**TRƯỜNG ĐẠI HỌC THỦ DẦU MỘT**

**VIỆN KỸ THUẬT - CÔNG NGHỆ**

****

**BÁO CÁO GIỮA KỲ**

**MÔN HỌC:**

**NHẬP MÔN TRÍ TUỆ NHÂN TẠO**

**ĐỀ TÀI: A.I CỜ VUA**

**GVHD : NGUYỄN LÊ HIỀN DUYÊN**

**NHÓM THỰC HIỆN**

|  |  |
| --- | --- |
| ***Nguyễn Tiến Phong*** | ***1824801030131*** |
|  |  |
|  |  |
|  |  |
|  |  |

*Bình Dương, tháng 4 năm 2021*

**MỤC LỤC**

[CHƯƠNG I: TỔNG QUAN 2](#_Toc69065170)

[1. Mô tả bài toán: 2](#_Toc69065171)

[2. Phương pháp sử dụng: 2](#_Toc69065172)

[a. Giải thuật tìm kiếm Minimax: 2](#_Toc69065173)

[b. Giải thuật cắt tỉa Alpha Beta: 3](#_Toc69065174)

[CHƯƠNG II: QUÁ TRÌNH THỰC HIỆN 5](#_Toc69065175)

[1. Bàn cờ và các quân cờ: 5](#_Toc69065176)

[a. Các quân cờ (Piece): 5](#_Toc69065177)

[b. Bàn cờ: 5](#_Toc69065178)

[2. Tìm kiếm nước đi hợp lệ (Move Generation): 7](#_Toc69065179)

[a. Cách di chuyển: 7](#_Toc69065180)

[b. Nước đi (Move): 7](#_Toc69065181)

[c. Move Generator: 8](#_Toc69065182)

[3. Tìm kiếm và hàm đánh giá (Search and Evaluation): 10](#_Toc69065183)

[a. Evaluation: 10](#_Toc69065184)

[b. Move Ordering: 10](#_Toc69065185)

[c. Search: 11](#_Toc69065186)

[TÀI LIỆU THAM KHẢO 12](#_Toc69065187)

# 

# LỜI MỎ ĐẦU

# CHƯƠNG I: TỔNG QUAN

## Mô tả bài toán:

* Sử dụng giải thuật tìm kiếm viết AI cho trò chơi cờ vua.

## Phương pháp sử dụng:

### Giải thuật tìm kiếm Minimax:

* Giải thuật Minimax là một thuật toán đệ quy lựa chọn bước đi kế tiếp trong một trò chơi có hai người.
* Xét một trò chơi đối kháng trong đó hai người thay phiên đi nước đi của mình như tic-tac-toe, cờ vua, cờ tướng, cờ caro, cờ vây…
* Khi chơi bạn có thể khai triển hết không gian trạng thái nhưng khó khăn chủ yếu là bạn phải tính toán được phản ứng và nước đi của đối thủ mình như thế nào?
* Cách xử lý đơn giản là bạn giả sử đối thủ của bạn cũng sử dụng kiến thức về không gian trạng thái giống bạn.
* Giải thuật Minimax áp dụng giả thuyết này để tìm kiếm không gian trạng thái của trò chơi.

**Các bước giải thuật Minimax:**

* Nếu như đạt đến giới hạn tìm kiếm (đến tầng dưới cùng của cây tìm kiếm tức là trạng thái kết thúc của trò chơi).
* Tính giá trị của thế cờ hiện tại ứng với người chơi ở đó. Ghi nhớ kết quả.
* Nếu như mức đang xét là của người chơi cực tiểu (nút MIN), áp dụng thủ tục Minimax này cho các con của nó. Ghi nhớ kết quả nhỏ nhất.
* Nếu như mức đang xét là của người chơi cực đại (nút MAX), áp dụng thủ tục Minimax này cho các con của nó. Ghi nhớ kết quả lớn nhất.

