TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

———————— * ————————

BÁO CÁO MÔN HỌC

HỌC PHẦN: NHẬP MÔN TRÍ TUỆ NHÂN TẠO

Đề tài :

XÂY DỰNG GAME CỜ VUA SỬ DỤNG TRÍ TUỆ NHÂN TẠO

Giảng viên hướng dẫn: TS Đỗ Tiến Dũng

Sinh viên thực hiện:

Lớp: **136803**

Lê Đức Quân	20204682
Mai Vũ Duy	20204646
Nguyễn Hữu Truyền	20204698
Bùi Ngọc Thành	20204607

Hà Nội, tháng 11 năm 2022

Mục lục

I. Giới thiệu và mô tả về bài toán

1. Giới thiệu trò chơi cờ vua

Cờ vua là trò chơi 2 người, thể loại **zero-sum**.

Một người chơi sẽ cầm quân cờ màu trắng và người còn lại sẽ cầm quân cờ đen. Trò chơi sử dụng một bàn cờ hình vuông chia thành 64 ô vuông nhỏ hơn với 8 hàng ngang và 8 hàng dọc. Mỗi người chơi sẽ bắt đầu với 16 quân cờ, bao gồm 8 con tốt, 2 mã, 2 tượng, 2 xe, 1 hậu và 1 vua

Nguồn gốc : Hình thức chơi hiện tại của cờ vua bắt đầu xuất hiện ở Nam Âu ở nửa sau của thế kỷ 15. Ngày nay, cờ vua là một trong những trò chơi phổ biến nhất thế giới, được hàng triệu người trên toàn thế giới chơi tại nhà, ở câu lạc bộ, trên trực tuyến

Mặc dù chỉ có 64 ô và 32 quân cờ trên bàn cờ nhưng số lượng nước đi có thể được thì còn vượt xa cả số lượng các nguyên tử có trong vũ trụ.

2. Bài toán cần giải quyết

Xây dựng game cờ vua có sử dụng giải thuật trí tuệ nhân tạo

3. Mục tiêu, kết quả mong muốn

- Trò chơi có chế độ chơi (người vs người) và (người vs máy), trong đó máy có khả năng chơi cờ thông minh
- Áp dụng các thuật toán trong học phần trí tuệ nhân tạo vào trò chơi
- Phần mềm chạy ổn định và đưa ra quyết định các nước đi nhanh chóng

II. Phương pháp áp dụng

- **Các phương pháp áp dụng:**
  - Các thuật toán tìm kiếm nước đi : Alpha-beta prunning và Minimax
  - Các hàm đánh giá heuristic
- **Lý do lựa chọn :** Bởi vì lựa chọn hay đoán các nước đi tốt ở trạng thái hiện tại thì rất khó đạt được, chương trình cờ vua sẽ dựa trên việc **tìm kiếm** các nước đi và **đánh giá** bàn cờ sau khi thực hiện nước đi

III. Nội dung phương pháp

1. Thuật toán Minimax

Minimax là giải thuật là một thuật toán đệ quy lựa chọn bước đi kế tiếp trong một trò chơi có hai người bằng cách so sánh các Node trên cây trò chơi

Hai đối thủ trong trò chơi được gọi là MIN và MAX luân phiên thay thế nhau đi. MAX đại diện cho người quyết dành thắng lợi và cố gắng tối đa hóa ưu thế của mình, ngược lại người chơi đại diện cho MIN lại cố gắng giảm điểm số của MAX và cố gắng làm cho điểm số của mình càng âm càng tốt. Giả thiết đưa ra MIN và MAX có kiến thức như nhau về không gian trạng thái trò chơi và cả hai đối thủ đều cố gắng như nhau.

```
function MINIMAX-DECISION(state) returns an action
        v←MAX-VALUE(state)
    return the action in SUCCESSORS(state) with value v

function MAX-VALUE(state) returns a ultility state
        if TERMINAL-TEST(state) then return UTILITY(state)
    v← -∞
    for a,s in SUCCESSORS(state)do
        v←MAX(v,MIN-VALUE(s))
return v

function MIN-VALUE(state) returns a ultility state
        if TERMINAL-TEST(state) then return UTILITY(state)
    v← -∞
    for a,s in SUCCESSORS(state)do
        v←MIN(v,MAX-VALUE(s))
return v
```

Thuật toán Minimax

- **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 sót trạng thái.
- **Khuyết điểm** : Đối với các trò chơi có không gian trạng thái lớn như cờ vua 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. Thuật toán cắt tỉa Alpha - Beta

Thuật toán alpha-beta ra đời và dần thay thế thuật toán Minimax. Một trong số những ưu điểm đáng chú ý của alpha-beta đó là tinh giản được quá trình tìm nghiệm, loại trừ được một số trường hợp không thích hợp phát sinh nhưng vẫn đảm bảo không ảnh hưởng đến kết quả sau cùng.

