TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

=======***=======

BÁO CÁO THỰC NGHIỆM

HỌC PHẦN: TRÍ TUỆ NHÂN TẠO

Đề tài: Nghiên cứu ứng dụng trí tuệ nhân tạo trong trò chơi cờ caro 3x3.

GVHD	**: Trần Thanh Huân**
Nhóm	**: 17**
Thành viên	**: Hà Phương Nam**
	Nguyễn Quang Hoàng
	Phạm Hoàng Tuấn
Lớp	**2024IT609400**

Hà Nội, Năm 2025

MỤC LỤC

MỤC LỤC **1**
DANH MỤC HÌNH ẢNH **3**
CHƯƠNG 1 : GIỚI THIỆU ĐỀ TÀI CỜ CARO 3X3 ..... **5**
1.1. LÝ DO CHỌN ĐỀ TÀI 5
1.2. MỤC TIÊU ĐỀ TÀI 5
1.2.1 Xây dựng chương trình chơi Cờ Caro 3x3 5
1.2.2 Áp dụng thuật toán tìm kiếm trong trí tuệ nhân tạo: 5
1.2.3 Phân tích và đánh giá chiến lược của máy: 5
1.2.4 Củng cố kiến thức và thực hành về AI 5
1.3 PHẠM VI NGHIÊN CỨU 6
1.3.1.Phạm vi trò chơi 6
1.3.2.Phạm vi thuật toán 6
1.3.3.Phạm vi kỹ thuật 6
1.3.4.Phạm vi người dùng 6
1.3.5. Phạm vi đánh giá 6
1.4 Ý NGHĨA THỰC TIỄN 6
CHƯƠNG 2:THUẬT TOÁN MINIMAX , CẮT TỈA ALPHA-BETA **7**
2.1.THUẬT TOÁN MINIMAX : 7
2.1.1.Tổng quan chung thuật toán Minimax 7
2.1.2. Giải thuật Minimax 7
2.1.3 Ưu điểm và nhược điểm 10
2.2.THUẬT TOÁN CẮT TỈA ALPHA-BETA 11
2.2.1. Tổng quan chung thuật toán cắt tỉa Alpha-Beta 11
2.2.2.Giải thuật cắt tỉa Alpha-Beta 11
2.2.3. Ưu điểm và nhược điểm 15
2.3. HÀM ĐÁNH GIÁ 15
2.3.1.Tổng quan chung hàm đánh giá 15
2.3.2. Hàm đánh giá trong thuật toán Minimax (Cờ Caro 3x3) 15
CHƯƠNG 3: XÂY DỰNG CHƯƠNG TRÌNH **17**
3.1 MÔ TẢ BÀI TOÁN 17
3.1.1 Biểu diễn trạng thái 17

3.1.2. Mô tả không gian trạng thái của bài toán: 18
3.1.3. Mục tiêu của cây trạng thái 19
3.2 CÀI ĐẶT THUẬT TOÁN 19
3.2.1 Ngôn ngữ và công cụ....19
3.2.2 Khởi tạo bàn cờ 19
3.2.3 Hàm in bàn cờ....19
3.2.4 Kiểm tra người thắng....20
3.2.5 Hàm đánh giá trạng thái bàn cờ....20
3.2.6 Kiểm tra bàn cờ đầy....20
3.2.7 Thuật toán Minimax có cắt tỉa Alpha-Beta 21
3.2.8 Tìm nước đi tốt nhất cho máy 22
3.2.9 Vòng lặp chính điều khiển trò chơi 22
3.2.10 Kết quả....24
KẾT LUẬN 27
TÀI LIỆU THAM KHẢO....28

DANH MỤC HÌNH ẢNH

Hình ảnh 1.Giải thuật Minimax.....8
Hình ảnh 2.Giải thuật cắt tỉa Alpha-Beta 11
Hình ảnh 3. Giải thuật hàm đánh giá.....16
Hình ảnh 4. Không gian trạng thái bài toán.....18
Hình ảnh 5. Khởi tạo bàn cờ.....19
Hình ảnh 6. Hàm in bàn cờ.....19
Hình ảnh 7. Hàm kiểm tra người thắng.....20
Hình ảnh 8 . Hàm đánh giá trạng thái bàn cờ.....20
Hình ảnh 9 . Hàm kiểm tra bàn cờ đầy.....20
Hình ảnh 11. Hàm tìm nước đi tốt nhất cho máy 22

LỜI NÓI ĐẦU

Trong bối cảnh công nghệ thông tin ngày càng phát triển mạnh mẽ, trí tuệ nhân tạo (AI) đang đóng vai trò quan trọng trong nhiều lĩnh vực như y tế, giáo dục, tài chính và đặc biệt là trong phát triển phần mềm thông minh. Việc tiếp cận và tìm hiểu các thuật toán nền tảng của AI không chỉ giúp sinh viên củng cố kiến thức lý thuyết, mà còn mở rộng tư duy lập trình và khả năng giải quyết vấn đề.

Với mong muốn vận dụng kiến thức đã học vào thực tế, chúng em lựa chọn đề tài "Ứng dụng thuật toán tìm kiếm vào trò chơi Cờ Caro 3x3" làm bài tập lớn môn Trí tuệ nhân tạo. Đây là một đề tài tuy đơn giản về mặt trò chơi nhưng lại chứa đựng nhiều yếu tố tư duy chiến lược, rất phù hợp để triển khai thuật toán Minimax – một trong những thuật toán tìm kiếm cơ bản và phổ biến trong AI.

Thông qua đề tài này, chúng em mong muốn xây dựng được một chương trình có khả năng chơi cờ thông minh, giúp người học hiểu rõ hơn về quá trình ra quyết định của máy tính trong môi trường có tính cạnh tranh. Đồng thời, đề tài cũng góp phần rèn luyện kỹ năng phân tích, thiết kế thuật toán và lập trình ứng dụng thực tiễn.

Chúng em xin chân thành cảm ơn thầy Trần Thanh Huân đã tận tình hướng dẫn, hỗ trợ và tạo điều kiện thuận lợi để chúng em hoàn thành tốt bài tập lớn này. Những góp ý và định hướng từ thầy là động lực giúp chúng em nâng cao kiến thức và hoàn thiện kỹ năng chuyên môn.

Chúng em kính mong nhận được sự góp ý của thầy cô để đề tài được hoàn thiện hơn.

Trân trọng cảm ơn!

CHƯƠNG 1 : GIỚI THIỆU ĐỀ TÀI CỜ CARO 3X3

1.1. LÝ DO CHỌN ĐỀ TÀI

- Cờ Caro là một trò chơi dân gian quen thuộc với nhiều thế hệ, đặc biệt là trong môi trường học đường. Với luật chơi đơn giản nhưng đòi hỏi tư duy chiến thuật, Cờ Caro trở thành một đối tượng lý tưởng để nghiên cứu và áp dụng các thuật toán trong lĩnh vực Trí tuệ nhân tạo (AI).