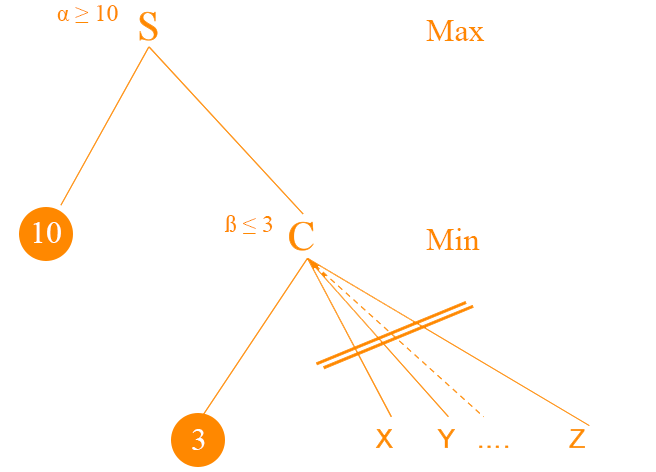
### Giải thuật cắt tỉa Alpha Beta:

**Khái niệm:**

* Giải thuật cắt tỉa Alpha-beta từng được nhiều nhà khoa học máy tính đề xuất ý tưởng và không ngừng được cải tiến cho đến ngày nay.
* Giải thuật này thường sử dụng chung với thuật toán tìm kiếm Minimax nhằm hỗ trợ giảm bớt các không gian trạng thái trong cây trò chơi, giúp thuật toán Minimax có thể tìm kiếm sâu và nhanh hơn.
* Giải thuật cắt tỉa Alpha-beta có nguyên tắc đơn giản "Nếu biết là trường hợp xấu thì không cần phải xét thêm".

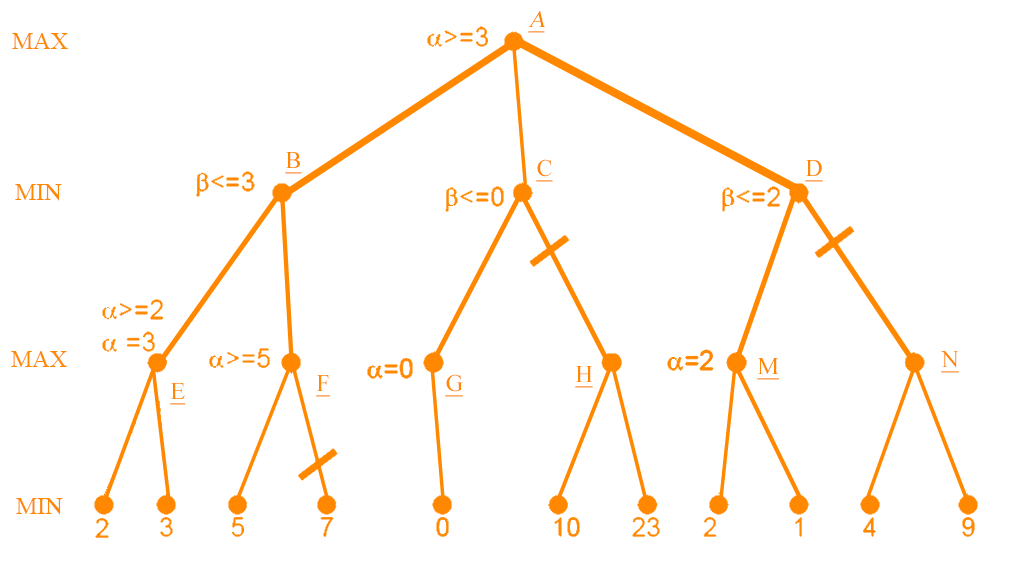
**Chiến lược cắt tỉa:**

* Nút Max có một giá trị alpha (lớn hơn hoặc bằng alpha – luôn tăng), nút min có một giá trị beta (nhỏ hơn hoặc bằng 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.



Hình Ví dụ Minimax + alphabeta

* Đầu tiên là xét cây từ trái sang phải ta sẽ thấy S là Max, theo chiến lược đưa ra vậy chúng ta sẽ có giá trị alpha ≥ 10 tại S.
* Tiếp theo, ở C ở đây là nút Min (trạng thái trò chơi dành cho Max) tức là sẽ lấy giá trị nhỏ nhất của các nút con ở dưới. Nếu như vậy thì giá trị chúng ta phải lấy là beta ≤ 3.
* Sau khi xác định được alpha và beta, chúng ta có thể dễ dàng xác định việc có cắt tỉa hay không. Ở nút S (Max), giá trị alpha luôn ≥ 10 (luôn tăng) nhưng ở C (Min) thì giá trị luôn luôn ≤ 3 (luôn giảm), nên việc xét các con còn lại ở C là không cần thiết.
* Nếu theo khoảng thì hiện tại chúng ta chỉ nhận khoảng ≥ 10 tại nút gốc  S, vậy thì đâu cần bận tâm đến việc khoảng ≤ 3 tại nút C.