Chiến lược cắt tỉa:

- Nút Max có một giá trị $alpha$ (luôn tăng), nút Min có một giá trị $beta$ (luôn giảm).
- Khi chưa có $alpha$ và $beta$ xác định thì thực hiện tìm kiếm sâu (depth-first) để xác định được $alpha$, $beta$ và truyền ngược lên các nút cha.

Phương pháp cắt tỉa alpha-beta:

- Nếu một nhánh tìm kiếm nào đó không thể cải thiện đối với giá trị (hàm tiện ích) mà chúng ta đã có, thì không cần xét đến nhánh tìm kiếm đó nữa!
- Việc cắt tỉa các nhánh tìm kiếm (“tỏi”) không ảnh hưởng đến kết quả cuối cùng

```
function alphabeta(node, depth, α, β, maximizingPlayer) is
    if depth = 0 or node is a terminal node then
        return the heuristic value of node
    if maximizingPlayer then
        value := −∞
        for each child of node do
            value := max(value, alphabeta(child, depth − 1,
α, β, FALSE))
            α := max(α, value)
            if value ≥ β then
                break (* β cutoff *)
        return value
    else
        value := +∞
        for each child of node do
            value := min(value, alphabeta(child, depth − 1,
α, β, TRUE))
            β := min(β, value)
            if value ≤ α then
                break (* α cutoff *)
        return value
```

```
(* Initial call *)
alphabeta(origin, depth, -∞, +∞, TRUE)
```

Thuật toán Alpha-Beta pruning

3. Các hàm heuristic áp dụng

3.1 Tổng điểm từng quân cờ

Giá trị này có ảnh hưởng nhất trong các hàm đánh giá. Mỗi quân cờ sẽ được gắn điểm riêng

Trong chương trình cờ vua hiện tại, sử dụng bảng điểm sau:

Quân cờ	Giá trị
Tốt	100
Mã	320
Tượng	330
Xe	500
Hậu	900
Vua	**20000**

Source: Simplified Evaluation Function - Chessprogramming wiki

Điểm tối đa của 1 bên dựa vào tổng điểm từng quân cờ là:

$$20000 + 9 * 900 + 2 * 500 + 4 * 300 = 30300$$

3.2 Điểm theo vị trí của từng quân

Vì mỗi quân cờ sẽ phát huy hết điểm mạnh dựa vào vị trí của nó trên bàn cờ. Do đó mỗi quân cờ sẽ có một ma trận điểm thưởng 8×8

- Ma trận điểm của quân mã:

```
knight_scores = [[-50, -40, -30, -30, -30, -30, -40, -50],
                 [-40, -20, 0, 0, 0, 0, -20, -40],
                 [-30, 0, 10, 15, 15, 10, 0, -30],
                 [-30, 5, 15, 20, 20, 15, 5, -30],
                 [-30, 0, 15, 20, 20, 15, 0, -30],
                 [-30, 5, 10, 15, 15, 10, 5, -30],
                 [-40, -20, 0, 5, 5, 0, -20, -40],
                 [-50, -40, -30, -30, -30, -30, -40, -50]]
```

Source: Simplified Evaluation Function - Chessprogramming wiki

Ma trận được tính dựa trên tính chất của quân mã. Mã đi theo hình chữ L nên rất yếu khi ở biên với góc bàn cờ và ngược lại càng mạnh khi ở gần trung tâm

- Ma trận điểm của quân tốt:

```
pawn_scores = [[0, 0, 0, 0, 0, 0, 0, 0],
               [50, 50, 50, 50, 50, 50, 50, 50],
               [10, 10, 20, 30, 30, 20, 10, 10],
               [5, 5, 10, 25, 25, 10, 5, 5],
               [0, 0, 0, 20, 20, 0, 0, 0],
               [5, 5, 10, 0, 0, 10, 5, 5],
               [5, 10, 10, 20, 20, 10, 10, 5],
               [0, 0, 0, 0, 0, 0, 0, 0]]
```

