
        except ValueError:
            print("Vui lòng nhập số.")

    board = initial_state(sz) # Truyền kích thước vào đây

    print("Chọn người chơi (X đi trước, O đi sau)")

    # Vòng lặp chọn phe
    while True:
        user_choice = input("Bạn muốn chơi X hay O? ").upper()
        if user_choice in ["X", "O"]:
            user = user_choice
            break
        print("Lựa chọn không hợp lệ. Vui lòng nhập X hoặc O.")

    # Gán phe cho AI
    if user == "X":
        ai = "O"
    else:
        ai = "X"

    # Vòng lặp game
    while True:
        game_over = terminal(board)
        current_turn = player(board) # Lấy người chơi hiện tại

        # In bàn cờ hiện tại dùng numpy cho đẹp
        print("\nBàn cờ hiện tại:")
        print(np.array(board))

        if game_over:
            win = winner(board)
            if win is None:
                print("Kết quả: HÒA.")
            else:
                print(f"Kết quả: {win} THẮNG.")
            break

        # --- Lượt của AI ---
        if current_turn == ai:
            print(f"AI ({ai}) đang suy nghĩ...")
            move = minimax(board)
            board = result(board, move)

        # --- Lượt của Người chơi ---
        else:
            print(f"Lượt của BẠN ({user})")
            while True:
                try:
                    print("Nhập toạ độ đánh (Hàng [0-2], Cột [0-2]):")
                    i = int(input("Hàng: "))
                    j = int(input("Cột:  "))

                    # Kiểm tra nước đi có hợp lệ không
                    if (i, j) in actions(board):
                        board = result(board, (i, j))
                        break
                    else:
                        print("Ô này đã có người đánh hoặc toạ độ sai. Thử lại.")
                except ValueError:
                    print("Vui lòng chỉ nhập số.")
```

```
--- TRÒ CHƠI TIC TAC TOE (AI MINIMAX) ---
Nhập kích thước bàn cờ (3, 4, 5...): 3
```

```
Chọn người chơi (X đi trước, O đi sau)
Bạn muốn chơi X hay O? x

Bàn cờ hiện tại:
[[None None None]
 [None None None]
 [None None None]]
Lượt của BẠN (X)
Nhập toạ độ đánh (Hàng [0-2], Cột [0-2]):
Hàng: 0
Cột:  2

Bàn cờ hiện tại:
[[None None 'X']
 [None None None]
 [None None None]]
AI (O) đang suy nghĩ...

Bàn cờ hiện tại:
[[None None 'X']
 [None 'O' None]
 [None None None]]
Lượt của BẠN (X)
Nhập toạ độ đánh (Hàng [0-2], Cột [0-2]):
Hàng: 1
Cột:  2

Bàn cờ hiện tại:
[[None None 'X']
 [None 'O' 'X']
 [None None None]]
AI (O) đang suy nghĩ...

Bàn cờ hiện tại:
[[None None 'X']
 [None 'O' 'X']
 [None None 'O']]
Lượt của BẠN (X)
Nhập toạ độ đánh (Hàng [0-2], Cột [0-2]):
Hàng: 0
Cột:  1

Bàn cờ hiện tại:
[[None 'X' 'X']
 [None 'O' 'X']
 [None None 'O']]
AI (O) đang suy nghĩ...

Bàn cờ hiện tại:
[['O' 'X' 'X']
 [None 'O' 'X']
 [None None 'O']]
Kết quả: O THẮNG.
```

```
import copy
import math

# Các hằng số đại diện
X = "X"
O = "O"
EMPTY = None

def initial_state(n=3):
    """
    Tạo bàn cờ kích thước n x n
    """
    board = []
    for i in range(n):
        row = []
        for j in range(n):
            row.append(EMPTY)
        board.append(row)
    return board

def print_board_formatted(board):
    """
    Hàm in bàn cờ đẹp, thay thế None bằng số thứ tự (1, 2, 3...)
    để người chơi dễ chọn ô.
    """
    n = len(board)
    line = "-" * (n * 6 + 1) # Tạo đường kẻ ngang
    print(line)
```