Hình Cây trò chơi

* Ở đây chúng ta cũng xét từ trái qua phải bắt đầu từ nút gốc và nút con bên trái sẽ được ưu tiên duyệt trước.
* Xét duyệt từ trên gốc xuống sâu (vì ban đầu chưa hề tồn tại giá trị alpha hay beta của các nút).
* Nút đầu tiên ta duyệt là E sẽ gặp giá trị 2 (alpha ≥ 2), khi đó ở trên chưa có giá trị beta để ta có thể so sánh nên sẽ bắt đầu duyệt con tiếp theo của nút E đó và ở đây ta sẽ chọn cho alpha = 3 (Max).
* Lưu ý là luôn luôn duyệt từ trái sang phải và phải lần lượt từng nhánh một, sau đó sang nhánh tiếp theo cùng gốc. Vậy nên tiếp theo chúng ta sẽ đưa giá trị alpha này lên nút B (Min) và nút B – beta ≤ 3, sau đó nút F sẽ được duyệt, và ta phải tìm alpha của F.
* Khi duyệt nút đầu tiên mang giá trị 5 vậy alpha của F – alpha ≥ 5.
* Tại B – beta ≤ 3 và tại F – alpha ≥ 5. Như vậy chúng ta không cần xem xét các nút con còn lại của F vì cái ta cần ở đây chỉ là khoảng ≤ 3 nên ta cắt toàn bộ các con còn lại.
* Sau khi duyệt toàn bộ các con của B thì tại B – beta = 3, và tại nút A – alpha ≥ 3.

# CHƯƠNG II: QUÁ TRÌNH THỰC HIỆN

## Bàn cờ và các quân cờ:

### Các quân cờ (Piece):



Hình : Cách lưu dữ liệu quân cờ

* Mỗi quân cờ sẽ được gán một giá trị tượng trưng cho loại và màu của chúng.
* Hai bit đầu tiên cho ta biết về màu của quân cờ và 3 bit cuối cùng cho ta biết về loại của nó.

Ex: 01011: Knight trắng

### Bàn cờ:

* **Hiển thị các ô trên bàn cờ:**
* Bàn cờ gồm 8x8 ô có màu được hiển thị xen kẽ.
* Màu của một ô có thể được xác định bằng cách lấy (rank + file) % 2. Nếu nó bằng 0 thì ô này sẽ có màu sáng. Nếu nó khác 0 thì ô này sẽ có màu tối.
* **Lớp Board:**

##### Chức năng:

* Bao gồm một mảng 1 chiều Squares mỗi phần tử là một quân cờ trên bàn cờ. Nếu phần tử có giá trị bằng 0 thì ô đó không có quân cờ nào.



Hình : Các mảng lưu vị trí của từng loại quân cờ

* Lớp PieceList chứa vị trí các ô bị chiếm giữ.
* Mỗi mảng là lưu 2 đối tượng PieceList một dành cho các quân trắng và một dành cho các quân đen.
* Lớp PieceList dùng để tiện cho việc tìm kiếm các nước đi hợp lệ thông qua việc lưu trữ vị trí các ô mà từng loại quân cờ đang chiếm giữ.

##### Hiển thị các quân cờ trên bàn cờ:

* Forsyth – Edwards Notation(FEN): qui ước dùng để miêu tả một thế cờ:
* Vị trí các quân cờ: bắt đầu từ 8 và đi dần xuống 1, các dòng được ngăn cách bằng dấu ‘/’. Các quân cờ được kí hiệu bằng các chữ cái (pawn = "P", knight = "N", bishop = "B", rook = "R", queen = "Q" and king = "K"), nếu màu trắng thì sẽ in hoa, màu đen sẽ viết thường. Các ô trống được ghi bằng số.
* Màu tới lượt đi tiếp: “w” nghĩa là tới lượt trắng, “b” là tới lượt đen.
* Quyền nhập thành (castling): Nếu cả 2 phe không có quyền kí hiệu là “-“. Nếu có, “K” nghĩa là trắng có quyền nhập thành bên phía king, “Q” nghĩa là trắng có quyền nhập thành bên phía queen., “k” nghĩa là đen có quyền nhập thành bên phía king, “q” nghĩa là đen có quyền nhập thành bên phía queen.
* Nếu không có ô nào là mục tiêu en passant thì kí hiệu là “-“. Nếu một quân pawn vừa đi tới 2 bước, đây này là ô phía sau quân pawn đó.
* Halfmove clock: số nửa lượt đã trôi qua từ lúc một quân pawn bị ăn hoặc tiến tới.
* Fullmove clock: Số lượt fullmove kể từ đầu trận, bắt đầu bằng 1 và tăng 1 sau lượt của quân đen.