- Ma trận điểm của quân tượng:

```
bishop_scores = [[-20, -10, -10, -10, -10, -10, -10, -20],
                 [-10, 0, 0, 0, 0, 0, 0, -10],
                 [-10, 0, 5, 10, 10, 5, 0, -10],
                 [-10, 5, 5, 10, 10, 5, 5, -10],
                 [-10, 0, 10, 10, 10, 10, 0, -10],
                 [-10, 10, 10, 10, 10, 10, 10, -10],
                 [-10, 5, 0, 0, 0, 0, 5, -10],
                 [-20, -10, -10, -10, -10, -10, -10, -20]]
```

- Ma trận điểm của quân xe:

```
rook_scores = [[0, 0, 0, 0, 0, 0, 0, 0],
               [5, 10, 10, 10, 10, 10, 10, 5],
               [-5, 0, 0, 0, 0, 0, 0, -5],
               [-5, 0, 0, 0, 0, 0, 0, -5],
               [-5, 0, 0, 0, 0, 0, 0, -5],
               [-5, 0, 0, 0, 0, 0, 0, -5],
               [-5, 0, 0, 0, 0, 0, 0, -5],
               [0, 0, 0, 5, 5, 0, 0, 0]]
```

- Ma trận điểm của quân hậu:

```
queen_scores = [[-20, -10, -10, -5, -5, -10, -10, -20],
                [-10, 0, 0, 0, 0, 0, 0, -10],
                [-10, 0, 5, 5, 5, 5, 0, -10],
                [-5, 0, 5, 5, 5, 5, 0, -5],
                [0, 0, 5, 5, 5, 5, 0, -5],
                [-10, 5, 5, 5, 5, 5, 0, -10],
                [-10, 0, 5, 0, 0, 0, 0, -10],
                [-20, -10, -10, -5, -5, -10, -10, -20]]
```

Ma trận điểm của quân vua:

```
king_middle_score = [[-30, -40, -40, -50, -50, -40, -40, -30],
                     [-30, -40, -40, -50, -50, -40, -40, -30],
                     [-30, -40, -40, -50, -50, -40, -40, -30],
                     [-30, -40, -40, -50, -50, -40, -40, -30],
                     [-20, -30, -30, -40, -40, -30, -30, -20],
                     [-10, -20, -20, -20, -20, -20, -20, -10],
                     [20, 20, 0, 0, 0, 0, 20, 20],
                     [20, 30, 10, 0, 0, 10, 30, 20]]

king_end_score = [[-50, -40, -30, -20, -20, -30, -40, -50],
                  [-30, -20, -10, 0, 0, -10, -20, -30],
                  [-30, -10, 20, 30, 30, 20, -10, -30],
                  [-30, -10, 30, 40, 40, 30, -10, -30],
                  [-30, -10, 30, 40, 40, 30, -10, -30],
                  [-30, -10, 20, 30, 30, 20, -10, -30],
                  [-30, -30, 0, 0, 0, 0, -30, -30],
                  [-50, -30, -30, -30, -30, -30, -30, -50]]
```