- Trong phạm vi đề tài này, nhóm chọn Cờ Caro kích thước 3x3 (hay còn gọi là *tic-tac-toe*) nhằm giới hạn không gian trạng thái và dễ dàng áp dụng các thuật toán tìm kiếm như Minimax hoặc Alpha-Beta pruning. Đây là một bài toán nhỏ nhưng mang tính học thuật cao, giúp minh họa rõ ràng cách mà máy tính có thể ra quyết định dựa trên việc đánh giá trạng thái và dự đoán nước đi của đối phương.

1.2. MỤC TIÊU ĐỀ TÀI

1.2.1 Xây dựng chương trình chơi Cờ Caro 3x3

- Phát triển một ứng dụng cho phép người chơi thi đấu với máy tính trong môi trường trò chơi Cờ Caro kích thước 3x3.

- Đảm bảo giao diện đơn giản, trực quan và dễ sử dụng.

1.2.2 Áp dụng thuật toán tìm kiếm trong trí tuệ nhân tạo:

- Sử dụng thuật toán Minimax để giúp máy tính đưa ra các nước đi tối ưu trong mọi trạng thái trò chơi.

- (Tùy chọn mở rộng) Tích hợp thuật toán Alpha-Beta pruning để tối ưu hiệu suất tìm kiếm.

1.2.3 Phân tích và đánh giá chiến lược của máy:

- Trình bày cách thuật toán ra quyết định dựa trên việc mô phỏng các nước đi khả dĩ và đánh giá trạng thái thắng/thua/hòa.

- Kiểm chứng khả năng "chơi không thua" của máy trong mọi ván đấu.

1.2.4 Củng cố kiến thức và thực hành về AI

- Tạo điều kiện áp dụng lý thuyết trí tuệ nhân tạo vào một bài toán cụ thể, thực tế và dễ tiếp cận.

- Giúp người học rèn luyện kỹ năng lập trình, tư duy giải thuật và xây dựng hệ thống ra quyết định tự động.

1.3 Phạm vi nghiên cứu

1.3.1.Phạm vi trò chơi

- Trò chơi được triển khai là Cờ Caro với kích thước 3x3 (hay còn gọi là *Tic-Tac-Toe*).
- Có 2 người chơi: người dùng và máy tính.
- Người chơi và máy luân phiên đánh dấu "X" và "O" trên bảng 3x3 cho đến khi thắng, hòa hoặc hết lượt đi.

1.3.2.Phạm vi thuật toán

- Áp dụng thuật toán Minimax để mô phỏng quá trình ra quyết định của máy tính.
- Trong trường hợp mở rộng, có thể sử dụng thêm Alpha-Beta Pruning để giảm số lượng trạng thái cần duyệt và tối ưu hiệu suất.

1.3.3.Phạm vi kỹ thuật

- Chương trình được xây dựng dưới dạng ứng dụng console hoặc giao diện đồ họa đơn giản (tùy theo yêu cầu hoặc năng lực nhóm).
- Không sử dụng thư viện AI phức tạp, mọi thuật toán được cài đặt thủ công nhằm mục đích học tập.

1.3.4.Phạm vi người dùng

- Ứng dụng dành cho một người chơi tương tác với hệ thống máy tính.
- Không bao gồm chế độ chơi hai người hoặc nhiều cấp độ khó.

1.3.5. Phạm vi đánh giá

- Đề tài tập trung vào độ chính xác và chiến lược của thuật toán, chưa ưu tiên hiệu năng xử lý cho các phiên bản trò chơi mở rộng như 5x5 hoặc đa chiều.

1.4 Ý nghĩa thực tiễn

Đề tài không chỉ giúp sinh viên nắm vững kiến thức về thuật toán tìm kiếm trong AI mà còn rèn luyện khả năng tư duy logic, phân tích trạng thái và lập trình giải quyết vấn đề. Mặc dù đơn giản, Cờ Caro 3x3 là nền tảng tốt để tiếp cận các trò chơi phức tạp hơn như Cờ vua, Cờ tướng hoặc các hệ thống ra quyết định thông minh trong tương lai.

CHƯƠNG 2:THUẬT TOÁN MINIMAX , CẮT TỈA ALPHA-BETA

2.1.Thuật toán Minimax :

2.1.1.Tổng quan chung thuật toán Minimax

Khái niệm : Minimax là 1 thuật toán tìm kiếm theo chiều sâu , dựa trên cây trò chơi được sử dụng để xác định nước đi tối ưu trong các trò chơi đối kháng 2 người , nơi 1 người chơi cố gắng tối đa điểm số (Max) và người chơi còn lại cố gắng tối thiểu điểm số (Min).

Cây trò chơi : một sơ đồ hình cây thể hiện từng trạng thái, từng trường hợp của trò chơi theo từng nước đi.

+ Mỗi node biểu diễn 1 trạng thái của trò chơi hiện tại trên cây trò chơi.

+ Node được gọi nút lá là tại đó trò chơi kết thúc (trạng thái trò chơi lúc đó có thể thắng, thua hoặc hòa).

Nguyên lý hoạt động : Hai người chơi trong game được đại diện là MAX và MIN. MAX đại diện cho người chơi luôn muốn chiến thắng và cố gắng tối ưu hóa ưu thế của mình còn MIN đại diện cho người chơi cố gắng cho người MAX giành số điểm càng thấp càng tốt.

2.1.2. 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.

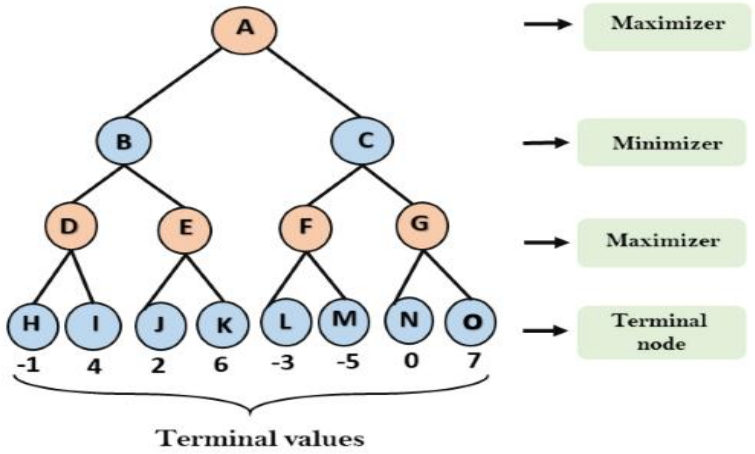
```python
def minimax(board, depth, is_maximizing):
    winner = check_winner(board)
    if winner != 0:
        return winner * 10  # 10 cho AI thắng, -10 cho người thắng
    if all(cell != 0 for row in board for cell in row):
        return 0  # Hòa

    if is_maximizing:
        best_score = -float('inf')
        for i in range(3):
            for j in range(3):
                if board[i][j] == 0:
                    board[i][j] = -1  # AI đánh
                    score = minimax(board, depth + 1, False)
                    board[i][j] = 0
                    best_score = max(best_score, score)
        return best_score
    else:
        best_score = float('inf')
        for i in range(3):
            for j in range(3):
                if board[i][j] == 0:
                    board[i][j] = 1  # Người đánh
                    score = minimax(board, depth + 1, True)
                    board[i][j] = 0
                    best_score = min(best_score, score)
        return best_score
```