Hình : Chuỗi FEN khởi đầu mặc định của mối ván cờ

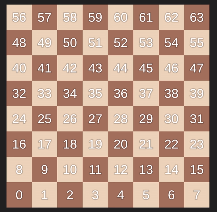
* Từ một chuỗi FEN, chường trình sẽ chia nó ra thành 5 phần tương ứng để xử lí.



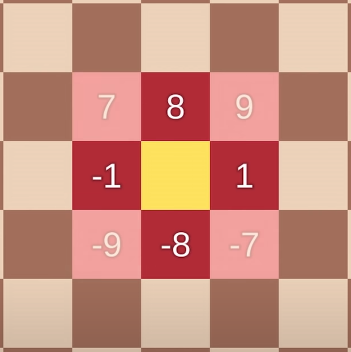
Hình : Cách load vị trí các quân cờ từ FEN

## Tìm kiếm nước đi hợp lệ (Move Generation):

### Cách di chuyển:

* Bàn cờ có thể được xem như là một mảng 1 chiều gồm 64 phần tử.

Hình : Index của bàn cờ

* Từ một ô, một quân cờ có thể đi đến một ô khác bằng cách thêm một giá trị hướng đi.
* Một quân pawn có thể đi tới bằng cách lấy vị trí hiện tại cộng với 8.

Hình : Cách các quân cờ đi chuyển

* Một quân bishop có thể đi chéo bằng cách lấy vị trí hiện tại cộng 9.



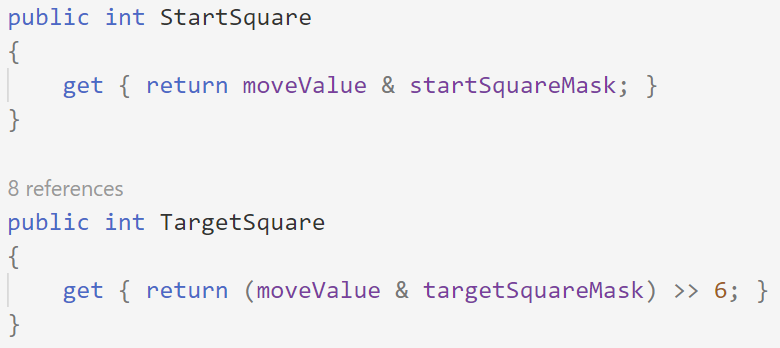
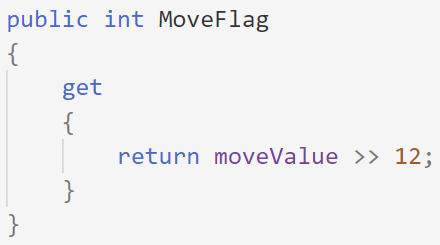
Hình 9: Hướng di chuyển được lưu dưới dạng mảng

### Nước đi (Move):

Hình : Thông tin nước đi

* Dữ liệu của nước đi được xử lí dưới dạng bit. Cụ thể là ushort 16bit.
* Sáu bit đầu tiên lưu điểm xuất phát của nước đi, 6 bit tiếp theo lưu đích đến của nước đi, 4 bit cuối cùng lưu loại nước đi(nhập thành/thăng cấp/en passant).

#### Cách xử lí dữ liệu:



Hình 12: Cách lấy loại nước đi từ moveValue

Hình 11: Cách lấy StartSquare và TargetSquare từ moveValue

### Move Generator:



Hình : Tìm kiếm các nước đi hợp lệ cho tất cả các quân cờ

* Lớp MoveGenerator có nhiệm vụ tìm ra các nước đi hợp lệ từ một thế cờ cho trước.
* Đầu tiên, nó sẽ tính toán các vị trí mà quân địch có thể tấn công.
* Các vị trí mà một quân cờ có thể tấn công được lưu dưới dạng bitboard.
* Bitboard: cách lưu trữ một bàn cờ dưới dạng bit với mỗi bit là một ô trên bàn cờ. Bit nào có giá trị bằng 1 nghĩa là nó đang bị tấn công.
* Vì bàn cờ vua có 64 ô nên kiểu dữ liệu ulong sẽ được sử dụng.