```
        print(line)
        for i in range(n):
            row_str = "| "
            for j in range(n):
                cell_val = board[i][j]
                if cell_val is None:
                    # Công thức tính số thứ tự: (hàng * độ_rộng) + cột + 1
                    display_val = str(i * n + j + 1)
                else:
                    display_val = cell_val

                # Căn chỉnh in cho đẹp (đệm thêm khoảng trắng)
                row_str += f"{display_val:^3} | "
            print(row_str)
            print(line)

    def player(board):
        """
        Xác định lượt chơi hiện tại (X luôn đi trước nếu bàn cờ trống)
        """
        count_x = sum(row.count(X) for row in board)
        count_o = sum(row.count(O) for row in board)

        if count_x > count_o:
            return O
        else:
            return X

    def actions(board):
        """
        Trả về tập hợp các vị trí ô trống (hàng, cột)
        """
        res = set()
        board_len = len(board)
        for i in range(board_len):
            for j in range(board_len):
                if board[i][j] == EMPTY:
                    res.add((i, j))
        return res

    def result(board, action):
        """
        Trả về trạng thái bàn cờ mới sau khi đi 1 nước
        """
        curr_player = player(board)
        re = copy.deepcopy(board)
        (i, j) = action
        re[i][j] = curr_player
        return re

    def get_horizontal_winner(board):
        """Kiểm tra thắng theo hàng ngang"""
        board_len = len(board)
        for i in range(board_len):
            win = board[i][0]
            if win is None:
                continue
            match = True
            for j in range(board_len):
                if board[i][j] != win:
                    match = False
                    break
            if match:
                return win
        return None

    def get_vertical_winner(board):
        """Kiểm tra thắng theo hàng dọc"""
        board_len = len(board)
        for i in range(board_len):
            winner_val = board[0][i]
            if winner_val is None:
                continue
            match = True
            for j in range(board_len):
                if board[j][i] != winner_val:
                    match = False
                    break
```

```
            if match:
                return winner_val
        return None

    def get_diagonal_winner(board):
        """Kiểm tra thắng theo 2 đường chéo"""
        board_len = len(board)

        # 1. Đường chéo chính
        win = board[0][0]
        if win is not None:
            match = True
            for i in range(board_len):
                if board[i][i] != win:
                    match = False
                    break
            if match:
                return win

        # 2. Đường chéo phụ
        win = board[0][board_len - 1]
        if win is not None:
            match = True
            for i in range(board_len):
                if board[i][board_len - 1 - i] != win:
                    match = False
                    break
            if match:
                return win

        return None

    def winner(board):
        """Tổng hợp kiểm tra người thắng"""
        return (get_horizontal_winner(board) or
                get_vertical_winner(board) or
                get_diagonal_winner(board) or None)

    def terminal(board):
        """Kiểm tra game kết thúc chưa"""
        if winner(board) is not None:
            return True

        for row in board:
            if EMPTY in row:
                return False
        return True

    def utility(board):
        """Điểm số: X thắng +1, O thắng -1, Hoà 0"""
        val = winner(board)
        if val == X:
            return 1
        elif val == O:
            return -1
        return 0

    def maxValue(state):
        """Lượt của X (Max)"""
        if terminal(state):
            return utility(state)

        v = -math.inf
        for action in actions(state):
            v = max(v, minValue(result(state, action)))
        return v

    def minValue(state):
        """Lượt của O (Min)"""
        if terminal(state):
            return utility(state)

        v = math.inf
        for action in actions(state):
            v = min(v, maxValue(result(state, action)))
        return v
```

