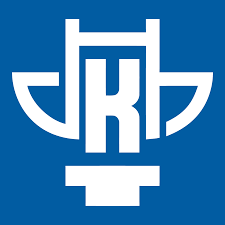
BỘ XÂY DỰNG

**TRƯỜNG ĐẠI HỌC KIẾN TRÚC HÀ NỘI**



**…**

**NGHIÊN CỨU KHOA HỌC**

**ĐỀ TÀI: NGHIÊN CỨU VÀ ĐÁNH GIÁ MỘT SỐ THUẬT TOÁN TRÍ TUỆ NHÂN TẠO ỨNG DỤNG XÂY DỰNG TRÒ CHƠI CỜ VUA**

**SINH VIÊN THỰC HIÊN: 1. Trần Trọng Nghĩa (NT)**

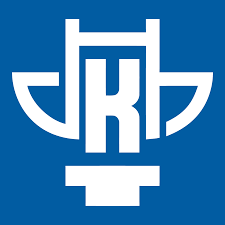
**2. Giáp Hoàng Việt Tiến**

**GIẢNG VIÊN HƯỚNG DẪN: ThS.Nguyễn Thị Nguyệt**

Hà Nội, 5/2024

BỘ XÂY DỰNG

**TRƯỜNG ĐẠI HỌC KIẾN TRÚC HÀ NỘI**



**…**

**NGHIÊN CỨU KHOA HỌC**

**ĐỀ TÀI: NGHIÊN CỨU VÀ ĐÁNH GIÁ MỘT SỐ THUẬT TOÁN TRÍ TUỆ NHÂN TẠO ỨNG DỤNG XÂY DỰNG TRÒ CHƠI CỜ VUA**

**SINH VIÊN THỰC HIÊN: 1. Trần Trọng Nghĩa (NT)**

**2. Giáp Hoàng Việt Tiến**

**GIẢNG VIÊN HƯỚNG DẪN: ThS.Nguyễn Thị Nguyệt**

Hà Nội, 5/2024

**MỤC LỤC**

[MỤC LỤC 3](#_Toc167228314)

[LỜI MỞ ĐẦU 5](#_Toc167228315)

[DANH MỤC TỪ VIẾT TẮT 6](#_Toc167228316)

[HÌNH ẢNH DANH MỤC HÌNH ẢNH 7](#_Toc167228317)

[CHƯƠNG 1: CƠ SỞ LÝ THUYẾT 8](#_Toc167228318)

[1.1 Tổng quan về trí tuệ nhân tạo 8](#_Toc167228319)

[1.1.1 Khái niệm trí tuệ nhân tạo 8](#_Toc167228320)

[1.1.2 Phân loại trí tuệ nhân tạo 8](#_Toc167228321)

[1.1.3 Ứng dụng của trí tuệ nhân tạo 10](#_Toc167228322)

[1.1.4 Mặt tích cực và hạn chế của trí tuệ nhân tạo. 11](#_Toc167228323)

[1.2 Tổng quan về đề tài 11](#_Toc167228324)

[1.2.1. Lịch sử phát triển của cờ vua. 11](#_Toc167228325)

[1.2.2. Quy tắc cơ bản trong cờ vua. 12](#_Toc167228326)

[1.2.3 Input và Output của trò chơi cờ vua. 13](#_Toc167228327)

[1.3 Lựa chọn ngôn ngữ và công nghệ cho đề tài 14](#_Toc167228328)

[1.3.1 Ngôn ngữ lập trình python 14](#_Toc167228329)

[1.3.2 Thư viện sử dụng 14](#_Toc167228330)

[CHƯƠNG 2: THUẬT TOÁN ÁP DỤNG VÀO TRÒ CHƠI CỜ VUA 15](#_Toc167228331)

[2.1 Thuật toán Naive Algorithm 15](#_Toc167228332)

[2.1.1 Khái niệm 15](#_Toc167228333)

[2.1.2 Một số đặc điểm của thuật toán Naive Algorithm: 15](#_Toc167228334)

[2.1.3 Áp dụng vào đề tài 15](#_Toc167228335)

[2.2 Thuật toán Greedy algorithm 17](#_Toc167228336)

[2.2.1 Khái niệm 17](#_Toc167228337)

[2.2.2 Mã giả 17](#_Toc167228338)

[2.2.3 Áp dụng vào đề tài 18](#_Toc167228339)

[2.3 Thuật toán MiniMax 20](#_Toc167228340)

[2.3.1 Khái niệm 20](#_Toc167228341)

[2.3.2 Mã giả 20](#_Toc167228342)

[2.3.3 Áp dụng vào đề tài 21](#_Toc167228343)

[2.4. Thuật toán negamax chính xác với cắt alpha-beta 22](#_Toc167228344)

[2.4.1 Khái niệm 22](#_Toc167228345)

[2.4.2 Mã giả 23](#_Toc167228346)

[2.4.3 Áp dụng vào đề tài 24](#_Toc167228347)

[CHƯƠNG 3: PHÁT TRIỂN VÀ TRIỂN KHAI ỨNG DỤNG 28](#_Toc167228348)

[3.1 Cài đặt các thư viện cần thiết 28](#_Toc167228349)

[3.2 Huấn luyện các nước đi hợp lệ 28](#_Toc167228350)

[3.2.1 Huấn luyện cho con tốt 28](#_Toc167228351)

[3.2.2 Huấn luyện con xe 31](#_Toc167228352)

[3.2.3 Huấn luyện cho con mã 32](#_Toc167228353)

[3.2.4 Huấn luyện cho con tượng 33](#_Toc167228354)

[3.2.5 Huấn luyện cho con vua 35](#_Toc167228355)

[3.2.6 Huấn luyện cho con hậu 36](#_Toc167228356)

[3.2.7 Huấn luyện con vua nhập thành 36](#_Toc167228357)

[3.3 Giao diện trò chơi 38](#_Toc167228358)

[3.4 Các bước khởi chạy chương trình 39](#_Toc167228359)

[KẾT LUẬN 42](#_Toc167228360)

[TÀI LIỆU THAM KHẢO 43](#_Toc167228361)

**LỜI MỞ ĐẦU**

Trong thời đại số hóa và công nghệ thông tin phát triển mạnh mẽ như hiện nay, trò chơi không chỉ đơn giản là một phương tiện giải trí, mà còn là một lĩnh vực đa dạng và phức tạp, nơi mà khoa học máy tính và trí tuệ nhân tạo đã tạo ra những cơ hội mới cho sáng tạo và nghiên cứu. Trong bối cảnh này, chúng ta không thể bỏ qua trò chơi cờ vua, một trò chơi kỳ diệu với hơn 1500 năm lịch sử và vẫn đang thu hút hàng triệu người trên khắp thế giới.

Nghiên cứu này ra đời với mục tiêu kết hợp hai phương pháp quan trọng trong lĩnh vực trí tuệ nhân tạo và trò chơi - thuật toán Minimax và thuật toán Greedy - để xây dựng một trò chơi cờ vua có trí thông minh cao và có khả năng thách thức người chơi.

Thuật toán Minimax đã được áp dụng rộng rãi trong trò chơi cờ vua để tạo ra một hệ thống ra quyết định tốt nhất dựa trên việc dự đoán các bước tiếp theo của đối thủ và lựa chọn bước đi tốt nhất cho mình. Trong khi đó, thuật toán tham lam được sử dụng để xác định các chiến lược tối ưu trong một tình huống cụ thể, dựa trên lợi ích ngắn hạn. Khi kết hợp cả hai thuật toán này, chúng ta có thể tạo ra một trò chơi cờ vua độc đáo với khả năng tư duy tối ưu và kỹ thuật chơi tâm linh.