Hình : Các bitboard lưu dữ liệu tấn công của đối phương

* Giả sử ô thứ 7 đang bị tấn công ta có thể thêm ô thứ 7 vào mộtbằng cách:

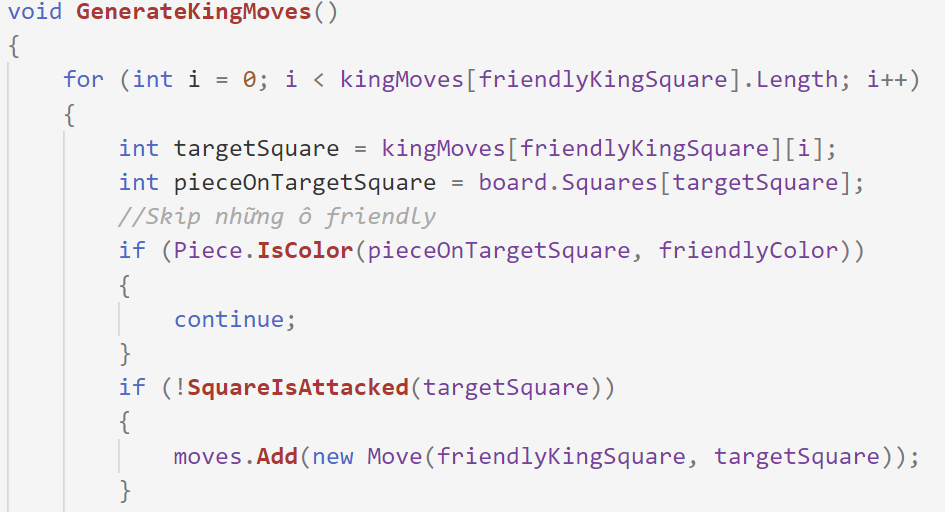


Hình : Cách thêm 1 vị trí vào bitboard

#### Các bước tìm kiếm nước đi hợp lệ cho King:

##### Lặp đến các ô mà King có thể đi tới:

* Nếu ô đi tới không bị tấn công thì thêm nước đi tới ô đó vào danh sách nước đi hợp lệ.
* Kiểm tra ô đi tới có phải ô nhập thành không. Nếu phải thì thêm nước đi nhập thành.