```
    def minimax(board):
        """Hàm AI quyết định nước đi"""
        current_player = player(board)
        best_move = None

        if current_player == X:
            best_score = -math.inf
            for action in actions(board):
                val = minValue(result(board, action))
                if val > best_score:
                    best_score = val
                    best_move = action
        else:
            best_score = math.inf
            for action in actions(board):
                val = maxValue(result(board, action))
                if val < best_score:
                    best_score = val
                    best_move = action

        return best_move

    # --- KHỐI CHẠY CHƯƠNG TRÌNH CHÍNH (MAIN) ---
    if __name__ == "__main__":
        print("--- TRÒ CHƠI TIC TAC TOE (AI MINIMAX) ---")

        while True:
            try:
                sz = int(input("Nhập kích thước bàn cờ (3, 4, 5...): "))
                if sz >= 3:
                    break
                print("Kích thước phải từ 3 trở lên.")
            except ValueError:
                print("Vui lòng nhập số.")

        # Lưu ý: Với size >= 4, AI Minimax thuần (không cắt tỉa) sẽ chạy khá chậm.
        if sz > 3:
            print("LƯU Ý: Với bàn cờ lớn hơn 3x3, AI sẽ suy nghĩ lâu hơn.")

        board = initial_state(sz)

        print("Chọn người chơi (X đi trước, O đi sau)")
        while True:
            user_choice = input("Bạn muốn chơi X hay O? ").upper()
            if user_choice in ["X", "O"]:
                user = user_choice
                break
            print("Lựa chọn không hợp lệ. Vui lòng nhập X hoặc O.")

        # Gán phe cho AI
        if user == "X":
            ai = "O"
        else:
            ai = "X"

        # Vòng lặp game
        while True:
            game_over = terminal(board)
            current_turn = player(board)

            # In bàn cờ dạng số
            print("\nBàn cờ hiện tại:")
            print_board_formatted(board)

            if game_over:
                win = winner(board)
                if win is None:
                    print(">>> KẾT QUẢ: HÒA <<<")
                else:
                    print(f">>> KẾT QUẢ: {win} THẮNG <<<")
                break

            # --- Lượt của AI ---
            if current_turn == ai:
                print(f"AI ({ai}) đang suy nghĩ...")
                move = minimax(board)
                board = result(board, move)
```

```
        # --- Lượt của Người chơi ---
        else:
            print(f"Lượt của BẠN ({user})")
            n = len(board)

            while True:
                try:
                    # Nhập số thay vì toạ độ
                    pos = int(input(f"Nhập vị trí ô muốn đánh (1-{n*n}): "))

                    # Chuyển đổi số thành (hàng, cột)
                    i = (pos - 1) // n
                    j = (pos - 1) % n

                    if (i, j) in actions(board):
                        board = result(board, (i, j))
                        break
                    else:
                        print("Ô này không hợp lệ hoặc đã có người đánh. Thử lại.")

                except ValueError:
                    print("Vui lòng chỉ nhập số nguyên.")
```

```
-------------------
|  4  |  5  |  6  |
-------------------
|  7  |  8  |  9  |
-------------------
AI (O) đang suy nghĩ...

Bàn cờ hiện tại:
-------------------
|  X  |  2  |  3  |
-------------------
|  4  |  O  |  6  |
-------------------
|  7  |  8  |  9  |
-------------------
Lượt của BẠN (X)
Nhập vị trí ô muốn đánh (1-9): 4

Bàn cờ hiện tại:
-------------------
|  X  |  2  |  3  |
-------------------
|  X  |  O  |  6  |
-------------------
|  7  |  8  |  9  |
-------------------
AI (O) đang suy nghĩ...

Bàn cờ hiện tại:
-------------------
|  X  |  2  |  3  |
-------------------
|  X  |  O  |  6  |
-------------------
|  O  |  8  |  9  |
-------------------
Lượt của BẠN (X)
Nhập vị trí ô muốn đánh (1-9): 2

Bàn cờ hiện tại:
-------------------
|  X  |  X  |  3  |
-------------------
|  X  |  O  |  6  |
-------------------
|  O  |  8  |  9  |
-------------------
AI (O) đang suy nghĩ...