Nghiên cứu này sẽ trình bày quá trình phát triển và triển khai trò chơi cờ vua dựa trên các nguyên tắc của thuật toán Minimax và thuật toán tham lam. Chúng ta sẽ xem xét cách áp dụng các khái niệm toán học và lý thuyết trò chơi vào thiết kế và phát triển trò chơi, cũng như các thách thức và cơ hội trong quá trình này.

Bên cạnh việc tạo ra một trò chơi cờ vua thông minh, nghiên cứu này còn mục tiêu khám phá và đánh giá hiệu suất của thuật toán Minimax và thuật toán tham lam trong ngữ cảnh của trò chơi cờ vua. Chúng ta sẽ nghiên cứu cách mà hai thuật toán này đối phó với các tình huống khác nhau trong trò chơi và xem xét cách chúng ảnh hưởng đến quyết định của người chơi và chất lượng trải nghiệm trò chơi.

Chúng ta cũng sẽ xem xét các khía cạnh thú vị khác của việc kết hợp trí tuệ nhân tạo và trò chơi, chẳng hạn như khả năng học tập của hệ thống trong quá trình chơi cờ vua. Làm thế nào máy tính có thể cải thiện hiệu suất của mình dựa trên kinh nghiệm trước đây? Làm thế nào nó có thể đánh giá và thích nghi với chiến lược của người chơi?

Cuối cùng, chúng ta hi vọng rằng nghiên cứu này sẽ giúp mở rộng kiến thức và tạo ra những trải nghiệm thú vị cho cộng đồng yêu thích cờ vua và trí tuệ nhân tạo. Chúng ta mong đợi rằng thông qua việc kết hợp sâu sắc giữa lý thuyết và thực tiễn, nghiên cứu này sẽ đóng góp vào sự phát triển của trò chơi cờ vua và nghiên cứu về trí tuệ nhân tạo, cũng như tạo ra cơ hội cho những ứng dụng tiềm năng trong các lĩnh vực khác của cuộc sống.

# DANH MỤC TỪ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| STT | Từ viết tắt | Giải thích |
| 1 | AI | **Artificial Intelligence** |
| 2 | ANI | Trí tuệ nhân tạo hẹp |
| 3 | AGI | Trí tuệ nhân tạo chung |
| 4 | ASI | Siêu trí tuệ nhân tạo |
| 5 | IBM | International Business Machines |
| 6 | LSTM | Long Short-Term Memory |

# 

# HÌNH ẢNH DANH MỤC HÌNH ẢNH

[Hình 3.3. 1: Hỉnh ảnh giao diện cờ vua 39](#_Toc167228052)

[Hình 3.3. 2: Hình ảnh phần mềm pycharm 39](#_Toc167228053)

[Hình 3.3. 3: Hình ảnh lưu file chương trình 40](#_Toc167228054)

[Hình 3.3. 4: Hình ảnh lựa chọn file ChessMain và bấm vào run để chạy chương trình 41](#_Toc167228055)

[Hình 3.3. 5: Hình ảnh hoàn tất hiển thị chương trình 41](#_Toc167228056)

**CHƯƠNG 1: CƠ SỞ LÝ THUYẾT**

## 1.1 Tổng quan về trí tuệ nhân tạo

### 1.1.1 Khái niệm trí tuệ nhân tạo

Trí tuệ nhân tạo (**Artificial Intelligence**) là một lĩnh vực nghiên cứu của khoa học máy tính và khoa học tính toán nói chung. Có nhiều quan điểm khác nhau về trí tuệ nhân tạo và do vậy có nhiều định nghĩa khác nhau về lĩnh vực khoa học này.

Mục đích của trí tuệ nhân tạo là xây dựng các thực thể thông minh. Tuy nhiên, do rất khó định nghĩa thế nào là thực thể thông minh nên cũng khó thống nhất định nghĩa trí tuệ nhân tạo. Theo một tài liệu được sử dụng rộng rãi trong giảng dạy trí tuệ nhân tạo hiện nay, các định nghĩa có thể nhóm thành bốn nhóm khác nhau, theo đó, trí tuệ nhân tạo là lĩnh vực nghiên cứu việc xây dựng các hệ thống máy tính có đặc điểm sau:

1) Hệ thống hành động như người.

2) Hệ thống có thể suy nghĩ như người

3) Hệ thống có thể suy nghĩ hợp lý

4) Hệ thống hành động hợp lý

### 1.1.2 Phân loại trí tuệ nhân tạo

Trí tuệ nhân tạo được phân loại dựa vào hai yếu tố:

- Phân loại dựa trên sự phát triển, thành thạo của chúng.

- Phân loại dựa trên sự tương đồng của AI với trí tuệ con người, khả năng suy nghĩ hay có cảm nhận thực sự như con người.

Dựa vào yếu tố đầu tiên, ta có thể phân loại AI thành 3 loại:

Artificial Narrow Intelligence (ANI - Trí tuệ nhân tạo hẹp): còn có tên gọi khác là Weak AI, là một loại AI được lập trình để thực hiện một hoặc vài nhiệm vụ cụ thể, không thể so sánh với khả năng nhận thức tổng thể của con người. Weak AI tập trung vào công việc cụ thể và có thể thay thế hoặc vượt trội con người trong nhiệm vụ đó. Tuy nhiên, do chỉ được lập trình cho một công việc cụ thể, loại AI này chỉ có thể xử lý dữ liệu dựa trên thông tin đã được huấn luyện và không thể thực hiện các nhiệm vụ mà nó không được đào tạo. Mặc dù tên gọi "Weak AI" có thể khiến ta nghĩ rằng nó có khả năng hạn chế, nhưng thực tế, đây là loại AI phổ biến và được sử dụng rộng rãi trong thực tế. Các ứng dụng của nó bao gồm trợ lý ảo như Siri, hệ thống đề xuất, chatbot, xe tự lái, nhận diện khuôn mặt, hình ảnh và giọng nói.

Artificial General Intelligence (AGI - Trí tuệ nhân tạo chung): là một dạng của AI có trí thông minh tương đương với con người. Điều này có nghĩa là AGI có khả năng tự nhận thức, giải quyết vấn đề, học hỏi và lập kế hoạch cho tương lai mà không cần sự huấn luyện trước. So với Trí tuệ Nhân tạo Hẹp (ANI), AGI phức tạp và hiệu suất cao hơn nhiều. Tuy nhiên, cho đến nay, chúng ta vẫn chưa thể tạo ra một máy tính thực sự có khả năng AGI, và AGI vẫn chỉ là một lý thuyết trên giấy. Tuy nhiên, tiến bộ đã được đạt được và có thể tạo ra một dạng gọi là AGI một phần.

Artificial Super Intelligence (ASI - Siêu trí tuệ nhân tạo): là một loại AI vượt xa cả Trí Tuệ Nhân Tạo Hẹp (Weak AI) và Trí Tuệ Nhân Tạo Tổng hợp (AGI). ASI có khả năng tự nhận thức đủ để vượt qua khả năng của não bộ và hành vi của con người. Điểm mạnh của ASI nằm ở trí nhớ, xử lý dữ liệu, và khả năng ra quyết định, đều vượt trội hơn con người do sử dụng một mạng lưới neuron hệ số lượng lớn hơn rất nhiều so với não bộ con người. Tuy nhiên, hiện tại, việc tạo ra ASI vẫn chỉ tồn tại trong tưởng tượng và trong tác phẩm viễn tưởng, và con đường để phát triển ASI còn rất xa, ít nhất là trước khi có thể tạo ra AGI.