Hình : Khái quát tìm nước đi hợp lệ cho King

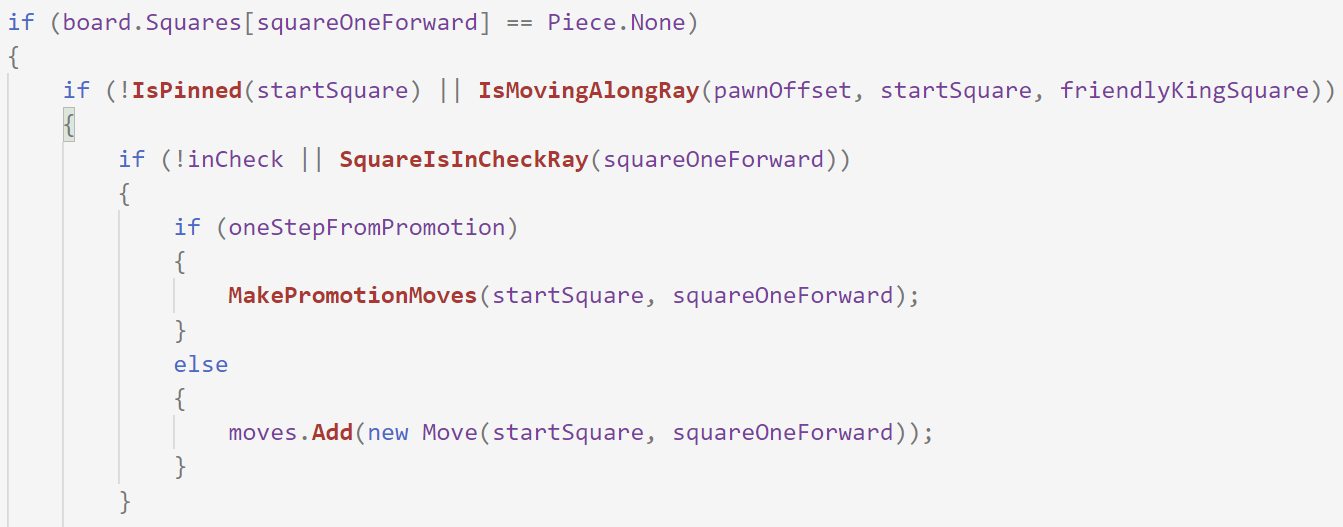
#### 

Hình : Thêm nước đi nhập thành

#### Các bước tìm kiếm nước đi hợp lệ cho Pawn:

##### Lặp lần lượt tất cả các ô mà các quân pawn đồng minh đang chiếm giữ:

* Nếu ô hiện tại đang bị pin thì bỏ qua.
* Nếu ô phía trước nó không có quân nào thì thêm nước đi tới 1 ô.
* Nếu là nước đi Promotion thì thêm nước đi promotion.
* Nếu nó đang ở vị trí bắt đầu và 2 ô phía trước không có quân nào thì thêm nước đi tới 2 ô.