Bàn cờ hiện tại:
-------------------
|  X  |  X  |  O  |
-------------------
|  X  |  O  |  6  |
-------------------
|  O  |  8  |  9  |
-------------------
>>> KẾT QUẢ: O THẮNG <<<
```

```
import os
import math

# Hàm kiểm tra người chiến thắng
def GetWinner(board):
    """
    Trả về người thắng ('X' hoặc 'O') nếu có, ngược lại trả về None.
    """
    # Kiểm tra hàng ngang
    if board[0] == board[1] and board[1] == board[2]:
        return board[0]
    elif board[3] == board[4] and board[4] == board[5]:
        return board[3]
    elif board[6] == board[7] and board[7] == board[8]:
        return board[6]
    # Kiểm tra hàng dọc
    elif board[0] == board[3] and board[3] == board[6]:
        return board[0]
    elif board[1] == board[4] and board[4] == board[7]:
        return board[1]
    elif board[2] == board[5] and board[5] == board[8]:
        return board[2]
    # Kiểm tra đường chéo
    elif board[0] == board[4] and board[4] == board[8]:
        return board[0]
    elif board[2] == board[4] and board[4] == board[6]:
        return board[2]

    return None

# Hàm in bàn cờ ra màn hình
def PrintBoard(board):
    """
    Xóa màn hình console và in trạng thái bàn cờ hiện tại.
    """
    os.system('cls' if os.name == 'nt' else 'clear')
    print(f'''
    {board[0]} | {board[1]} | {board[2]}
    ---------
    {board[3]} | {board[4]} | {board[5]}
    ---------
    {board[6]} | {board[7]} | {board[8]}
    ''')

# Hàm lấy danh sách các ô còn trống
def GetAvailableCells(board):
    """
    Trả về danh sách các chỉ số (index) của các ô chưa được đánh.
    """
    available = list()
    for cell in board:
        # Nếu ô đó không phải X cũng không phải O thì là ô trống (chứa số)
        if cell != "X" and cell != "O":
            available.append(cell)
    return available

# Thuật toán Minimax với Cắt tỉa Alpha-Beta
def minimax(position, depth, alpha, beta, isMaximizing):
    """
    Thuật toán AI cốt lõi để chọn nước đi tốt nhất.
    Trả về điểm số đánh giá của nước đi.
    """
    winner = GetWinner(position)

    # Đánh giá bàn cờ:
    # - Nếu X thắng: trả về 10 - độ sâu (thắng càng nhanh càng tốt)
    # - Nếu O thắng: trả về -10 + độ sâu (thua càng chậm càng tốt)
    # - Nếu hòa: trả về 0
    if winner != None:
        return 10 - depth if winner == "X" else -10 + depth

    if len(GetAvailableCells(position)) == 0:
        return 0

    if isMaximizing: # Lượt của người muốn Tối đa hóa điểm (thường là X)
        maxEval = -math.inf
        for cell in GetAvailableCells(position):
            # Thử đi nước này
```

```
                # Thử đi nước này
                position[cell - 1] = "X"
                # Gọi đệ quy
                eval_score = minimax(position, depth + 1, alpha, beta, False)
                # Hoàn tác nước đi (backtrack)
                position[cell - 1] = cell

                maxEval = max(maxEval, eval_score)
                alpha = max(alpha, eval_score)

                # Cắt tỉa Alpha-Beta
                if beta <= alpha:
                    break
            return maxEval
        else: # Lượt của người muốn Tối thiểu hóa điểm (thường là O)
            minEval = +math.inf
            for cell in GetAvailableCells(position):
                # Thử đi nước này
                position[cell - 1] = "O"
                # Gọi đệ quy
                eval_score = minimax(position, depth + 1, alpha, beta, True)
                # Hoàn tác nước đi (backtrack)
                position[cell - 1] = cell

                minEval = min(minEval, eval_score)
                beta = min(beta, eval_score)