Trong đó, AGI và ASI đều là Strong AI. Vì vậy, cũng có thể nói dựa trên mức độ thông minh, ta có thể chia AI thành 2 loại là Narrow AI và Strong AI.

Dựa vào yếu tố còn lại, người ta chia AI thành 4 loại:

Reactive Machines (AI phản ứng): là loại AI cổ điển nhất, chỉ phản ứng theo hiện tại và không có khả năng ghi nhớ hay học từ trải nghiệm. Mặc dù có khả năng phản ứng với môi trường xung quanh giống như trí não con người, nhưng nó chỉ có khả năng giới hạn trong việc phản ứng ngay tại thời điểm đó. Deep Blue, một ví dụ, là một máy tính AI phản ứng có khả năng chơi cờ vua và đã đánh bại tướng cờ vua Gary Kasparov trong một trận đấu. Tuy không thể học hay cải thiện sau thời gian, nhưng nó có khả năng nhận biết từng nước cờ và luật chơi để đưa ra những nước cờ tốt nhất.

Limited Memory (AI với trí nhớ giới hạn): vượt trội hơn so với AI phản ứng bằng khả năng lưu trữ dữ liệu và dự đoán từ trước, cho phép nó phân tích dữ liệu và học hỏi. Loại AI này thường được sử dụng trong các mô hình Học máy như Reinforcement Learning, Long Short-Term Memory (LSTM), và E-GAN. Hiện tại, đây là loại AI phổ biến nhất và được áp dụng rộng rãi trong các ứng dụng như trợ lý ảo giọng nói, chatbot, xe tự lái và nhận diện khuôn mặt.

Theory of Mind (Lý thuyết về tâm trí): trí thông minh của con người không chỉ xuất hiện trong quá trình ra quyết định và giải quyết vấn đề, mà còn bao gồm khả năng suy nghĩ và cảm nhận, ảnh hưởng đến hành vi của chúng ta. Với khả năng sống trong môi trường xã hội, con người không thể tồn tại độc lập. Do đó, Trí tuệ Nhân tạo cần có khả năng mô phỏng cách con người đưa ra quyết định xã hội, điều này là điểm đặc biệt phân biệt con người với các loài khác. Chúng ta có khả năng đoán định tâm trí của người khác (mặc dù không hoàn toàn chính xác) để xây dựng và duy trì các mối quan hệ, giao tiếp hiệu quả và hợp tác để đạt được mục tiêu chung.AI Tâm trí phát triển hơn so với hai loại AI trước bằng khả năng áp dụng lý thuyết về tâm trí và hiểu được cảm xúc và tâm lý của con người. Tâm trí con người rất phức tạp và thay đổi không dự đoán được. Để hiểu tâm trí, AI cần nắm vững các khái niệm tâm lý học, cảm thông với con người và hiểu cách con người nhận thức bản thân. Tuy nhiên, vì sự phức tạp này, AI Tâm trí hiện vẫn chỉ còn mức lý thuyết và cần sự phát triển ở nhiều lĩnh vực liên quan đến Trí tuệ Nhân tạo.

Seft-Awareness (Tự nhận thức): AI tự nhận thức đại diện cho giai đoạn phát triển cuối cùng của Trí tuệ Nhân tạo. Trong khi AI Tâm trí có khả năng hiểu tâm trí của con người, thì AI tự nhận thức sẽ phát triển và trải nghiệm cảm xúc riêng của nó, nhận thức và có ý thức về bản thân. Nó sẽ có nhu cầu, niềm tin, cảm xúc, và khả năng phát triển ý thức, nhận thức về sự tồn tại của bản thân trên thế giới này.

AI tự nhận thức là một chủ đề thường xuất hiện trong các tác phẩm khoa học viễn tưởng, và với khả năng có ý thức, loại AI này có thể có ý muốn tự bảo vệ bản thân tương tự như con người. May mắn là cho đến hiện tại, việc phát triển AI tự nhận thức vẫn còn rất xa và đang ở trong lĩnh vực của tưởng tượng.

### 1.1.3 Ứng dụng của trí tuệ nhân tạo

Trí tuệ nhân tạo (AI - Artificial Intelligence) đã có một loạt ứng dụng rộng rãi trong nhiều lĩnh vực của cuộc sống và công nghiệp. Dưới đây là một số ứng dụng phổ biến của trí tuệ nhân tạo:

* Xử lý Ngôn ngữ Tự nhiên:
* Chatbot: Tạo ra các trò chuyện tự động để hỗ trợ khách hàng, cung cấp thông tin hoặc giải đáp câu hỏi.
* Dịch thuật tự động: Dịch văn bản hoặc lời nói từ một ngôn ngữ sang ngôn ngữ khác một cách tự động.
* Phân tích tình cảm: Phát hiện và phân tích tình cảm trong văn bản, ví dụ: phát hiện cảm xúc trong mạng xã hội.
* Thị giác máy tính:
* Nhận dạng khuôn mặt: Xác định và nhận dạng khuôn mặt trong hình ảnh và video.
* Nhận dạng đối tượng: Phát hiện và xác định đối tượng trong hình ảnh và video, ví dụ: xe hơi, động vật, đồ vật.
* Xử lý hình ảnh y tế: Phân tích hình ảnh y tế để chẩn đoán và dự đoán bệnh.
* Ô tô tự hành (Self-driving Cars): Sử dụng AI để điều khiển và quản lý các hệ thống tự động trên ô tô, giúp xe tự lái di chuyển an toàn.
* Chẩn đoán y tế: Sử dụng dữ liệu y tế và học máy để đưa ra các chuẩn đoán y khoa, dự đoán bệnh và tối ưu hóa điều trị.
* Hệ thống gợi ý: Cung cấp gợi ý sản phẩm, nội dung hoặc dịch vụ dựa trên dữ liệu và hành vi người dùng trước đây.
* Quản lý chuỗi cung ứng và dự báo: Sử dụng AI để tối ưu hóa quy trình sản xuất, dự đoán nhu cầu và lập kế hoạch vận chuyển.
* Trò chơi và Giải đố: Sử dụng AI để tạo lối chơi đối thủ thông minh và thú vị trong trò chơi điện tử và giải đố.
* Tư vấn tài chính và giao dịch: Sử dụng AI để phân tích thị trường tài chính, dự đoán biến động giá cả, và thậm chí thực hiện các giao dịch tài chính tự động.
* Hỗ trợ quyết định trong kinh doanh: Cung cấp thông tin quan trọng và dự đoán cho doanh nghiệp để họ có thể đưa ra quyết định thông minh.
* Giáo dục và Học tập: Sử dụng AI để cung cấp tài liệu giáo dục cá nhân hóa, đánh giá và hỗ trợ học tập trực tuyến.

Những ứng dụng của trí tuệ nhân tạo đang liên tục mở rộng và phát triển trong nhiều lĩnh vực, và chúng có tiềm năng để cải thiện chất lượng cuộc sống và hiệu suất trong công việc.

### 1.1.4 Mặt tích cực và hạn chế của trí tuệ nhân tạo.

Công nghệ AI đã và đang được nghiên cứu một cách sâu rộng nhất là các nước phát triển trên thế giới. Vì vậy, trí tuệ nhân tạo mang đến nhiều tích cực cho cuộc sống con người và xã hội. AI xử lý được lượng lớn dữ liệu nhanh hơn con người và đưa ra dự đoán chính xác hơn khả năng của con người. Khối lượng dữ liệu khổng lồ được tạo ra hàng ngày gây nhiều khó khăn cho các nhà nghiên cứu, AI sử dụng học máy để có thể lấy những dữ liệu đó và nhanh chóng biến nó thành thông tin có thể thực hiện được.