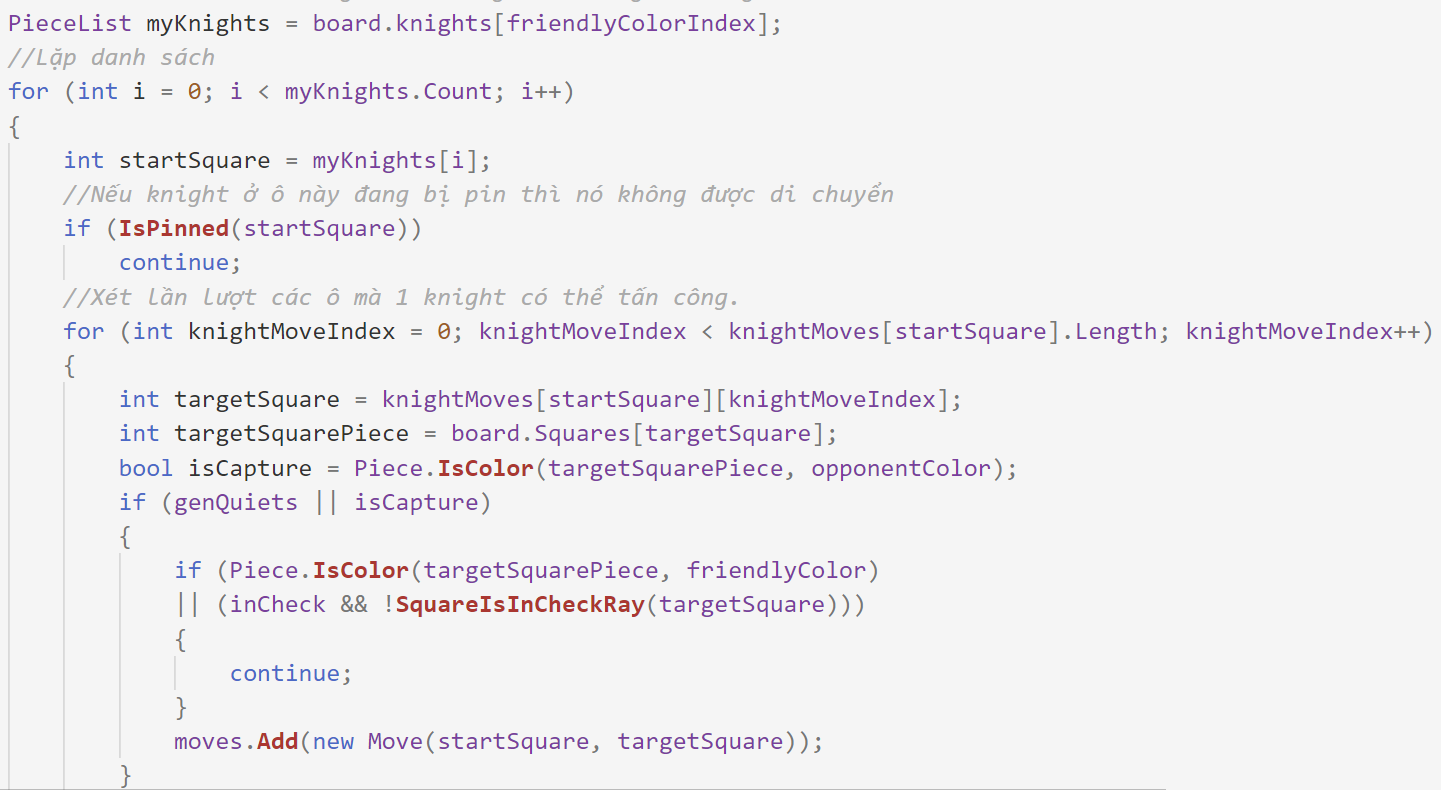


Hình : Tìm nước đi cho Pawn

#### Các bước tìm kiếm nước đi hợp lệ cho Knight:

##### Lặp lần lượt tất cả các ô mà các quân knight đồng minh đang chiếm giữ:

* Nếu ô hiện tại đang bị pin thì bỏ qua.
* Xét lần lượt các ô mà knight ở ô này có thể đi tới:
* Nếu ô đi tới trống hoặc không phải là quân đồng minh thì thêm nước đi vào danh sách.



Hình : Tìm nước đi cho Knight

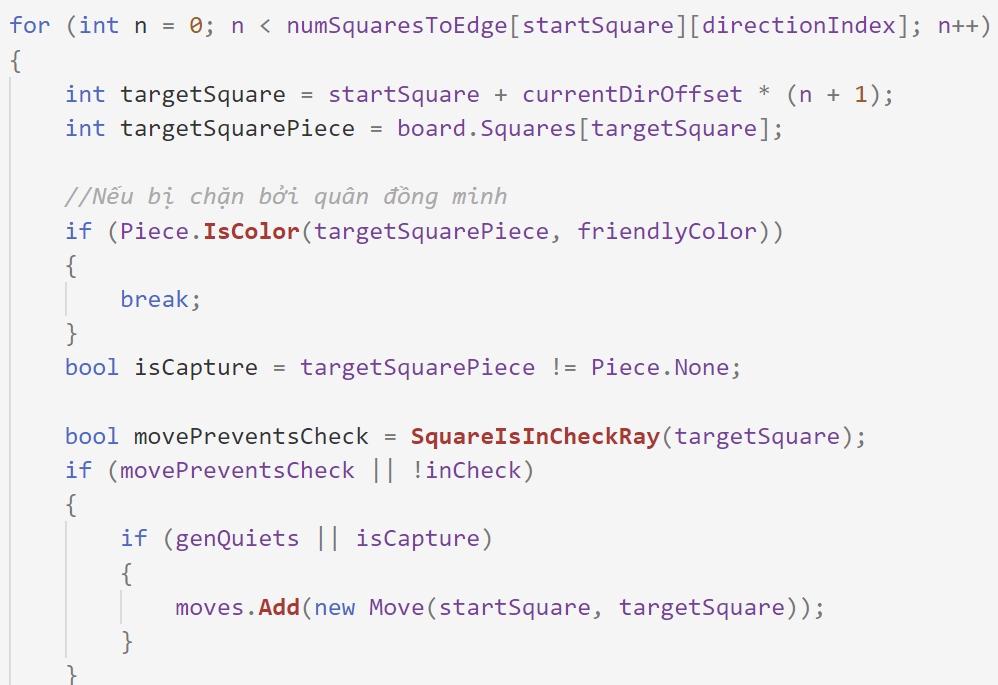
#### Các bước tìm kiếm nước đi hợp lệ cho Sliding Pieces(Bishop, Rook, Queen):

Các quân bishop, rook, queen có thể dùng chung 1 phương thức để tìm các nước đi hợp lệ. 

Hình : Tìm kiếm các nước đi hợp lệ cho Rook,Bishop,Queen

* Phương thức này sẽ dò tìm từ vị trí của một quân cờ, sau đó sẽ đi theo các hướng mà quân cờ này có thể di chuyển tới.
* Nếu ô hiện tại đang bị pin thì bỏ qua.
* Nếu ô hiện tại bị pin nhưng ô nó đi tới nằm trên đường pin thì có thể tiếp tục xét các điều kiện tiếp theo.
* Nếu ô đi tới trống hoặc nằm trên đường chiếu tướng của quân địch thì có thể thêm nước đi vào danh sách.