Hình ảnh 1.Giải thuật Minimax

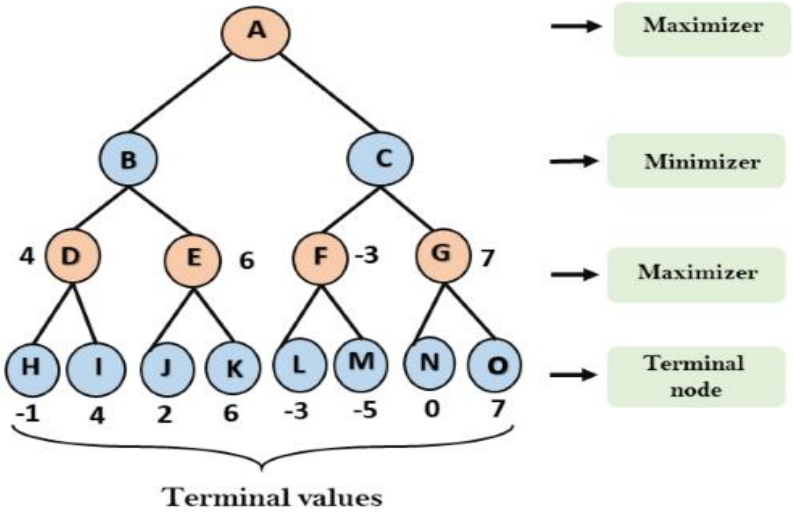
Ví dụ về cách hoạt động của thuật toán Minimax: Có 2 người chơi là Maximizer(sẽ cố gắng đạt được số điểm tối đa có thể) và người khác được gọi là Minimizer(sẽ cố gắng đạt được số điểm tối thiểu có thể.)

- Bước 1: Trong bước đầu tiên, thuật toán tạo ra Game Tree và áp dụng hàm tiện ích để nhận các giá trị và các trạng thái kết thúc. Trong sơ đồ cây dưới đây, hãy lấy A là trạng thái bắt đầu của Tree. Giả sử bộ tối đa hóa thực hiện lượt đi đầu tiên có giá trị ban đầu trong trường hợp xấu nhất = -∞ và minimizer sẽ thực hiện lượt tiếp theo có giá trị ban đầu trong trường hợp xấu nhất = +∞.



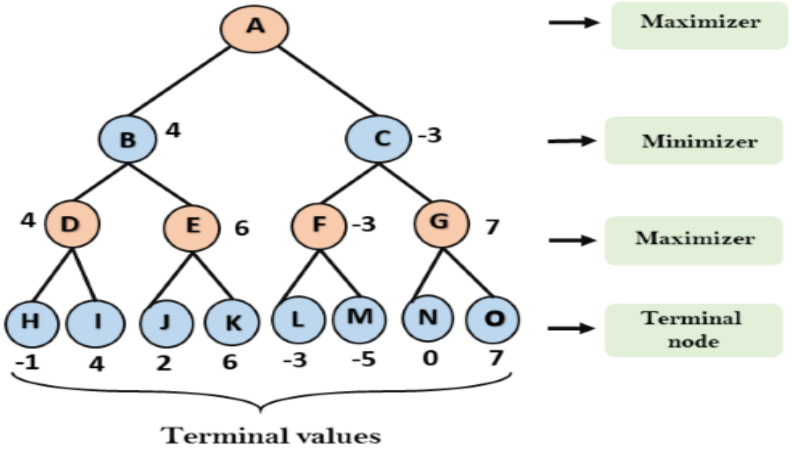
- Bước 2: Bây giờ, đầu tiên chúng ta tìm giá trị tiện ích cho Maximizer, giá trị ban đầu của nó là -∞, vì vậy chúng ta sẽ so sánh từng giá trị ở trạng thái đầu cuối với giá trị ban đầu của Maximizer và xác định các giá trị nút cao hơn. Nó sẽ tìm thấy mức tối đa trong số tất cả.

+ Đối với nút D max (-1, – -∞) => max (-1,4) = 4
+ Đối với nút E max (2, -∞) => max (2, 6) = 6
+ Đối với nút F max (-3, -∞) => max (-3, -5) = -3
+ Đối với nút G max (0, -∞) = max (0, 7) = 7



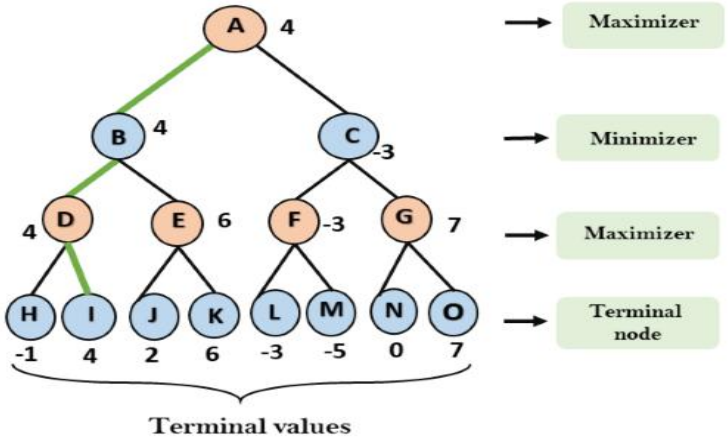
- Bước 3: Trong bước tiếp theo, đến lượt trình thu nhỏ, vì vậy nó sẽ so sánh giá trị tất cả các nút với + ∞ và sẽ tìm giá trị nút lớp thứ 3.

+ Đối với nút B = min (4,6) = 4
+ Đối với nút C = min (-3, 7) = -3



- Bước *4:* Bây giờ đến lượt Maximizer, nó sẽ lại chọn giá trị lớn nhất của tất cả các nút và tìm giá trị lớn nhất cho nút gốc. Trong Game Tree này, chỉ có 4 lớp, do đó chúng tôi truy cập ngay đến nút gốc, nhưng trong trò chơi thực, sẽ có nhiều hơn 4 lớp.

+ Đối với nút A max (4, -3) = 4



Đó là toàn bộ quy trình làm việc của trò chơi minimax hai người chơi.