Bên cạnh những mặt tích cực mà trí tuệ nhân tạo mang lại thì AI cũng có những hạn chế nhất định. Việc sử dụng trí tuệ nhân tạo là tốn kém rất nhiều khi sử dụng một lượng lớn dữ liệu mà người lập trình AI yêu cầu. Vì công nghệ này còn mới lạ và vô cùng phát triển, nên không phải tại quốc gia nào cũng có thể áp dụng sử dụng và nếu có sử dụng AI trong nhiều lĩnh vực thì phải tuân thủ quy định một cách nghiêm ngặt.

## 1.2 Tổng quan về đề tài

### 1.2.1. Lịch sử phát triển của cờ vua.

Cờ vua (Chess) là một môn thể thao trí óc nổi tiếng trên thế giới. Cờ vua có một lịch sử lâu đời và phát triển qua các giai đoạn quan trọng như sau:

Thế kỷ 6-7: Cờ vua bắt đầu có nguồn gốc từ Ấn Độ, được gọi là "Chaturanga." Ban đầu, trò chơi này được chơi trên bàn cờ 8x8 giống như ngày nay, với các quân cờ tượng trưng cho các loại quân lính trong quân đội Ấn Độ: Vua, Hậu (Rani), Xe (Ratha), Tượng (Gaja), Ngựa (Ashva), Tốt (Padàti), và Chốt (Ratha-pada). Trò chơi lan truyền đến Ba Tư và sau đó đến châu Âu qua thương mại và văn hóa.

Thế kỷ 10-15: Trò chơi cờ vua tiếp tục phát triển tại châu Âu. Quân cờ và luật chơi trở nên phức tạp hơn. Quân Hậu trở nên mạnh mẽ hơn và cung cấp tính cơ động lớn cho người chơi.

Thế kỷ 19: Luật chơi cờ vua được chuẩn hóa và định rõ như hiện nay. Trong thế kỷ này, cờ vua trở thành một môn thể thao phổ biến và phát triển mạnh mẽ. Các giải đấu quốc tế xuất hiện và người chơi nổi tiếng như Paul Morphy, Wilhelm Steinitz, và Emanuel Lasker xuất hiện.

Thế kỷ 20: Cờ vua trở thành một môn thể thao toàn cầu, với việc tổ chức các giải đấu quốc tế và thế giới. Những cái tên nổi tiếng như Bobby Fischer, Garry Kasparov và Magnus Carlsen đã xuất hiện và trở thành những huyền thoại của trò chơi này.

Hiện tại: Cờ vua không chỉ là một trò chơi truyền thống mà còn trở thành một trò chơi trực tuyến phổ biến. Các chương trình máy tính chơi cờ như Deep Blue và AlphaZero đã thách thức những tay cờ vua xuất sắc nhất thế giới.

Cờ vua vẫn tiếp tục thu hút người chơi trên toàn thế giới và được coi là một trò chơi tinh thần và chiến thuật kinh điển.

### 1.2.2. Quy tắc cơ bản trong cờ vua.

Cờ vua là một trò chơi chiến thuật hai người, nổi tiếng trên khắp thế giới. Trò chơi này thường được chơi trên một bàn cờ có kích thước 8x8 ô, được chia thành 64 ô có màu đen và trắng xen kẽ. Mỗi người chơi điều khiển một bên, một bên chơi với các quân cờ màu trắng, và bên kia chơi với các quân cờ màu đen. Mục tiêu của trò chơi là đặt Vua của đối phương vào tình trạng "bị chiếu bởi quân cờ khác" (chạm vào Vua đối phương), đồng thời phải tránh bị chiếu tương tự.

Dưới đây là một số quy tắc cơ bản trong cờ vua:

Quân cờ và cách di chuyển:

Vua: Di chuyển một bước trong bất kỳ hướng nào (ngang, dọc, chéo) trừ trường hợp di chuyển làm Vua bị chiếu.

Hậu (Queen): Di chuyển theo mọi hướng trên bàn cờ (ngang, dọc, chéo) một số bước không giới hạn.

Xe (Rook): Di chuyển theo đường ngang hoặc dọc một số bước không giới hạn.

Tượng (Bishop): Di chuyển theo đường chéo một số bước không giới hạn.

Mã (Knight): Di chuyển theo mô hình "L", bao gồm một bước theo một hướng và sau đó một bước theo hướng vuông góc.

Tốt (Pawn): Di chuyển một bước theo hướng về phía đối phương và bắt quân đối phương bằng cách di chuyển theo đường chéo. Tốt còn có thể di chuyển hai bước từ vị trí ban đầu.

Chiếu và cờ chặn (check and checkmate): Chiếu xảy ra khi Vua của một bên có khả năng bị bắt trong bước tiếp theo. Nếu Vua bị chiếu, người chơi phải di chuyển Vua hoặc bảo vệ Vua khỏi chiếu. Nếu không còn cách nào để bảo vệ Vua và Vua vẫn bị chiếu, trò chơi kết thúc với tình trạng "checkmate" (bị chiếu bại), và người chơi chiếu bên còn lại thắng

Hòa (draw): Trò chơi cờ vua có thể kết thúc bằng trạng thái hòa, như hòa cờ, hòa theo luật ba lần, hòa bằng bất khả kháng, và nhiều quy tắc khác.

Cờ vua là một trò chơi chiến thuật phức tạp, yêu cầu người chơi phải có sự tư duy chiến thuật, dự đoán và lập kế hoạch tốt để chiến thắng đối thủ. Trò chơi này đã tồn tại hàng ngàn năm và vẫn rất phổ biến và thu hút nhiều người trên toàn thế giới.

### 1.2.3 Input và Output của trò chơi cờ vua.

Trò chơi cờ vua thường không có một giao diện máy tính cố định, mà thay vào đó có rất nhiều ứng dụng, trang web, và phần mềm cho phép bạn chơi cờ vua trực tuyến. Input và output trong trò chơi cờ vua thường bao gồm:

Input (Đầu vào):

Chọn quân cờ: Người chơi sẽ chọn quân cờ mà họ muốn di chuyển bằng cách nhấp vào hoặc di chuyển con trỏ đến quân cờ đó.

Lựa chọn nước đi: Sau khi chọn quân cờ, người chơi sẽ chọn ô mục tiêu mà họ muốn di chuyển quân cờ đến bằng cách nhấp vào ô đó.

Nếu có các tình huống đặc biệt: Trong một số tình huống, như nếu bạn muốn quân tốt di chuyển hai bước từ vị trí ban đầu hoặc bạn muốn biến quân tốt thành một quân cờ khác sau khi nó đến hàng cuối bàn cờ, bạn cần phải cung cấp đầu vào phù hợp.

Output (Đầu ra):

Hiển thị nước đi: Sau khi người chơi đã chọn quân cờ và ô mục tiêu, hệ thống sẽ hiển thị nước đi này bằng cách di chuyển quân cờ đó đến ô mục tiêu.

Thông báo chiếu: Nếu một bên đặt Vua của đối phương vào tình trạng chiếu, hệ thống thường thông báo chiếu và có thể hiển thị thông báo cho người chơi về tình trạng này.

Thông báo kết thúc trò chơi: Trò chơi có thể kết thúc với các tình huống như chiếu bí bài, hòa cờ hoặc chiếu bí bài không cách nào cản trở. Trong trường hợp này, hệ thống thông báo kết quả trận đấu, ví dụ: "Người chơi trắng thắng," "Hòa cờ," hoặc "Chiếu bí bài."

