Báo cáo bài tập lớn môn Cơ sở trí tuệ nhân tạo

Dự Án : Cờ Vua với Trí Tuệ Nhân Tạo

Thành viên :

Tên	MSV	Đóng góp
Nguyễn Phương Đông	22022593	100%
Nguyễn Xuân Hiệp	22022591	100%
Phó Viết Tiến Anh	22022568	100%

▼ Giới thiệu sản phẩm

Trong dự án này, chúng tôi đã phát triển một trò chơi cờ vua với khả năng chơi cùng với AI sử dụng ngôn ngữ python cùng với các thư viện hỗ trợ như (pygame) để tạo ra 1 bàn cờ đẹp mắt và 1 con AI mạnh mẽ cho việc chơi cờ

▼ Thuật toán sử dụng

Minimax với cắt tỉa Alpha-Beta

```
In SmartMoveFinder.py file:
def findBestMinimaxMove(gs, validMoves):
    global nextMove
    nextMove = None
    random.shuffle(validMoves)
    findMiniMaxScore(gs, validMoves, DEPTH, -CHECKMATE, CHECI
    return nextMove

def findMiniMaxScore(gs, validMoves, depth, alpha, beta, turn
    global nextMove
    if depth == 0:
        return Evalute.evaluate_board(gs)
    if turnMutiplayer:
```

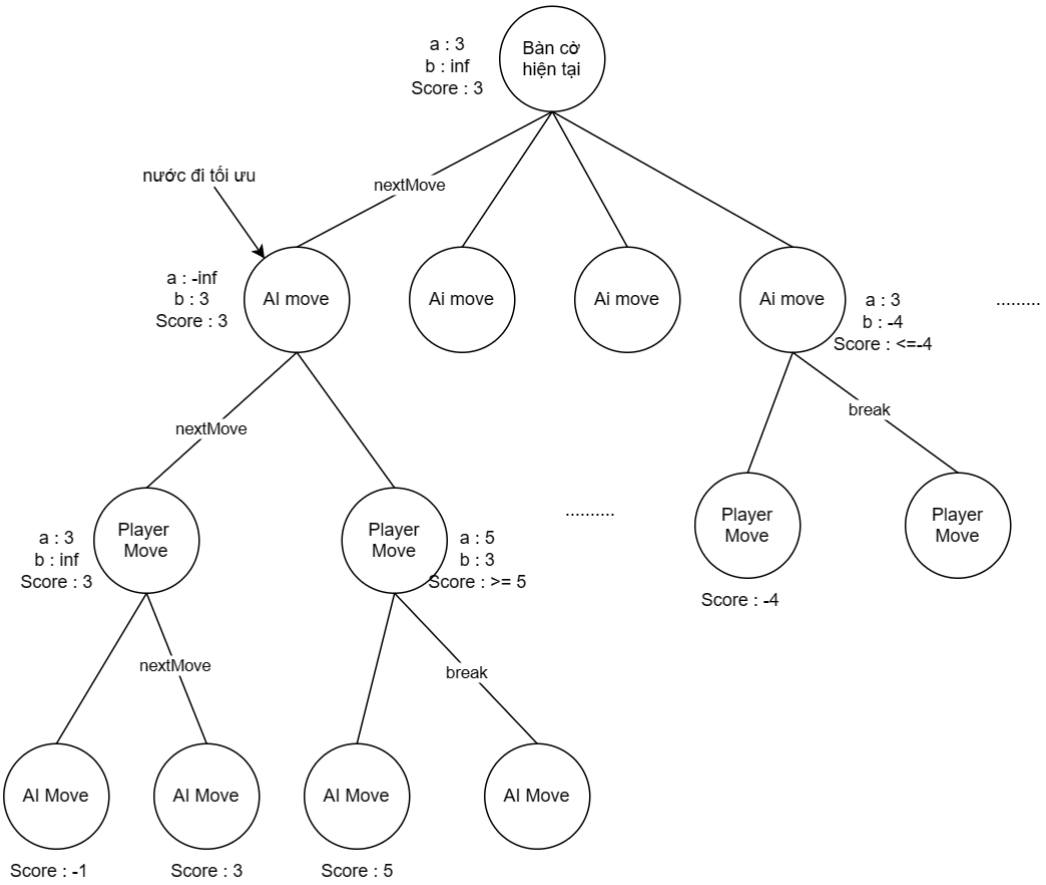
```
        maxScore = -CHECKMATE
        for move in validMoves:
            gs.makeMove(move)
            nextValidMoves = gs.getValidMoves()
            score = findMoveNegamax(gs, nextValidMoves, deptl
            if score > maxScore:
                maxScore = score
                if depth == DEPTH:
                    nextMove = move
            alpha = max(alpha, score)
            gs.undoMove()
            if beta <= alpha:
                break
        return maxScore
    else:
        minScore = CHECKMATE
        for move in validMoves:
            gs.makeMove(move)
            nextValidMoves = gs.getValidMoves()
            score = findMoveNegamax(gs, nextValidMoves, deptl
            if score < minScore:
                minScore = score
                if depth == DEPTH:
                    nextMove = move
            gs.undoMove()
            beta = min(beta, score)
            if beta <= alpha:
                break
        return minScore
```