2.1.3 Ưu điểm và nhược điểm

+ Ưu điểm: Tìm kiếm được mọi nước đi tiếp theo sau đó lựa chọn nước đi tốt nhất, vì giải thuật có tính chất vét cạn nên không bỏ soát trạng thái.

+ Nhược điểm:

● Đối với các trò chơi có không gian trạng thái lớn việc chỉ áp dụng giải thuật Minimax có lẽ không còn hiệu quả nữa do sự bùng nổ tổ hợp quá lớn.

● Giải thuật áp dụng nguyên lý vét cạn không tận dụng được thông tin của trạng thái hiện tại để lựa chọn nước đi, vì duyệt hết các trạng thái nên tốn thời gian

2.2.Thuật toán cắt tỉa Alpha-Beta

2.2.1. Tổng quan chung thuật toán cắt tỉa Alpha-Beta

Khái niệm: là một phiên bản cải tiến của thuật toán Minimax, tìm nước đi tối ưu mà không cần duyệt toàn bộ cây trò chơi. Bằng cách loại bỏ các nhánh không ảnh hưởng đến kết quả cuối cùng, Alpha-Beta giảm đáng kể thời gian tính toán mà vẫn đảm bảo kết quả tối ưu như Minimax.

Nguyên lý hoạt động: hoạt động tương tự Minimax nhưng sử dụng hai tham số Alpha và Beta để cắt tỉa các nhánh không cần thiết :

+ Alpha(α) :Giá trị tốt nhất mà người chơi Maximizer có thể đạt được
+ Beta (β) : Giá trị tốt nhất mà người chơi Minimizer có thể đạt được.
+ Khi $\alpha \geq \beta$, ta có thể cắt bỏ các nhánh còn lại vì chúng không thể cải thiện kết quả.

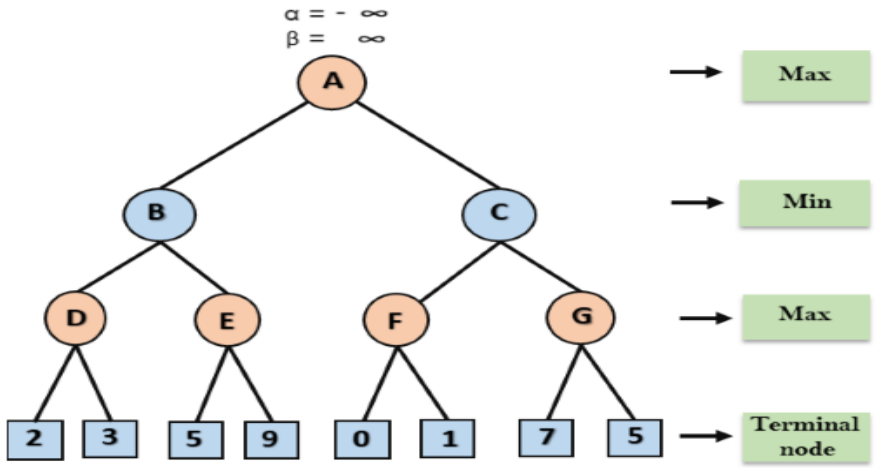
2.2.2.Giải thuật cắt tỉa Alpha-Beta

```
def alphabeta(board, depth, alpha, beta, is_maximizing):  2 usages
    if is_maximizing:
        max_eval = -float('inf')
        for move in get_available_moves(board):
            make_move(board, move,  param: 'X')
            eval = alphabeta(board, depth - 1, alpha, beta,  is_maximizing: False)
            undo_move(board, move)
            max_eval = max(max_eval, eval)
            alpha = max(alpha, eval)
            if beta <= alpha:
                break  # Cắt tỉa
        return max_eval
    else:
        min_eval = float('inf')
        for move in get_available_moves(board):
            make_move(board, move,  param: 'O')
            eval = alphabeta(board, depth - 1, alpha, beta,  is_maximizing: True)
            undo_move(board, move)
            min_eval = min(min_eval, eval)
            beta = min(beta, eval)
            if beta <= alpha:
                break  # Cắt
        return min_eval
```

Hình ảnh 2.Giải thuật cắt tỉa Alpha-Beta

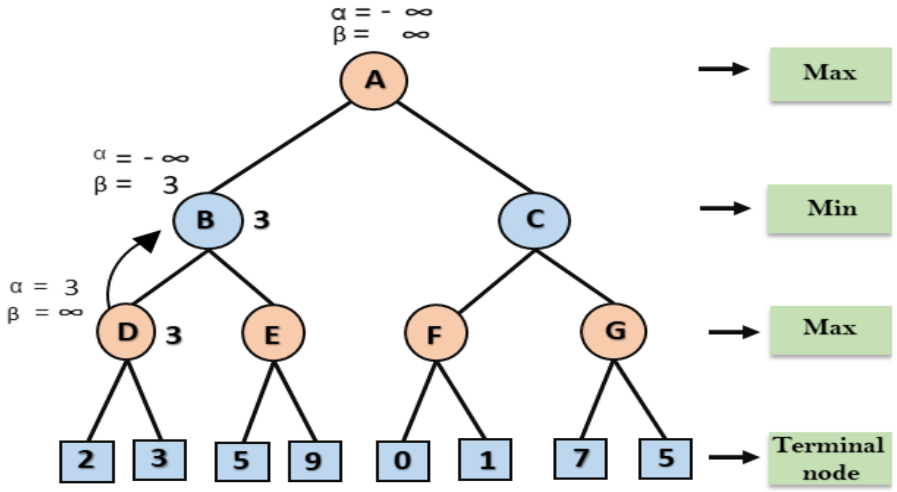
Ví dụ về hoạt động của thuật toán cắt tỉa Alpha-Beta:

Bước 1: Ở bước đầu tiên, người chơi Max sẽ bắt đầu di chuyển đầu tiên từ nút A nơi α = -∞ và β = + ∞, những giá trị alpha và beta này được truyền lại cho nút B nơi lại α = -∞ và β = + ∞ và nút B chuyển cùng một giá trị cho nút con D của nó.



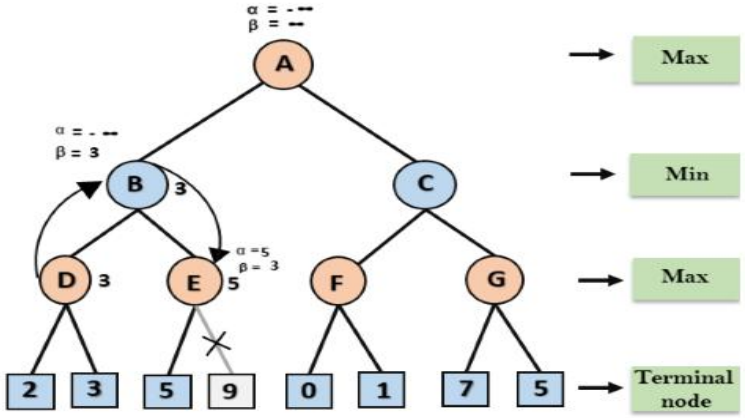
Bước 2: Tại nút D, giá trị của α sẽ được tính theo lượt của nó cho Max. Giá trị của α được so sánh với đầu tiên là 2 và sau đó là 3,và max (2,3) = 3 sẽ là giá trị của α tại nút D , giá trị của nút cũng sẽ là 3.