Trò chơi cờ vua trực tuyến thường cung cấp một giao diện đồ họa cho người chơi, cho phép họ thực hiện các hành động như chọn quân cờ và di chuyển chúng thông qua cửa sổ hoặc trang web. Hệ thống sẽ tự động kiểm tra luật và hiển thị kết quả của nước đi, giúp người chơi thực hiện trận đấu một cách thuận tiện.

## 1.3 Lựa chọn ngôn ngữ và công nghệ cho đề tài

### 1.3.1 Ngôn ngữ lập trình python

Python là [ngôn ngữ lập trình máy tính bậc cao](https://glints.com/vn/blog/ngon-ngu-lap-trinh-bac-cao/) thường được sử dụng để xây dựng trang web và phần mềm, tự động hóa các tác vụ và tiến hành [phân tích dữ liệu](https://glints.com/vn/blog/ky-nang-phan-tich-du-lieu/). Python là ngôn ngữ có mục đích chung, nghĩa là nó có thể được sử dụng để tạo nhiều chương trình khác nhau và không chuyên biệt cho bất kỳ vấn đề cụ thể nào.

Tính linh hoạt này, cùng với sự thân thiện với người mới bắt đầu, đã khiến nó trở thành một trong những ngôn ngữ lập trình được sử dụng nhiều nhất hiện nay. Một cuộc khảo sát được thực hiện bởi công ty phân tích ngành RedMonk cho thấy rằng đây là ngôn ngữ lập trình phổ biến thứ hai đối với các [nhà phát triển](https://glints.com/vn/blog/developer-la-gi/) vào năm [2021](https://redmonk.com/sogrady/2021/08/05/language-rankings-6-21/).

### 1.3.2 Thư viện sử dụng

1. Thư viện pygame

PyGame là một thư viện game mạnh mẽ và phổ biến cho Python. Nó cung cấp một bộ công cụ cùng các chức năng giúp việc tạo game và ứng dụng tương tác khác dễ dàng. Một tính năng hữu ích của PyGame là vẽ đối tượng, hình dạng nhất định bạn nên thử.

Pygame là một thư viện Python được sử dụng để phát triển các ứng dụng và trò chơi 2D. Nó cung cấp các công cụ và chức năng để tạo ra các ứng dụng đa phương tiện, đặc biệt là trò chơi 2D. Dưới đây là một số điểm quan trọng về thư viện Pygame:

Điều khiển cửa sổ và sự kiện: Pygame cung cấp các công cụ để quản lý cửa sổ, nhận các sự kiện như nhấn phím và click chuột, điều này giúp bạn tương tác với người chơi hoặc người dùng của ứng dụng.

Quản lý hình ảnh và âm thanh: Bạn có thể tải và quản lý hình ảnh, âm thanh, và các tài nguyên khác cho trò chơi của mình bằng Pygame. Điều này giúp bạn dễ dàng làm việc với các tài nguyên này trong ứng dụng của mình.

Sử dụng dễ dàng: Pygame là một thư viện dễ sử dụng cho người mới bắt đầu, đặc biệt là cho việc phát triển trò chơi 2D.

Hỗ trợ cộng đồng: Cộng đồng Pygame rất lớn và có nhiều tài liệu và tài liệu hướng dẫn sẵn sàng giúp bạn học và phát triển với Pygame.

2. Thư viện Random

Module Random trong Python. Python cung cấp sẵn một module cực kỳ dễ sử dụng để xử lý với các số ngẫu nhiên. Module này gọi là random, được cài đặt một bộ tạo số giả ngẫu nhiên và chứa các hàm cho phép chúng ta giải quyết trực tiếp nhiều vấn đề lập trình khác nhau sử dụng đến tính ngẫu nhiên.

# CHƯƠNG 2: THUẬT TOÁN ÁP DỤNG VÀO TRÒ CHƠI CỜ VUA

## 2.1 Thuật toán Naive Algorithm

### 2.1.1 Khái niệm

Thuật toán Naïve là một phương pháp đơn giản để giải quyết một vấn đề mà thường không hiệu quả bằng cách sử dụng cách tiếp cận trực tiếp và đơn giản nhất. Thuật toán Naïve thường được sử dụng để tạo ra một giải pháp đầu tiên cho một vấn đề cụ thể, nhưng thường không hiệu quả cho các tập dữ liệu lớn hoặc vấn đề phức tạp.

### 2.1.2 Một số đặc điểm của thuật toán Naive Algorithm:

1. Đơn giản: Thuật toán Naive thường dễ hiểu và dễ triển khai vì nó không đòi hỏi nhiều kiến thức phức tạp hoặc các kỹ thuật đặc biệt.

2. Hiệu suất thấp: Với các tập dữ liệu lớn hoặc vấn đề phức tạp, thuật toán Naïve thường không hiệu quả và có thể đòi hỏi thời gian tính toán lớn.

3. Không tận dụng được các cơ hội tối ưu: Thuật toán Naïve thường không tận dụng được các cơ hội tối ưu trong việc giải quyết vấn đề, do đó không đảm bảo giải pháp tối ưu.

4. Thường được sử dụng trong các trường hợp đơn giản hoặc để kiểm tra tính đúng đắn: Thuật toán Naïve thường được sử dụng khi cần một giải pháp nhanh chóng cho một vấn đề cụ thể hoặc khi cần kiểm tra tính đúng đắn của thuật toán trước khi triển khai các thuật toán hiệu suất cao hơn.

Tóm lại, thuật toán Naïve là một cách tiếp cận đơn giản và trực tiếp để giải quyết vấn đề, nhưng không phải lúc nào cũng hiệu quả và không thường được sử dụng trong các ứng dụng đòi hỏi hiệu suất cao hoặc tối ưu.

### 2.1.3 Áp dụng vào đề tài

Hàm getValidMoves trong mã được cung cấp triển khai thuật toán xác thực và tạo nước đi đơn giản cho một ván cờ. Dưới đây là bảng phân tích cách hoạt động của thuật toán này:

- Tạo tất cả các bước di chuyển có thể:

Đầu tiên, nó gọi hàm getAllPossibleMoves để tạo ra tất cả các nước đi có thể có cho quân cờ của người chơi hiện tại trên bàn cờ. Hàm này lặp lại trên tất cả các ô trên bảng và kiểm tra xem quân cờ trên mỗi ô có thuộc về người chơi hiện tại hay không. Nếu đúng như vậy, nó sẽ sử dụng chức năng di chuyển dành riêng cho quân cờ tương ứng (ví dụ: getPawnMoves, getRookMoves) để tạo các nước đi hợp lệ cho quân cờ đó.



Lặp lại di chuyển ngược:

Sau đó nó lặp lại danh sách các bước di chuyển được tạo ra. Khi xóa các phần tử khỏi danh sách trong khi lặp, sẽ an toàn hơn nếu quay lại để tránh hành vi không mong muốn.



Thực hiện từng bước di chuyển và kiểm tra:

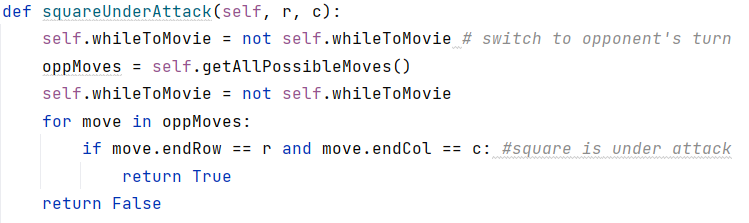
Đối với mỗi nước đi trong danh sách, nó sẽ thực hiện nước đi trên bảng bằng cách sử dụng hàm makeMove.