                # Cắt tỉa Alpha-Beta
                if beta <= alpha:
                    break
            return minEval

    # Hàm tìm nước đi tốt nhất cho máy (AI)
    def FindBestMove(currentPosition, AI):
        """
        Duyệt qua tất cả các ô trống, chạy hàm minimax để đánh giá.
        Trả về ô có giá trị tối ưu nhất.
        """
        bestVal = -math.inf if AI == "X" else +math.inf
        bestMove = -1

        for cell in GetAvailableCells(currentPosition):
            # Thử đi nước này
            currentPosition[cell - 1] = AI

            # Đánh giá nước đi bằng Minimax
            # Nếu AI là X -> lượt tiếp theo là O (isMaximizing = False)
            # Nếu AI là O -> lượt tiếp theo là X (isMaximizing = True)
            moveVal = minimax(currentPosition, 0, -math.inf, +math.inf, False if AI == "X" else True)

            # Hoàn tác
            currentPosition[cell - 1] = cell

            # Cập nhật nước đi tốt nhất
            if (AI == "X" and moveVal > bestVal):
                bestMove = cell
                bestVal = moveVal
            elif (AI == "O" and moveVal < bestVal):
                bestMove = cell
                bestVal = moveVal

        return bestMove

    def main():
        while True:
            player = input("Bạn muốn chơi quân nào (nhập X hoặc O): ").strip().upper()
            if player == 'X' or player == 'O':
                break
            print("Vui lòng chỉ nhập X hoặc O.")

        AI = "O" if player == "X" else "X"
        # Khởi tạo bàn cờ với các số từ 1 đến 9
        currentGame = [*range(1, 10)]

        # X luôn đi trước
        currentTurn = "X"
        counter = 0 # Đếm số nước đi để kiểm tra hòa
```

```
        while True:
            # Nếu đến lượt AI
            if currentTurn == AI:
                print(f"Máy ({AI}) đang suy nghĩ...")
                # LƯU Ý: Nếu AI đi trước (là X), nước đi đầu tiên tính toán khá lâu.
                # Để nhanh hơn, có thể hardcode nước đầu tiên vào giữa (số 5).
                cell = FindBestMove(currentGame, AI)
                currentGame[cell - 1] = AI
                currentTurn = player

            # Nếu đến lượt Người chơi
            elif currentTurn == player:
                PrintBoard(currentGame)
                while True:
                    try:
                        humanInput = int(input(f"Nhập số ô bạn muốn đi (1-9) cho quân {player}: ").strip())
                        if humanInput in currentGame:
                            currentGame[humanInput - 1] = player
                            currentTurn = AI
                            break
                        else:
                            print("Ô này không hợp lệ hoặc đã có người đi. Thử lại.")
                    except ValueError:
                        print("Vui lòng nhập một số nguyên từ 1 đến 9.")

            counter += 1

            # Kiểm tra thắng thua sau mỗi nước đi
            winner = GetWinner(currentGame)
            if winner != None:
                PrintBoard(currentGame)
                print(f"CHÚC MỪNG! {winner} ĐÃ CHIẾN THẮNG!!!")
                break

            # Kiểm tra hòa (khi bàn cờ đầy và không ai thắng)
            if winner == None and len(GetAvailableCells(currentGame)) == 0:
                PrintBoard(currentGame)
                print("TRẬN ĐẤU HÒA.")
                break

    if __name__ == "__main__":
        main()
```

```
Bạn muốn chơi quân nào (nhập X hoặc O): x

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

Nhập số ô bạn muốn đi (1-9) cho quân X: 1
Máy (O) đang suy nghĩ...

    X | 2 | 3
    ---------
    4 | O | 6
    ---------
    7 | 8 | 9

Nhập số ô bạn muốn đi (1-9) cho quân X: 2
Máy (O) đang suy nghĩ...

    X | X | O
    ---------
    4 | O | 6
    ---------
    7 | 8 | 9

Nhập số ô bạn muốn đi (1-9) cho quân X: 4
Máy (O) đang suy nghĩ...