Bước 3: Bây giờ thuật toán quay ngược lại nút B, trong đó giá trị của β sẽ thay đổi vì đây là lượt của Min, Bây giờ β = + ∞, sẽ so sánh với giá trị của các nút tiếp theo có sẵn, tức là min (∞, 3) = 3, do đó tại nút B bây giờ α = -∞ và β = 3.



- Trong bước tiếp theo, thuật toán duyệt qua nút kế tiếp tiếp theo của nút B là nút E, và các giá trị của α = -∞ và β = 3 cũng sẽ được chuyển.

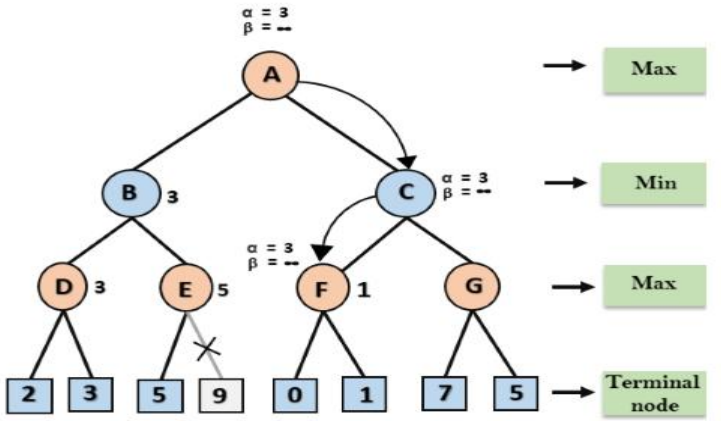
Bước 4: Tại nút E, Max sẽ đến lượt, giá trị của alpha sẽ thay đổi. Giá trị hiện tại của alpha sẽ được so sánh với 5, vì vậy max (-∞, 5) = 5, do đó tại nút E α = 5 và β = 3, trong đó α> = β, vì vậy kế thừa bên phải của E sẽ bị lược bỏ, và thuật toán sẽ không đi qua nó, và giá trị tại nút E sẽ là 5.



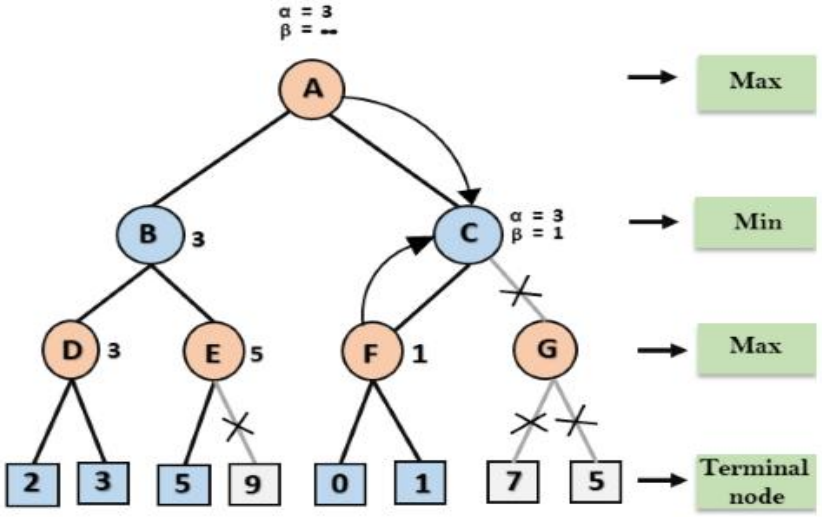
Bước 5: Ở bước tiếp theo, thuật toán lại chiếu ngược cây, từ nút B đến nút A. Tại nút A, giá trị của alpha sẽ được thay đổi giá trị lớn nhất có sẵn là 3 như max (-∞, 3) = 3 và β = + ∞, hai giá trị này bây giờ được chuyển đến người kế nhiệm bên phải của A là nút C.

- Tại nút C, α = 3 và β = + ∞, và các giá trị tương tự sẽ được chuyển cho nút F.

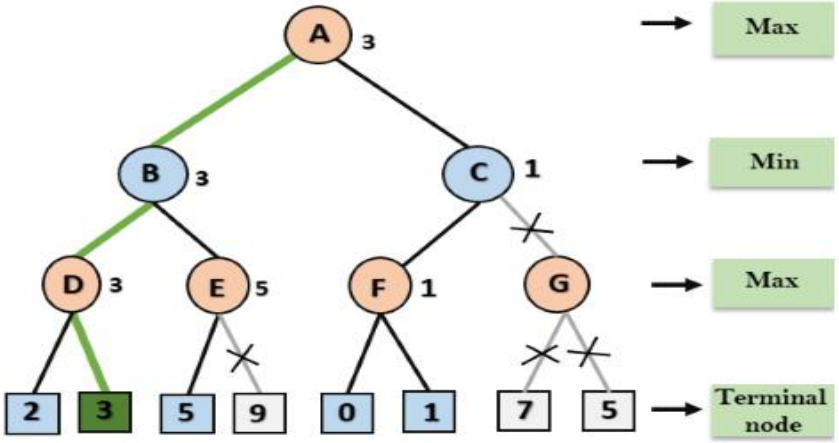
Bước 6: Tại nút F, một lần nữa giá trị của α sẽ được so sánh với nút con bên trái là 0 và max (3,0) = 3, sau đó so sánh với nút con bên phải là 1 và max (3,1) = 3 vẫn còn α vẫn là 3, nhưng giá trị nút của F sẽ trở thành 1.



Bước 7: Nút F trả về giá trị nút 1 cho nút C, tại C α = 3 và β = + ∞, ở đây giá trị beta sẽ được thay đổi, nó sẽ so sánh với 1 nên min (∞, 1) = 1. Bây giờ tại C, α = 3 và β = 1, và một lần nữa nó thỏa mãn điều kiện α> = β, vì vậy con tiếp theo của C là G sẽ bị lược bớt, và thuật toán sẽ không tính toàn bộ cây con G.



Bước 8: C bây giờ trả về giá trị từ 1 đến A ở đây giá trị tốt nhất cho A là max (3, 1) = 3. Sau đây là cây trò chơi cuối cùng là hiển thị các nút được tính toán và các nút chưa bao giờ tính toán. Do đó, giá trị tối ưu cho bộ cực đại là 3 cho ví dụ này.