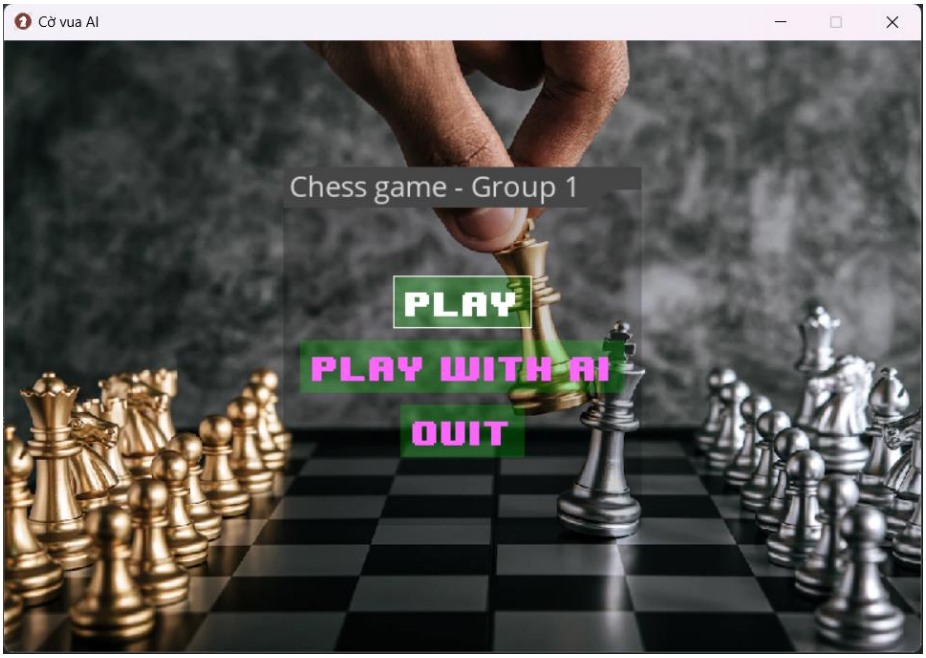
IV. Thực hiện, xây dựng chương trình

1. Mô tả chương trình

- Chương trình được xây dựng trên môi trường Window
- Ngôn ngữ lập trình sử dụng : Python
- Môi trường phát triển: PyCharm Community Edition
- Mã nguồn được lưu trữ trên github: [QuanBKIT1/G1-chess-ai (github.com)]
- Hướng dẫn cách chạy chương trình ở file README.md
- Các thư viện sử dụng: pygame, pygame-gui, pygame-menu

2. Giao diện chương trình

Menu trước khi vào game

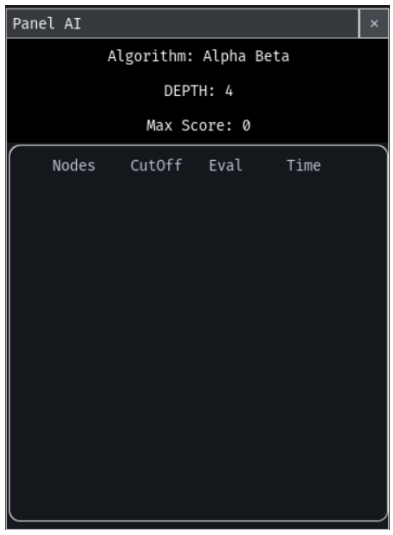


Giao diện ở chế độ người với người:

Giao diện ở chế độ người với máy:

Ở bên phải của bàn cờ có các bảng trạng thái:

Bảng 1



Bảng 2

Bảng 1: Hiển thị các thông tin của bên đang có quyền đi: Lượt đi, các nước đi có thể, có bị chiếu tướng hay không, và bảng hiển thị các nước đi của 2 bên tính từ lúc bắt đầu trận

Bảng 2: Hiển thị thông tin liên quan đến đến máy (AI): giải thuật, độ sâu tìm kiếm, điểm sau khi thực hiện tìm kiếm, và bảng chứa thông tin về số node tìm được, nhánh được cắt bỏ, số node lá (hay là số lần gọi hàm đánh giá) và thời gian chạy.

Các phím tắt hỗ trợ trong quá trình chơi game:

Phím Z : Để undo lại nước đi

Phím R: Để reset lại game

3. Tìm hiểu sơ lược về chương trình

- **Biểu diễn bàn cờ (Board Representation)** :

- Lưu trị vị trí các quân cờ để phục vụ cho việc **tìm kiếm, đánh giá** và vẽ giao diện

- Sử dụng mảng 8 × 8 với mỗi phần tử trong mảng là kiểu xâu với 2 kí tự đại diện mỗi quân cờ.

```
self.board = [
    ["bR", "bN", "bB", "bQ", "bK", "bB", "bN", "bR"],
    ["bp", "bp", "bp", "bp", "bp", "bp", "bp", "bp"],
    ["--", "--", "--", "--", "--", "--", "--", "--"],
    ["--", "--", "--", "--", "--", "--", "--", "--"],
    ["--", "--", "--", "--", "--", "--", "--", "--"],
    ["--", "--", "--", "--", "--", "--", "--", "--"],
    ["wp", "wp", "wp", "wp", "wp", "wp", "wp", "wp"],
    ["wR", "wN", "wB", "wQ", "wK", "wB", "wN", "wR"]]
```

Nhận xét:

Cách biểu diễn trực quan do đó giảm bớt độ phức tạp khi xây dựng chương trình

Nhược điểm: Hiệu năng kém so với nhiều phương pháp biểu diễn khác như **bitboards**

- **Sinh các nước đi hợp lệ (Legal Move Generation):**

- Các nước đi được gọi là hợp lệ nếu nước đi đó không để tướng bị chiếu
- Là 1 phần quan trọng việc xây dựng chương trình, phụ thuộc lớn vào việc **biểu diễn bàn cờ**.

Tham khảo: Generating Legal Chess Moves Efficiently • Peter Ellis Jones

Ý tưởng khi sinh các nước đi hợp lệ:

- Kiểm tra xem vua có bị chiếu không?

- Nếu vua bị chiếu thì việc sinh ra các nước đi sẽ như thế nào?