    X | X | O
    ---------
    X | O | 6
    ---------
    O | 8 | 9

CHÚC MỪNG! O ĐÃ CHIẾN THẮNG!!!
```

```
import os
import math

# Biến toàn cục để giới hạn độ sâu suy nghĩ cho bàn cờ lớn
MAX_DEPTH_LIMIT = 0

# Hàm kiểm tra người chiến thắng (Động - áp dụng cho mọi kích thước n)
def GetWinner(board, n):
    # Kiểm tra hàng ngang
    for row in range(n):
        start = row * n
        if all(board[start + i] == board[start] for i in range(1, n)):
            return board[start]

    # Kiểm tra hàng dọc
    for col in range(n):
        if all(board[col + i * n] == board[col] for i in range(1, n)):
            return board[col]

    # Kiểm tra đường chéo chính (Top-left -> Bottom-right)
    if all(board[i * (n + 1)] == board[0] for i in range(1, n)):
        return board[0]

    # Kiểm tra đường chéo phụ (Top-right -> Bottom-left)
    # Công thức index: (i + 1) * (n - 1)
    if all(board[(i + 1) * (n - 1)] == board[n - 1] for i in range(1, n)):
        return board[n - 1]

    return None

# Hàm in bàn cờ động
def PrintBoard(board, n):
    os.system('cls' if os.name == 'nt' else 'clear')
    print("\n")
    # Tạo đường kẻ ngang (ví dụ: ---------)
    horizontal_line = " " + "-" * (n * 4 - 1)

    for row in range(n):
        # Lấy các phần tử của hàng hiện tại
        row_start = row * n
        row_cells = board[row_start : row_start + n]

        # Format in ra: 1 | 2 | 3
        # Căn lề số cho đẹp (dùng :2s để số có 2 chữ số không bị lệch)
        print(" " + " | ".join(f"{str(cell):2s}" for cell in row_cells))

        if row < n - 1:
            print(horizontal_line)
    print("\n")

# Hàm lấy danh sách các ô còn trống
def GetAvailableCells(board):
    available = []
    for cell in board:
        # Nếu không phải chuỗi (X/O) mà là số (int) thì là ô trống
        if isinstance(cell, int):
            available.append(cell)
        # Fallback nếu bạn khởi tạo board bằng string số
        elif cell != "X" and cell != "O":
             available.append(cell)
    return available

# Thuật toán Minimax với Cắt tỉa Alpha-Beta và Giới hạn độ sâu
def minimax(position, depth, alpha, beta, isMaximizing, n):
    winner = GetWinner(position, n)

    # 1. Điều kiện dừng: Có người thắng
    if winner == "X": return 100 - depth
    if winner == "O": return -100 + depth

    # 2. Điều kiện dừng: Hòa
    avail_cells = GetAvailableCells(position)
    if len(avail_cells) == 0: return 0

    # 3. Điều kiện dừng: Quá độ sâu (QUAN TRỌNG CHO 4x4)
    if depth >= MAX_DEPTH_LIMIT:
```

```
            return 0 # Trả về hòa giả định (hoặc cần hàm heuristic đánh giá thế cờ tốt hơn)

        if isMaximizing: # Lượt X
            maxEval = -math.inf
            for cell in avail_cells:
                position[cell - 1] = "X"
                eval_score = minimax(position, depth + 1, alpha, beta, False, n)
                position[cell - 1] = cell # Backtrack

                maxEval = max(maxEval, eval_score)
                alpha = max(alpha, eval_score)
                if beta <= alpha: break
            return maxEval
        else: # Lượt O
            minEval = +math.inf
            for cell in avail_cells:
                position[cell - 1] = "O"
                eval_score = minimax(position, depth + 1, alpha, beta, True, n)
                position[cell - 1] = cell # Backtrack

                minEval = min(minEval, eval_score)
                beta = min(beta, eval_score)
                if beta <= alpha: break
            return minEval

    # Hàm tìm nước đi tốt nhất
    def FindBestMove(currentPosition, AI, n):
        bestVal = -math.inf if AI == "X" else +math.inf
        bestMove = -1

        avail_cells = GetAvailableCells(currentPosition)