2.2.3. Ưu điểm và nhược điểm

+ Ưu điểm: Tốc độ nhanh , tối ưu tài nguyên , tương thích tốt , hiệu suất tốt nhất khi sắp xếp đúng,...

+ Nhược điểm: Cài đặt phức tạp hơn , không cải thiện độ chính xác,…

2.3. Hàm đánh giá (Evaluation Function)

2.3.1.Tổng quan chung hàm đánh giá

- Khái niệm : Hàm đánh giá là một thành phần cốt lõi của thuật toán Minimax, dùng để ước lượng mức độ tốt của một trạng thái trò chơi khi chưa đạt đến trạng thái kết thúc

- Vai trò:

+ Gán một giá trị số cho trạng thái trò chơi, thể hiện lợi thế của một người chơi.

+ Giá trị dương thường biểu thị lợi thế cho MAX, giá trị âm biểu thị lợi thế cho MIN, và 0 biểu thị trạng thái cân bằng.

- Yêu cầu của hàm đánh giá:

+ Chính xác : Phản ánh đúng mức độ lợi thế của trạng thái.

+ Hiệu quả: Tính toán nhanh để không làm chậm thuật toán.

+ Phù hợp với trò chơi: Các yếu tố được đánh giá phải liên quan đến luật chơi và mục tiêu.

2.3.2. Hàm đánh giá trong thuật toán Minimax (Cờ Caro 3x3)

- Mục tiêu của hàm đánh giá :

Gán giá trị số cho mỗi trạng thái bàn cờ để Minimax xác định nước đi tốt nhất. Giá trị phản ánh mức độ có lợi cho người chơi MAX (‘X’) và bất lợi cho MIN (‘O’).

- Xác định giá trị cho trạng thái thắng/thua/hòa :

Với cờ caro , ta gán điểm như sau để biểu diễn kết quả cuối cùng của ván cờ :

'X' thắng : trả về **10**

'O' thắng : trả về **-10**

Hoà : trả về **0**

- Cách đánh giá :

+ Kiểm tra 8 đường chiến thắng: 3 hàng, 3 cột, 2 đường chéo

+ Nếu một đường có 3 quân giống nhau, xác định người thắng

+ Nếu không có bên nào thắng và không còn nước đi, trả về hòa

- Mã nguồn Python minh họa :

Hình minh họa hàm evaluate đơn giản theo quy ước trên. Hàm nhận vào board là danh sách 9 phần tử (mỗi phần tử có thể là 'X', 'O' hoặc None/ký tự khác đại diện cho ô trống) và trả về +10 nếu X thắng, −10 nếu O thắng, hoặc 0 trong các trường hợp khác.

```python
def evaluate(board):  1 usage
    lines = [
        (0, 1, 2), (3, 4, 5), (6, 7, 8),  # 3 hàng
        (0, 3, 6), (1, 4, 7), (2, 5, 8),  # 3 cột
        (0, 4, 8), (2, 4, 6)  # 2 đường chéo
    ]
    for a, b, c in lines:
        if board[a] == board[b] == board[c] and board[a] is not None:
            if board[a] == 'X':
                return 10
            elif board[a] == 'O':
                return -10
    return 0
```

Hình ảnh 3. Giải thuật hàm đánh giá

CHƯƠNG 3: XÂY DỰNG CHƯƠNG TRÌNH

3.1 Mô tả bài toán

a. Bàn cờ gồm 9 ô (3 hàng × 3 cột), ban đầu tất cả đều trống.

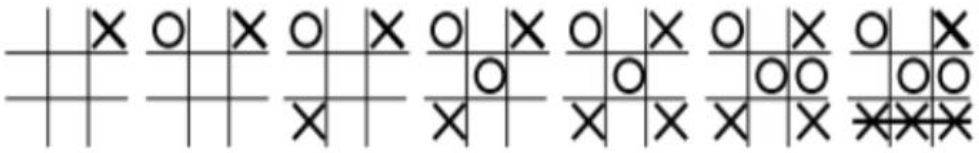
b. Hai người chơi thay phiên nhau đánh dấu vào một ô trống:

- Người chơi thứ nhất đánh ký hiệu "X".
- Người chơi thứ hai (máy hoặc người) đánh ký hiệu "O".

c. Người chơi nào đầu tiên có 3 ký hiệu giống nhau được sắp xếp liên tiếp theo hàng ngang, hàng dọc hoặc đường chéo sẽ chiến thắng.

d. Nếu tất cả các ô đã được đi nhưng không ai thắng, ván đấu kết thúc với kết quả hòa.

Ví dụ:



Các toán tử:

- Đánh dấu ô (1,1)
- Đánh dấu ô (1,2)
- Đánh dấu ô (1,3)
- Đánh dấu ô (2,1)
- Đánh dấu ô (2,2)
- Đánh dấu ô (2,3)
- Đánh dấu ô (3,1)
- Đánh dấu ô (3,2)
- Đánh dấu ô (3,3)

Trong đó i là chỉ số hàng (1 đến 3) và j là chỉ số cột (1 đến 3).

3.1.1 Biểu diễn trạng thái

a. Bàn cờ được biểu diễn bằng **ma trận 3x3** hoặc danh sách 9 phần tử.

b. Mỗi ô có thể có một trong ba trạng thái:)

+ 0: ô trống
+ 1: người chơi X
+ -1: người chơi O (AI)

Ví dụ trạng thái:

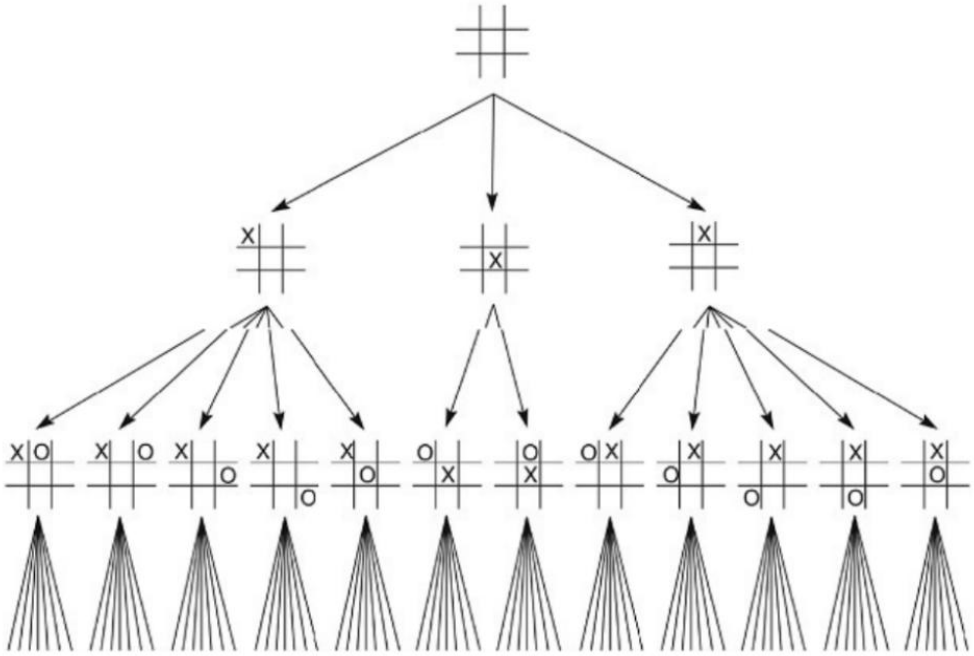
```
[
 [ 1,  0, -1],
 [ 0,  1, -1],
 [ 0,  0,  1]
]
```

Trạng thái trên cho thấy người chơi X đã thắng theo đường chéo chính.

