# 1

Nội dung các hàm trong mã nguồn

|  |  |
| --- | --- |
| Tên hàm | Chức năng |
| main | Gồm 3 phần chính:  Xử lý nhãn X hoặc O cho player và AI  Xử lý lặp của game  Hiển thị kết quả game |
| human\_turn | Hàm xử lý từ người chơi.  Nếu không còn ô trống trên bỏad hoặc game\_over thì thoát. Ngược lại, kiểm tra xem player có nhập đúng tọa độ từ phím để đánh hay không. Tọa độ hợp lệ thì đánh dấu X or O vào board |
| game\_over | Kiểm tra xem Player or AI đã ai thắng game hay chưa. Hàm con wins sẽ kiểm tra theo hàng, cột, và đường chéo có đủ 3 dấu X hoặc O chưa, nếu đủ thì đã win |
| ai\_turn | Hàm xử lý nước đi của AI.   * Nếu không có ô trống hoặc game\_over thì thoát * Nếu AI đi đầu tiên thì random tọa độ x,y để đánh. Ngược lại gọi hàm minimax để tìm ra x,y * Đánh dấu O/X của AI vào board tại độ độ x,y |
| minimax | Hàm AI để tìm ra lựa chọn tốt nhất – tìm ra tọa độ x,y  Đây là hàm đệ quy sử dung thuật toán minimax value tối đa hóa lợi ích nếu player là AI, ngược lại tối thiểu hóa nếu player là người |

# 2. Cải thiện thuật toán

Thêm 3 biến để điều chỉnh luật chơi và chiều sâu của thuật toán

# kích thước bảng tic-tac-toe

BOARD\_SIZE = 10

HUMAN = -1

COMP = +1

# tùy chỉnh luật chơi, 3,4 or 5 dấu liên tục

WIN\_THRESHOLD = 3

# chiều sâu của thuật toán minimax = min(DEPTH, current number of empty cells)

DEPTH = 10

# board = [

#     [0, 0, 0],

#     [0, 0, 0],

#     [0, 0, 0],

# ]

board = []

# khởi tạo mảng 2D với giá trị 0, kích thước BOARD\_SIZE

for i in range(0, BOARD\_SIZE):

    board\_row = []

    for j in range(0, BOARD\_SIZE):

        board\_row.append(0)

    board.append(board\_row)

Hàm đếm các Player liên tục theo 1 hướng xác định [delta\_i,delta\_j]

def count\_maker(state, player, current\_cell, delta\_i, delta\_j):

    next\_i = current\_cell[0] + delta\_i

    next\_j = current\_cell[1] + delta\_j

    if next\_i >= 0 and next\_i < BOARD\_SIZE and next\_j >= 0 and next\_j < BOARD\_SIZE:

        if state[next\_i][next\_j] == player:

            return 1 + count\_maker(state, player, [next\_i, next\_j], delta\_i, delta\_j)

    return 0

def count\_makers(state, player, current\_cell, delta\_i, delta\_j):

    x, y = current\_cell[0], current\_cell[1]

    count = 0

    if state[x][y] == player:

        count = 1 + count\_maker(state, player, current\_cell, delta\_i, delta\_j)

    return count

Cập nhật hàm wins để đáp ứng luật chơi từ cấu hình. Duyệt lần lượt 8 hướng từ 1 ô cho trước

def wins(state, player):

    """

    This function tests if a specific player wins. Possibilities:

    \* Three rows    [X X X] or [O O O]

    \* Three cols    [X X X] or [O O O]

    \* Two diagonals [X X X] or [O O O]

    :param state: the state of the current board

    :param player: a human or a computer

    :return: True if the player wins

    """

    for row\_index in range(0, BOARD\_SIZE):

        for col\_index in range(0, BOARD\_SIZE):

            # đi về phía trên

            total\_count = count\_makers(

                state, player, [row\_index, col\_index], -1, 0)

            if total\_count == WIN\_THRESHOLD:

                return True

            # đi về phía trên-phải

            total\_count = count\_makers(

                state, player, [row\_index, col\_index], -1, +1)

            if total\_count == WIN\_THRESHOLD:

                return True

Loại bỏ các cells trống mà hàng xóm củng trống trước khi chạy thuật toán

def ai\_enhance\_empty\_cells(state):

    arr\_empty\_cells = empty\_cells(state)

    best\_cells = []

    for item in arr\_empty\_cells:

        if check\_all\_boundary\_empty(state, item):

            best\_cells.append(item)

    return best\_cells