        # Nếu là bàn cờ 4x4 và là nước đi đầu tiên, random để đỡ phải tính lâu
        if n >= 4 and len(avail_cells) == n*n:
            return (n*n) // 2 # Đánh vào giữa

        for cell in avail_cells:
            currentPosition[cell - 1] = AI

            # Gọi minimax
            isMax = False if AI == "X" else True
            moveVal = minimax(currentPosition, 0, -math.inf, +math.inf, isMax, n)

            currentPosition[cell - 1] = cell # Backtrack

            if (AI == "X" and moveVal > bestVal):
                bestMove = cell
                bestVal = moveVal
            elif (AI == "O" and moveVal < bestVal):
                bestMove = cell
                bestVal = moveVal

        return bestMove

    def main():
        global MAX_DEPTH_LIMIT

        # Nhập kích thước bàn cờ
        while True:
            try:
                n = int(input("Nhập kích thước bàn cờ (ví dụ 3 cho 3x3, 4 cho 4x4): "))
                if n >= 3:
                    break
                print("Kích thước tối thiểu là 3.")
            except ValueError:
                print("Vui lòng nhập số nguyên.")

        # Cấu hình độ sâu dựa trên kích thước
        if n == 3:
            MAX_DEPTH_LIMIT = 9 # Duyệt hết
        elif n == 4:
            MAX_DEPTH_LIMIT = 5 # Duyệt 5 nước thôi (nếu cao hơn sẽ rất chậm)
        else:
            MAX_DEPTH_LIMIT = 4 # Bàn càng to càng phải giảm độ sâu

        while True:
            player = input("Bạn muốn chơi quân nào (X/O): ").strip().upper()
            if player in ['X', 'O']: break
```

```
    AI = "O" if player == "X" else "X"
    currentGame = [*range(1, n*n + 1)] # Tạo bàn cờ từ 1 đến n*n
    currentTurn = "X"

    while True:
        winner = GetWinner(currentGame, n)
        if winner:
            PrintBoard(currentGame, n)
            print(f"CHÚC MỪNG! {winner} ĐÃ CHIẾN THẮNG!!!")
            break

        if len(GetAvailableCells(currentGame)) == 0:
            PrintBoard(currentGame, n)
            print("TRẬN ĐẤU HÒA.")
            break

        if currentTurn == AI:
            print(f"Máy ({AI}) đang suy nghĩ (Độ sâu: {MAX_DEPTH_LIMIT})...")
            cell = FindBestMove(currentGame, AI, n)
            currentGame[cell - 1] = AI
            currentTurn = player
        else:
            PrintBoard(currentGame, n)
            while True:
                try:
                    humanInput = int(input(f"Nhập ô (1-{n*n}) cho {player}: "))
                    if humanInput in GetAvailableCells(currentGame):
                        currentGame[humanInput - 1] = player
                        currentTurn = AI
                        break
                    else:
                        print("Ô không hợp lệ.")
                except ValueError:
                    print(f"Nhập số từ 1 đến {n*n}.")

if __name__ == "__main__":
    main()
```

```
 7  | 8  | 9


Nhập ô (1-9) cho X: 1
Máy (O) đang suy nghĩ (Độ sâu: 9)...


 X  | 2  | 3
 -----------
 4  | O  | 6
 -----------
 7  | 8  | 9


Nhập ô (1-9) cho X: 3
Máy (O) đang suy nghĩ (Độ sâu: 9)...


 X  | O  | X
 -----------
 4  | O  | 6
 -----------
 7  | 8  | 9


Nhập ô (1-9) cho X: 8
Máy (O) đang suy nghĩ (Độ sâu: 9)...


 X  | O  | X
 -----------
 O  | O  | 6
 -----------
```

```
 -----------
 O  | O  | X
 -----------
 7  | X  | O


Nhập ô (1-9) cho X: 7


 X  | O  | X
 -----------
 O  | O  | X
 -----------
 X  | X  | O


TRẬN ĐẤU HÒA
```