Nó tạm thời chuyển sang lượt của đối thủ bằng cách chuyển đổi thuộc tính whileToMovie.



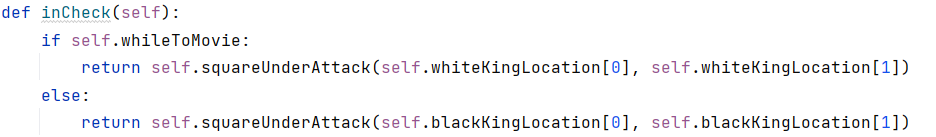
Nó kiểm tra xem đối thủ có thể tấn công vua của người chơi hiện tại hay không bằng cách sử dụng chức năng SquareUnderAttack. Nếu vua bị tấn công sau nước đi, điều đó có nghĩa là nước đi đó sẽ khiến vua của người chơi bị kiểm soát.



Loại bỏ các nước đi khiến Vua bị kiểm soát:

Nếu nước đi của đối thủ khiến vua của người chơi bị chiếu, nước đi đó sẽ bị xóa khỏi danh sách các nước đi hợp lệ.





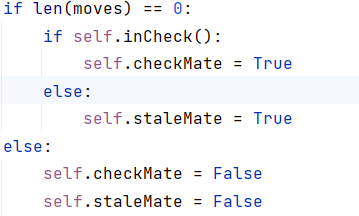
Hoàn tác di chuyển:

Sau khi kiểm tra, nó sẽ hoàn tác thao tác bằng cách sử dụng chức năng undoMove để đưa bảng về trạng thái ban đầu.



Kiểm tra Checkmate và Bế tắc:

Nếu, sau khi loại bỏ nước đi khiến vua bị chiếu, không còn nước đi hợp lệ nào, nó sẽ kiểm tra xem người chơi có bị chiếu hay không. Nếu người chơi đang bị chiếu, nó sẽ đặt thuộc tính checkMate thành True, cho biết rằng người chơi đang bị chiếu tướng. Ngược lại, nó đặt thuộc tính staleMate thành True, biểu thị sự bế tắc.



Trả lại các bước di chuyển hợp lệ:

Cuối cùng, hàm trả về danh sách các nước đi hợp lệ không khiến vua bị chiếu.



=>Thuật toán ngây thơ này tập trung vào việc đảm bảo rằng các nước đi được tạo ra không kiểm soát vua của người chơi.

## 2.2 Thuật toán Greedy algorithm

### 2.2.1 Khái niệm

Thuật toán Greedy algorithm là một thuật toán giải quyết một bài toán theo kiểu metaheuristic để tìm kiếm lựa chọn tối ưu địa phương ở mỗi bước đi với hy vọng tìm được tối ưu toàn cục.

### 2.2.2 Mã giả

Thuật toán Greedy algorithm có thể được áp dụng trong một số trường hợp trong game cờ vua, nhưng nó thường không đủ để đảm bảo tìm ra nước đi tối ưu trong tất cả các tình huống. Dưới đây là một ví dụ về mã giả của thuật toán Greedy để chọn nước đi trong game cờ vua, tập trung vào việc ưu tiên bảo vệ hoặc tấn công quân đối phương:

def greedy\_chess\_move(board):

best\_move = None

best\_score = -float('inf')

for move in generate\_possible\_moves(board):

board\_copy = make\_move(board, move)

score = evaluate\_position(board\_copy)

if score > best\_score:

best\_score = score

best\_move = move

return best\_move

def generate\_possible\_moves(board):

# Tạo danh sách các nước đi có thể thực hiện trên bảng cờ

moves = []

# Loops to generate moves...

return moves

def make\_move(board, move):

# Thực hiện nước đi trên bảng cờ và cập nhật trạng thái bảng

new\_board = board.copy()

# Make the move...

return new\_board

def evaluate\_position(board):

# Đánh giá trạng thái hiện tại của bảng cờ và trả về một điểm số

score = 0

# Evaluate the position...

return score

Trong ví dụ này, greedy\_chess\_move chọn nước đi dựa trên việc đánh giá tất cả các nước đi có thể và chọn nước có điểm số cao nhất. Tuy nhiên, để thực sự làm cho thuật toán tham lam hiệu quả trong game cờ vua, bạn cần phải cải tiến hàm đánh giá (evaluate\_position) và xác định cách ưu tiên các nước đi dựa trên tiến trình và tình huống cụ thể trong trò chơi.

Lưu ý rằng, trong cờ vua, thuật toán tham lam không đảm bảo tìm ra nước đi tối ưu trong tất cả các trường hợp, và để thực hiện một trò chơi cờ vua mạnh mẽ, bạn cần phải sử dụng các thuật toán tìm kiếm và đánh giá phức tạp hơn như MiniMax và Alpha-Beta Pruning.

### 2.2.3 Áp dụng vào đề tài

pieceScore: Đây là một từ điển trong đó mỗi loại quân cờ cờ vua được gán một điểm số. Các giá trị này thể hiện giá trị tương đối của các quân cờ trên bàn cờ. Ví dụ, quân hậu (Queen) có giá trị 10 điểm, quân xe (Rook) có giá trị 5 điểm, và cờ tốt (Pawn) có giá trị 1 điểm.



CHECKMATE và STALEMATE: Đây là các hằng số được sử dụng để biểu thị tình huống trong trò chơi. CHECKMATE có giá trị 1000, và STALEMATE có giá trị 0. Được sử dụng để xác định tình huống chiếu bí hoặc hòa.



findBestMove(gs, validMoves): Hàm này là trọng tâm của mã và sử dụng giải thuật đơn giản để tìm nước đi tốt nhất dựa trên điểm số vật chất của các quân cờ. Đây là cách nó hoạt động:

Hàm nhận một đối tượng GameState (gs) và danh sách các nước đi hợp lệ (validMoves) làm đầu vào.



Dấu này quan trọng để đánh giá đúng các nước đi từ cả hai bên.

maxScore: Biến này được khởi tạo với giá trị -CHECKMATE, đại diện cho điểm số tồi tệ nhất có thể (bị chiếu bí trước trí tuệ nhân tạo).



bestMove: Biến này được khởi tạo là None và sẽ lưu trữ nước đi tốt nhất được tìm thấy.



Vòng lặp qua validMoves: Hàm này duyệt qua từng nước đi trong danh sách validMoves.



Thực hiện nước đi: Nó thực hiện nước đi hiện tại trên bàn cờ bằng cách gọi gs.makeMove(playerMove). Điều này tạm thời cập nhật trạng thái trò chơi.



Tính toán điểm số:

Nếu nước đi dẫn đến chiếu bí (gs.checkMate là True), nó thiết lập score thành CHECKMATE. Chiếu bí là kết quả tốt nhất có thể, nên nó gán điểm số cao nhất.



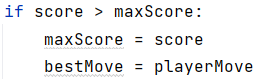
Nếu nước đi dẫn đến tình huống hòa (gs.staleMate là True), nó thiết lập score thành STALEMATE. Hòa là kết quả trung lập, nên nó gán một điểm số trung lập.



Nếu không phải chiếu bí hay hòa, nó tính toán điểm số sử dụng hàm scoreMaterial(), xem xét trạng thái hiện tại của bàn cờ. Dấu của điểm số được điều chỉnh dựa trên turnMultiplier.