Hàm `findBestMinimaxMove(gs, validMoves)` là hàm để tìm nước đi tốt nhất cho máy tính. Trong hàm này, sử dụng biến toàn cục `nextMove` để có thể lưu trữ nước đi tiếp theo ở mọi hàm phục vụ cho hàm `findMiniMaxScore`, danh sách các nước đi hợp lệ được xáo trộn để tăng tính ngẫu nhiên, sau đó hàm `findMiniMaxScore` được gọi để tìm kiếm đệ quy nước đi tốt nhất.

Trong `findMiniMaxScore(gs, validMoves, depth, alpha, beta, turnMutiplayer)`, thuật toán **Minimax với cắt tỉa Alpha-Beta** được thực hiện. Nó đánh giá trạng thái của bảng bằng cách sử dụng một phương pháp tìm kiếm đệ quy theo chiều sâu. Nếu đạt đến độ sâu cụ thể (**bằng 0**) hoặc một nút kết thúc, nó sẽ trả về giá trị đánh giá của trạng thái bảng bằng cách sử dụng hàm `Evalute.evaluate_board(gs)` **-Đây là 1 hàm đánh giá chúng ta sẽ bàn ở phần dưới của báo cáo**.

- Biến `turnMutiplayer` là biến kiểm tra xem đang là lượt của ai `turnMutiplayer==True` Nếu là lượt của người chơi còn AI là `False`
- Thuật toán lặp qua các `validMoves` - Các nước đi hợp lệ trên bàn cờ . Sau đó nó thực hiện nước đi đó và tính điểm nó đạt được cho nước đi tiếp theo thông qua hàm `Evalute.evaluate_board(gs)` **.** Sau đó nó tính toán xem điểm đạt được ở nước đi đó là tối ưu không **:**
  - Nếu là AI nó cần cực đại hóa điểm số
  - Nếu là người chơi nó cần cực tiểu hóa điểm số
- Thuật toán cũng đồng thời duy trì 2 giá trị `a lpha beta` **để cắt tỉa cây**
  - Nếu là người chơi nó cần cực đại hóa điểm số `alpha` với điểm AI
  - Nếu là AI nó cần cực tiểu hóa điểm số `beta` với người chơi
  - nếu alpha lớn hơn hoặc bằng beta nó sẽ cắt tỉa nhánh đó

  Dưới đây là ví dụ về cách thuật toán hoạt động



▼ Hàm đánh giá

Hàm đánh giá nhằm để tính điểm số dựa trên các quân trên bàn cờ và vị trí của chúng . Từ đó dựa vào điểm số để tìm nước đi tối ưu thông qua hàm đánh giá `Evalute.evaluate_board(gs)` và thuật toán Minimax và cắt tỉa alpha-beta chúng ta đã nêu ở trên.

```
In Evalute.py file:
def evaluate_board(gs) -> float:
    total = 0
    end_game = check_end_game(gs.board)
    if gs.checkMate:
        if gs.whiteToMove:
```

```
            return -CHECKMATE # black win
        else:
            return CHECKMATE #white win
    elif gs.staleMate:
        return STALEMATE

    for row in range(8):
        for square in range(8):
            piece = gs.board[row][square]
            if piece  == "--":
                continue
            value = piece_value[piece[1]] + evaluate_piece(pi
            total += value if piece[0] == "w" else -value
    return total
```

Có 2 đầu điểm chúng ta cần chú ý là:

Đlểm số của các quân cờ và Điểm vị trí các quân cờ:

- **Đlểm số các quân cờ** : Mỗi quân cờ được gán giá trị nhất đính tùy thuộc vào sức mạnh và mức độ quan trọng của từng con. Càng quan trọng thì điểm càng cao
- **Điểm số cho vị trí từng quân trên bàn cờ** : được tính bằng hàm `evaluate_piece(piece, square, location, end_game)` .Mỗi quân cờ đều có 1 list mảng 2 chiều tương ứng với bàn cờ chứa các điểm ở các vị trí khác nhau của 1 quân cờ
- Hàm dánh giá sẽ duyệt qua toàn bộ bàn cờ và tính tổng **Đlểm số các quân cờ + Điểm số cho vị trí từng quân trên bàn cờ.** Nếu là quân trắng thì cộng điểm, quân đen thì trừ điểm. Điều này cho thấy điểm số cao thì càng có lợi cho quân trắng và ngược lại