3.1.2. Mô tả không gian trạng thái của bài toán:

a. Gốc cây:

- Bàn cờ trống - trạng thái ban đầu.



Hình ảnh 4. Không gian trạng thái bài toán

- Người chơi X đi đầu tiên — mỗi nhánh từ gốc tương ứng với một nước đi khác nhau của X vào một trong 9 ô trống.

b. Lượt đầu tiên:

- Người chơi X đi đầu tiên — mỗi nhánh từ gốc tương ứng với một nước 5đi khác nhau của X vào một trong 9 ô trống.

c. Lượt thứ hai:

- Sau khi X đi, máy (O) hoặc người chơi tiếp theo đánh vào một ô trống còn lại.
- Mỗi nhánh con của X sinh ra nhiều nhánh nhỏ hơn – thể hiện các nước đi của O.
- Mỗi cặp nước đi X – O tạo ra các trạng thái trò chơi tiếp theo.

d. Lượt thứ ba:

- Tiếp tục lượt X → cây được mở rộng ra nữa, với mỗi trạng thái O lại sinh ra nhiều nhánh X.

3.1.3. Mục tiêu của cây trạng thái

- Mô hình hóa toàn bộ không gian nước đi của trò chơi.
- Dùng trong thuật toán Minimax để:
- Duyệt từ lá → gốc.
- Gán giá trị thắng/thua/hòa cho các trạng thái lá.
- Lựa chọn nước đi tốt nhất cho người chơi ở gốc (thường là AI).

3.2 Cài đặt thuật toán

3.2.1 Ngôn ngữ và công cụ

a. Ngôn ngữ lập trình: Python

b. Thư viện sử dụng: ***import math***

3.2.2 Khởi tạo bàn cờ

```
board = [[0 for _ in range(3)] for _ in range(3)]
```

Hình ảnh 5. Khởi tạo bàn cờ

- Tạo ma trận 3x3, mỗi ô có giá trị:

+ 0: ô trống
+ 1: người chơi (ký hiệu 'X')
+ -1: máy (ký hiệu 'O')

3.2.3 Hàm in bàn cờ

```
def print_board():
    symbol = {0: ' ', 1: 'X', -1: 'O'}
    for row in board:
        print(" | ".join(symbol[cell] for cell in row))
        print("-" * 9)
```

Hình ảnh 6. Hàm in bàn cờ

- In ma trận bàn cờ ra màn hình với định dạng dễ đọc, thể hiện rõ các lượt đánh.

3.2.4 Kiểm tra người thắng

```
def check_winner(player):
    for i in range(3):
        if all(board[i][j] == player for j in range(3)): return True  # hàng
        if all(board[j][i] == player for j in range(3)): return True  # cột
    if all(board[i][i] == player for i in range(3)): return True      # chéo chính
    if all(board[i][2 - i] == player for i in range(3)): return True  # chéo phụ
    return False
```

Hình ảnh 7. Hàm kiểm tra người thắng

- Kiểm tra xem người chơi có thắng không bằng cách kiểm tra hàng, cột và hai đường chéo.
- player = 1 → người, player = -1 → máy.

3.2.5 Hàm đánh giá trạng thái bàn cờ

```
def evaluate():
    if check_winner(-1):
        return 1     # Máy thắng
    elif check_winner(1):
        return -1    # Người thắng
    return 0         # Hòa hoặc chưa ai thắng
```

Hình ảnh 8 . Hàm đánh giá trạng thái bàn cờ

- Trả về:

+ 1: máy thắng
+ -1: người thắng
+ 0: hòa hoặc chưa có ai thắng

- Giúp xác định kết quả trò chơi tại một trạng thái bất kỳ.

3.2.6 Kiểm tra bàn cờ đầy

```
def is_full():
    return all(board[i][j] != 0 for i in range(3) for j in range(3))
```

Hình ảnh 9 . Hàm kiểm tra bàn cờ đầy

- Trả về True nếu tất cả ô đã được đánh → kết thúc game.

3.2.7 Thuật toán Minimax có cắt tỉa Alpha-Beta

- Minimax: thuật toán đánh giá tất cả các nước đi có thể xảy ra để chọn nước đi tối ưu.

- Alpha-Beta Pruning:

+ alpha: giá trị tốt nhất mà người MAX (máy) có thể đảm bảo tại nút đó.

+ beta: giá trị tốt nhất mà người MIN (người chơi) có thể đảm bảo.

+ Nếu beta ≤ alpha, dừng đánh giá nhánh hiện tại (cắt tỉa).

- Hai chế độ:

+ is_maximizing = True: máy tính đang chọn nước đi tốt nhất.

+ is_maximizing = False: mô phỏng lượt đi của người chơi.

```
def minimax(depth, is_maximizing, alpha, beta):
    score = evaluate()
    if score != 0 or is_full():
        return score

    if is_maximizing:
        max_eval = -math.inf
        for i in range(3):
            for j in range(3):
                if board[i][j] == 0:
                    board[i][j] = -1  # Máy đánh
                    eval = minimax(depth + 1, False, alpha, beta)
                    board[i][j] = 0
                    max_eval = max(max_eval, eval)
                    alpha = max(alpha, eval)
                    if beta <= alpha:
                        return max_eval
        return max_eval
    else:
        min_eval = math.inf
        for i in range(3):
            for j in range(3):
                if board[i][j] == 0:
                    board[i][j] = 1  # Người đánh
                    eval = minimax(depth + 1, True, alpha, beta)
                    board[i][j] = 0
                    min_eval = min(min_eval, eval)
                    beta = min(beta, eval)
                    if beta <= alpha:
                        return min_eval
        return min_eval
```

Hình ảnh 10 . Thuật toán Minimax có cắt tỉa Alpha-Beta

3.2.8 Tìm nước đi tốt nhất cho máy

```
def best_move():
    best_val = -math.inf
    move = (-1, -1)
    for i in range(3):
        for j in range(3):
            if board[i][j] == 0:
                board[i][j] = -1
                move_val = minimax(0, False, -math.inf, math.inf)
                board[i][j] = 0
                if move_val > best_val:
                    best_val = move_val
                    move = (i, j)
    return move
```

Hình ảnh 11. Hàm tìm nước đi tốt nhất cho máy

- Duyệt toàn bộ các ô trống.
- Gọi minimax() để đánh giá từng nước đi và chọn nước có giá trị tốt nhất.