So sánh điểm số: Nó so sánh điểm số score tính cho nước đi hiện tại với maxScore đã tìm thấy cho đến nay. Nếu điểm số mới cao hơn, nó cập nhật maxScore và ghi lại nước đi hiện tại là bestMove.



Hoàn tác nước đi: Sau khi đánh giá một nước đi, nó hoàn tác nước đi trên bàn cờ bằng cách gọi gs.undoMove(). Điều này khôi phục trạng thái trò chơi trước khi đánh giá nước đi tiếp theo.



Trả về nước đi tốt nhất: Sau khi đánh giá tất cả các nước đi hợp lệ, hàm trả về bestMove, đó là nước đi có điểm số cao nhất được tìm thấy trong quá trình đánh giá.



Hàm này sử dụng một hàm đánh giá đơn giản (điểm số vật chất) để tìm nước đi tốt nhất trong số các nước đi hợp lệ. Hãy nhớ rằng các động cơ cờ vua tiên tiến hơn sử dụng hàm đánh giá và thuật toán tìm kiếm phức tạp hơn để chơi mạnh hơn.

## 2.3 Thuật toán MiniMax

### 2.3.1 Khái niệm

Thuật toán MiniMax là một thuật toán recursive hoặc backtrack được sử dụng trong việc ra quyết định và lý thuyết trò chơi. Nó cung cấp một nước đi tối ưu cho người chơi giả sử rằng đối thủ cũng đang chơi một cách tối ưu.

### 2.3.2 Mã giả

Mã giả của thuật toán MiniMax trong Python được thực hiện bằng cách sử dụng đệ quy để tìm kiếm tất cả các tình huống có thể xảy ra trong trò chơi và chọn ra nước đi tốt nhất cho một người chơi cố định. Dưới đây là một phiên bản đơn giản của thuật toán MiniMax:

def min\_max (game\_state, depth, maximizing\_player):

if depth == 0 or game\_over(game\_state):

return evaluate(game\_state)

if maximizing\_player:

max\_eval = float('-inf')

for child\_state in generate\_children(game\_state):

eval = min\_max (child\_state, depth - 1, False)

max\_eval = max (max\_eval, eval)

return max\_eval

else:

min\_eval = float('inf')

for child\_state in generate\_children(game\_state):

eval = min\_max (child\_state, depth - 1, True)

min\_eval = min (min\_eval, eval)

return min\_eval

def find\_best\_move (game\_state, depth):

best\_move = None

max\_eval = float('-inf')

for child\_state in generate\_children(game\_state):

eval = min\_max (child\_state, depth - 1, False)

if eval > max\_eval:

max\_eval = eval

best\_move = child\_state

return best\_move

Trong mã giả này, MiniMax là hàm chính thực hiện thuật toán MiniMax. game\_state đại diện cho trạng thái hiện tại của trò chơi, depth là độ sâu tìm kiếm (giới hạn độ phức tạp), và maximizing\_player là một biến boolean cho biết người chơi hiện tại đang cố gắng tối ưu hóa giá trị hay không.

Bạn cần thay thế các hàm game\_over, evaluate, generate\_children bằng các hàm thích hợp cho trò chơi cụ thể mà bạn muốn áp dụng thuật toán MiniMax. game\_over kiểm tra xem trò chơi đã kết thúc hay chưa, evaluate tính toán giá trị của trạng thái trò chơi, và generate\_children tạo ra tất cả các trạng thái con có thể từ trạng thái hiện tại.

Cuối cùng, find\_best\_move là hàm để tìm ra nước đi tốt nhất dựa trên giá trị tối ưu được tính toán bằng thuật toán MiniMax.

### 2.3.3 Áp dụng vào đề tài

pieceScore: Đây là một từ điển trong đó mỗi loại quân cờ cờ vua được gán một điểm số. Các giá trị này thể hiện giá trị tương đối của các quân cờ trên bàn cờ. Ví dụ, quân hậu (Queen) có giá trị 10 điểm, quân xe (Rook) có giá trị 5 điểm, và cờ tốt (Pawn) có giá trị 1 điểm.



CHECKMATE và STALEMATE: Đây là các hằng số được sử dụng để biểu thị tình huống trong trò chơi. CHECKMATE có giá trị 1000, và STALEMATE có giá trị 0. Được sử dụng để xác định tình huống chiếu bí hoặc hòa.



findBestMove(gs, validMoves): Hàm này là trọng tâm của mã và sử dụng giải thuật đơn giản để tìm nước đi tốt nhất dựa trên điểm số vật chất của các quân cờ. Đây là cách nó hoạt động:

Hàm nhận một đối tượng GameState (gs) và danh sách các nước đi hợp lệ (validMoves) làm đầu vào.



Nó sử dụng biến turnMultipliter để xác định lượt của người chơi hiện tại (1 nếu là lượt người chơi trắng, -1 nếu là lượt người chơi đen).



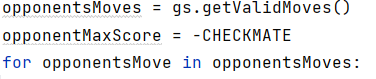
Nó tạo hai biến: opponentMinMaxScore (đại diện cho điểm số tốt nhất của đối thủ) và bestPlayerMove (đại diện cho nước đi tốt nhất của người chơi hiện tại).



Tiếp theo, nó lặp qua danh sách các nước đi hợp lệ của người chơi hiện tại (playerMove) và thực hiện mỗi nước đi trên bàn cờ tạm thời (gs.makeMove(playerMove)).



Sau đó, nó tính toán danh sách nước đi hợp lệ của đối thủ (opponentsMoves) và tiếp tục lặp qua các nước đi của đối thủ để xác định điểm số tốt nhất cho đối thủ (opponentMaxScore).



Nếu tình huống là chiếu bí (gs.checkMate), điểm số của nước đi này sẽ bằng -CHECKMATE (chiếu bí).



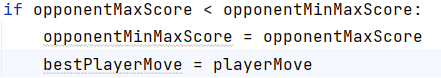
Nếu tình huống là hòa (gs.staleMate), điểm số của nước đi này sẽ bằng STALEMATE (hòa).



Nếu không, nó tính toán điểm số dựa trên hàm scoreMaterial() và áp dụng đối với trạng thái bàn cờ sau nước đi.



Cuối cùng, nếu điểm số của nước đi của đối thủ lớn hơn opponentMinMaxScore, thì opponentMinMaxScore sẽ được cập nhật và bestPlayerMove sẽ được gán nước đi hiện tại.



Hàm tiếp tục lặp qua các nước đi khác của người chơi hiện tại và cuối cùng trả về bestPlayerMove là nước đi tốt nhất mà nó đã tìm thấy.



## 2.4. Thuật toán negamax chính xác với cắt alpha-beta

### 2.4.1 Khái niệm

Khái niệm của thuật toán Negamax với cắt alpha-beta là một phương pháp tối ưu hóa trong tìm kiếm cây trò chơi, thường được áp dụng trong các trò chơi hai người tuần tự như cờ vua, cờ caro, và các trò chơi board game khác.

Minimax là một thuật toán tìm kiếm cây được sử dụng trong trò chơi hai người lượt đi, trong đó mỗi nút của cây đại diện cho một trạng thái của trò chơi và mỗi cạnh biểu thị một bước đi hợp lệ. Thuật toán đánh giá mỗi nút của cây bằng cách lần lượt duyệt qua các nút con, tối đa hóa giá trị ước lượng nếu đến lượt người chơi hiện tại và tối thiểu hóa giá trị ước lượng nếu đến lượt đối thủ.