Hình : Các đường có quân bị pinned



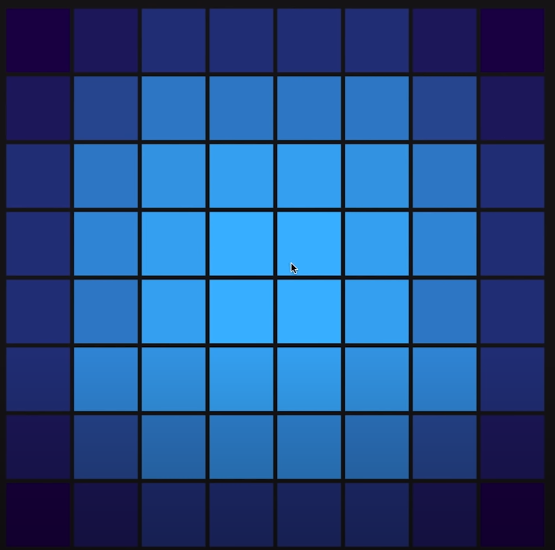
Hình : Tìm nước đi hợp lệ teo hướng di chuyển

## 

Hình : Tìm nước đi hợp lệ cho Sliding Pieces

## Tìm kiếm và hàm đánh giá (Search and Evaluation):

### Evaluation:

* Thực hiện đánh giá tĩnh thế cờ hiện tại.
* Giá trị trả về thể hiện lợi thế hoặc bất lợi của phe đang đến lượt.
* Nếu giá trị trả về < 0 có nghĩa là thực hiện nước đi sẽ mang đến bất lợi và ngược lại.

#### Các tiêu chí đánh giá:

* Tổng giá trị của các quân cờ của 2 phe.
* Mop-up: khi ván cờ đã đến end-game, không còn quân tốt nào trên sân, các nước cờ dồn King của đối phương vào góc của bàn cờ sẽ được cộng điểm.
* Piece-Square table: mỗi ô trên bàn cờ có giá trị khác nhau đối với từng loại quân cờ. Vì vậy các nước đi tới những ô này sẽ được cộng điểm.

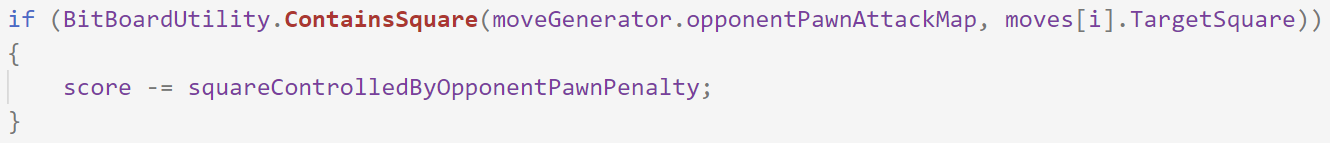
Hình : Knight Piece-Square table

### Move Ordering:

* Trước khi thực hiện tìm kiếm ta nên sắp xếp các nước đi theo thứ tự giá trị giảm dần để đẩy nhanh quá trình cắt tỉa alpha beta.
* Ngoài ra, trước khi sắp xếp, các nước đi lấy quân yếu hơn ăn quân mạnh hơn như PxQ sẽ được cộng điểm, các nước Promotion cũng sẽ được cộng điểm, các nước đi vào ô mà quân Pawn của đối phương có thể tấn công sẽ bị trừ điểm.

Hình 23: Cộng điểm cho các nước rất tốt





Hình : Trừ điểm các nước đi vào ô quân pawn có thể tấn công

Hình : Sắp xếp các nước đi dựa trên score

### 

### Search:

Áp dụng giải thuật Minimax, chương trình sẽ liên tục thực hiện một nước và các con của nó. Cho đến khi đã đạt đến độ sâu nhất định, thế cờ sẽ được đánh giá và giá trị sẽ được sử dụng.



Phương thức tìm kiếm cũng giống với giải thuật minimax.

Với tham số truyền vào là độ sâu hiện tại, alpha, beta với một điểm khác biệt là tham số plyFromRoot – số nước đi của 2 phe kể từ thế cờ gốc.

# TÀI LIỆU THAM KHẢO

[How to Implement the Hill Climbing Algorithm in Python | by Hein de Haan | Towards Data Science](https://towardsdatascience.com/how-to-implement-the-hill-climbing-algorithm-in-python-1c65c29469de)

[Psychic Origami - Tackling the travelling salesman problem: hill-climbing](https://psychicorigami.com/2007/05/12/tackling-the-travelling-salesman-problem-hill-climbing/)

[(294) Travelling Salesman Problem using Hill Climbing in Python | Artificial Intelligence - YouTube](https://www.youtube.com/watch?v=d1lzvynySac&ab_channel=ThinkXAcademy)