3.2.9 Vòng lặp chính điều khiển trò chơi

```
def play_game():
  print("Chào mừng đến với trò chơi Tic Tac Toe!")
  print("Bạn là 'X' (giá trị 1), máy là 'O' (giá trị -1).")
  print("Hãy nhập vị trí theo định dạng: hàng cột (vd: 1 1 để đánh ô góc trên bên trái).")
  print_board()

  while True:
    # ======= Lượt của người chơi =======
    while True:
      try:
        # Nhập nước đi (hàng và cột từ 1 đến 3)
        i, j = map(int, input("Nhập nước đi của bạn (hàng cột): ").split())
        i - = 1  # chuyển về chỉ số từ 0
        j - = 1
        # Kiểm tra hợp lệ và ô còn trống
```

```
            if 0 <= i < 3 and 0 <= j < 3 and board[i][j] == 0:
                board[i][j] = 1  # Người chơi đánh vào ô (i,j)
                break
            else:
                print("Vị trí không hợp lệ hoặc đã được đánh. Vui lòng thử lại.")
        except:
            print("Lỗi nhập. Hãy nhập đúng 2 số nguyên cách nhau bằng dấu cách.")
    print_board()

    # Kiểm tra kết quả sau lượt người chơi
    if check_winner(1):
        print("Bạn đã thắng! Xin chúc mừng!")
        break
    if is_full():
        print("Trò chơi hòa!")
        break

    # ======= Lượt của máy =======
    print("Máy đang suy nghĩ...")
    i, j = best_move()       # AI chọn nước đi tốt nhất
    board[i][j] = -1         # Máy đánh vào ô (i,j)
    print(f"Máy đánh vào ô ({i + 1}, {j + 1}):")
    print_board()

    # Kiểm tra kết quả sau lượt máy
    if check_winner(-1):
        print("Máy đã thắng. Chúc bạn may mắn lần sau!")
        break
    if is_full():
        print("Trò chơi hòa!")
        break
```

3.2.10 Kết quả

```
Chào mừng đến với trò chơi Tic Tac Toe!
Bạn là 'X' (giá trị 1), máy là 'O' (giá trị -1).
Hãy nhập vị trí theo định dạng: hàng cột (vd: 1 1 để đánh ô góc trên bên trái).
  |   |
---------
  |   |
---------
  |   |
---------
Nhập nước đi của bạn (hàng cột): 2 2
  |   |
---------
  | X |
---------
  |   |
---------
Máy đang suy nghĩ...
Máy đánh vào ô (1, 1):
O |   |
---------
  | X |
---------
  |   |
---------
Nhập nước đi của bạn (hàng cột): 3 1
O |   |
---------
  | X |
---------
X |   |
```

```
---------
Máy đang suy nghĩ...
Máy đánh vào ô (1, 3):
O |   | O
---------
  | X |
---------
X |   |
---------
Nhập nước đi của bạn (hàng cột): 1 2
O | X | O
---------
  | X |
---------
X |   |
---------
Máy đang suy nghĩ...
Máy đánh vào ô (3, 2):
O | X | O
---------
  | X |
---------
X | O |
---------
Nhập nước đi của bạn (hàng cột): 2 1
O | X | O
---------
X | X |
---------
X | O |
---------
```

```
Máy đang suy nghĩ...
Máy đánh vào ô (2, 3):
O | X | O
---------
X | X | O
---------
X | O |
---------
Nhập nước đi của bạn (hàng cột): 3 3
O | X | O
---------
X | X | O
---------
X | O | X
---------
Trò chơi hòa!
```

KẾT LUẬN

Trong khuôn khổ bài tập lớn học phần Trí tuệ Nhân tạo, nhóm đã lựa chọn và triển khai đề tài "Ứng dụng thuật toán tìm kiếm vào trò chơi Cờ Caro 3x3". Mặc dù đây là một trò chơi đơn giản về luật chơi và không gian trạng thái tương đối nhỏ, nhưng lại là một môi trường lý tưởng để áp dụng các thuật toán trí tuệ nhân tạo kinh điển như Minimax và cắt tỉa Alpha-Beta.

Thông qua quá trình nghiên cứu và thực hiện, nhóm đã:

- Tìm hiểu và phân tích chi tiết nguyên lý hoạt động của thuật toán Minimax và Alpha-Beta pruning.
- Hiểu được vai trò và cách xây dựng hàm đánh giá trong các trò chơi đối kháng. Cài đặt thành công một chương trình chơi Cờ Caro 3x3 có khả năng đưa ra các nước đi tối ưu, đảm bảo máy không thua trong mọi trường hợp.
- Rèn luyện kỹ năng tư duy giải thuật, lập trình Python và mô hình hóa bài toán AI trong môi trường thực tế.
- Kết quả đạt được không chỉ là một chương trình chơi cờ thông minh mà còn thể hiện sự hiểu biết sâu sắc hơn về cách AI ra quyết định trong môi trường cạnh tranh. Đây là nền tảng hữu ích để tiếp cận các bài toán phức tạp hơn như Cờ vua, Cờ tướng hay các hệ thống hỗ trợ ra quyết định trong tương lai.

Nhóm xin chân thành cảm ơn thầy Trần Thanh Huân đã tận tình hướng dẫn trong suốt quá trình thực hiện. Chúng em hy vọng sẽ tiếp tục nhận được sự góp ý quý báu từ thầy cô để hoàn thiện hơn các kỹ năng và kiến thức của mình.

TÀI LIỆU THAM KHẢO

1. Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson Education.
2. Jones, A. M. (2023). "Building a Smart Tic-Tac-Toe AI." *Journal of Game AI*, Vol. 5, No. 2, pp. 45–58, March 2023.
3. Brown, K. L. (2024). "Minimax Algorithm for Board Games." In *Proceedings of the IEEE International Conference on AI and Games*, London, UK, pp. 112–118.
4. Wikipedia contributors. *Minimax*. [Online]. Available: https://en.wikipedia.org/wiki/Minimax (Accessed: May 20, 2025).
5. GeeksforGeeks. *Minimax Algorithm in Game Theory*. [Online]. Available: https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/ (Accessed: May 20, 2025).
6. Google AI. *DeepMind and AlphaGo*. [Online]. Available: https://deepmind.com/research/highlighted-research/alphago (Accessed: May 20, 2025).
7. YouTube – Tech with Tim. *Minimax Algorithm Explained*. [Video]. Available: https://www.youtube.com/watch?v=l-hh51ncgDI (Accessed: May 18, 2025).