- **Các chiến lược tìm kiếm**

Như đã giới thiệu ở phần trên, chương trình sử dụng 2 thuật toán tìm kiếm là Minimax và Alpha-Beta Prunning

Sau đây là phương thức gọi hàm sử dụng Minimax và Alpha-Beta Prunning (nằm trong file ./engnie/AIEngine)

```
def __MiniMax(self, gs: GameState, depth, maximizingPlayer):
```

```
def __AlphaBetaPruning(self, gs: GameState, depth, alpha, beta, maximizingPlayer):
```

```
if depth == 0:
    self.total_nodes_leaf += 1
    self.total_node += 1
    return AIEngine.evaluation(gs)
```

Nếu depth == 0 khi đó gọi trả về giá trị của hàm đánh giá

Nếu depth != 0 thì khi đó sinh các nước đi hợp lệ từ bàn cờ, duyệt các nước đi hợp lệ và gọi đệ quy:

```
moves = gs.getValidMoves()
if maximizingPlayer:
    maxEval = - math.inf
    for move in moves:
        gs.makeMove(move)
        eval_score = self.__MiniMax(gs, depth - 1, False)
        gs.undoMove()
        if eval_score > maxEval:
            maxEval = eval_score
            if depth == DEPTH:
                self.bestMove = move
    return maxEval
```

- **Các hàm đánh giá trạng thái bàn cờ**

Như đã đề cập ở phần trên, chương trình sử dụng 2 hàm đánh giá cơ bản là: **Tổng điểm từng quân cờ** và **Điểm theo vị trí của từng quân**

Phương thức gọi hàm đánh giá được sử dụng trong chương trình:

```
# Heuristic 1
@staticmethod
def getMaterialScore(gs):
    white_score = 0
    black_score = 0
    for row in range(len(gs.board)):
        for col in range(len(gs.board[row])):
            piece = gs.board[row][col]
            if piece != "--":
                if piece[0] == "w":
                    white_score += piece_score[piece[1]]
                else:
                    black_score += piece_score[piece[1]]
    return black_score - white_score
```

Tổng điểm từng quân cờ

```
# Heuristic 2
@staticmethod
def getPiecePositionScore(gs):
    score = 0
    for row in range(len(gs.board)):
        for col in range(len(gs.board[row])):
            piece = gs.board[row][col]
            if piece != "--":
                if piece[0] == "b":
                    score += piece_position_scores[piece][row][col]
                else:
                    score -= piece_position_scores[piece][row][col]
    return score
```

Điểm theo vị trí của từng quân

4. Kiểm thử và Nhận xét

- **Kiểm thử :**

Khi đặt độ sâu là 4, AI mất trung bình 4s để thực hiện nước đi

Dựa vào bảng trên, thời gian trung bình là 12000 nodes/s

Khi đặt độ sâu là 3, AI mất trung bình 0.2s để thực hiện nước đi

- **Kết quả** : Một chương trình trò chơi cờ vua ứng dụng trí tuệ nhân tạo trong việc tính toán các nước đi của máy, dễ sử dụng và gần gũi với người dùng
- **Hạn chế :**
  - Máy thực hiện các nước đi chậm chạp khi tăng độ sâu tìm kiếm lên 5
  - Giao diện chưa được hấp dẫn
  - Các hàm đánh giá còn đơn giản nên chưa đánh giá đủ tốt thế trận

5. Hướng phát triển

- Cải thiện hiệu năng của chương trình để máy thực hiện nhanh ở độ sâu lớn hơn
- Tìm hiểu thêm các hàm đánh giá để cải thiện chất lượng chơi cờ của máy
- Cải tiến phiên bản cờ vua mới với đầy đủ các chức năng như: Vẽ các hiệu ứng khi di chuyển quân cờ, lựa chọn độ khó của máy, thay đổi màu nền bàn cờ trực tiếp,…

V. Thông tin khác

1. Phân công công việc

- Lê Đức Quân : Viết mã nguồn, làm báo cáo, làm slide
- Mai Vũ Duy : Tìm hiểu các hàm đánh giá, làm báo cáo
- Bùi Ngọc Thành : Nghiên cứu thuật toán, làm slide
- Nguyễn Hữu Truyền : Nghiên cứu thuật toán, làm báo cáo

2. Tham khảo

1. Chessprogramming wiki
2. “Artificial Intelligence A Modern Approach (3rd Edition)” - Stuart J. Russell and Peter Norvig

3. Generating Legal Chess Moves Efficiently • Peter Ellis Jones