Negamax là một biến thể của Minimax. Thay vì sử dụng hai hàm đánh giá riêng biệt cho lượt của người chơi và đối thủ, Negamax chỉ sử dụng một hàm đánh giá và áp dụng độ sâu đối chiếu âm để thay đổi dấu của giá trị đánh giá nếu đến lượt của đối thủ.

Cắt alpha-beta là một kỹ thuật tối ưu hóa để giảm số lượng nút cần phải duyệt qua trong cây tìm kiếm Minimax. Kỹ thuật này sử dụng hai tham số alpha và beta để đánh giá giá trị tốt nhất có thể đạt được cho người chơi hiện tại và giới hạn giá trị cần đạt được cho đối thủ. Khi giá trị của một nút con đã rõ ràng nằm ngoài khoảng [alpha, beta], cây tìm kiếm có thể bị cắt bỏ, giảm bớt thời gian tính toán.

### 2.4.2 Mã giả

Mã giả của thuật toán Negamax với cắt alpha-beta áp dụng vào trò chơi cờ vua bằng Python. Để minh họa, chúng ta sẽ sử dụng một biến thể đơn giản của trò chơi cờ vua, trong đó mỗi bước di chuyển sẽ chỉ là việc di chuyển một quân cờ từ một ô đến một ô khác trên bàn cờ.

# Hàm negamax với cắt alpha-beta cho trò chơi cờ vua

def negamax\_alpha\_beta(board, depth, alpha, beta, color):

if depth == 0 or board.is\_game\_over():

return color \* evaluate\_board(board)

max\_value = float('-inf')

for move in board.generate\_moves():

board.make\_move(move)

value = -negamax\_alpha\_beta(board, depth - 1, -beta, -alpha, -color)

board.undo\_move(move)

max\_value = max(max\_value, value)

alpha = max(alpha, value)

if alpha >= beta:

break # Cắt alpha-beta

return max\_value

# Hàm đánh giá bàn cờ

def evaluate\_board(board):

# Giả sử hàm đánh giá đơn giản trả về giá trị dương nếu màu trắng đang thắng và âm nếu màu đen đang thắng.

if board.is\_white\_winner():

return 1

elif board.is\_black\_winner():

return -1

else:

return 0

# Hàm chính để gọi negamax

def negamax\_with\_alpha\_beta(board, depth, color):

return negamax\_alpha\_beta(board, depth, float('-inf'), float('inf'), color)

### 2.4.3 Áp dụng vào đề tài

Xếp Hạng Các Cờ Quân (Piece\_score)

Biến mảnh\_score là một sơ đồ giữa ký hiệu của mỗi quân cờ và số tương ứng với sức mạnh của nó trong trò chơi.

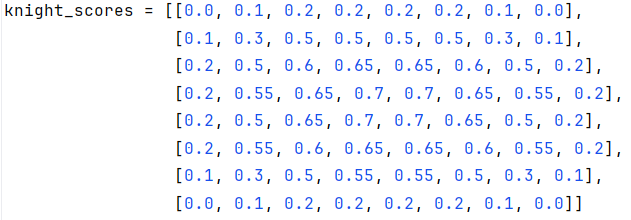


Các lá cờ quân đội được gán số dựa trên giá trị tương đối của chúng trên bàn cờ, ví dụ như quân hậu ("Q") có số cao nhất (9) vì sức mạnh của nó trong trò chơi.

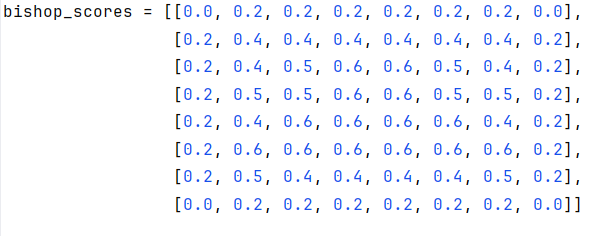
Điểm Số Vị Trí (Piece\_position\_scores)

Biến\_place\_position\_scores là một sơ đồ vị trí trung tâm từ điển của mỗi quân cờ trên bàn cờ và một ma trận tương ứng với vị trí đó.

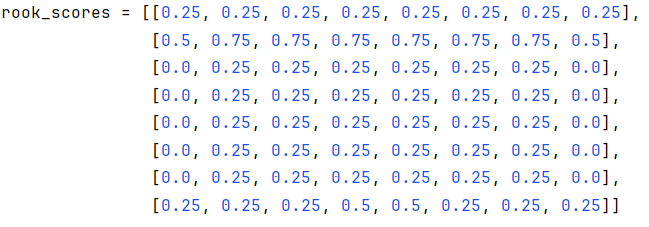
Điểm số vị trí giúp đánh giá sức mạnh của quân cờ dựa trên vị trí của chúng trên bàn cờ, giúp thuật toán đánh giá giá một cách linh hoạt hơn.



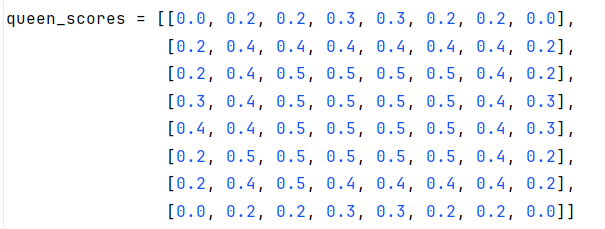
Bảng điểm vị trí cho quân mã



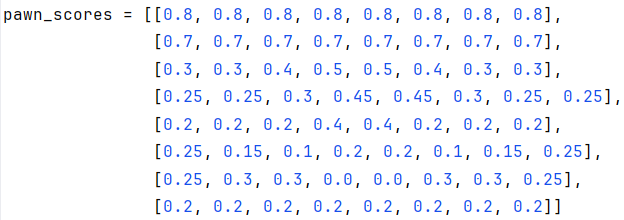
Bảng điểm vị trí cho quân tịnh



Bảng điểm vị trí cho quân xe



Bảng điểm vị trí cho quân hậu



Bảng điểm vị trí cho quân tốt

Hằng Số và Thâm Dò (CHECKMATE, Bế tắc, DEPTH)

CHECKMATE: Điểm số được phân bổ cho vấn đề tướng.



STALEMATE: Điểm số được chỉ định cho vấn đề bí ẩn (không thể thực hiện bất kỳ bước đi hợp lệ nào).



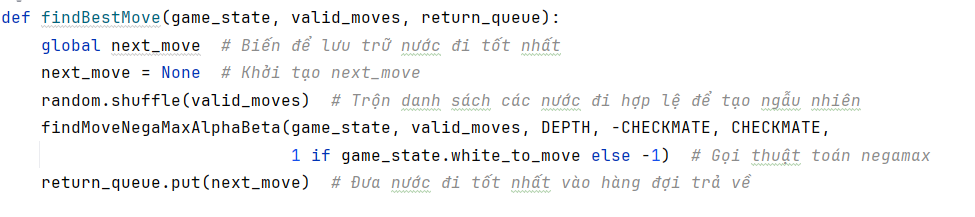
DEPTH: Sâu của tìm kiếm thuật toán, quyết định độ sâu mà thuật toán sẽ duyệt trong tìm kiếm cây.



Hàm Tìm Nước Đi Tốt Nhất (findBestMove)

Hàm này là điểm khởi đầu cho công việc tìm nước đi tốt nhất trong trạng thái trò chơi hiện tại.

Nó khởi động công việc tìm kiếm nước đi tốt nhất bằng thuật toán negamax với cắt alpha-beta.



Hàm Tìm Nước Đi Negamax Alpha-Beta (findMoveNegaMaxAlphaBeta)

Đây là hàm đệ quy thực hiện thuật toán negamax chính xác với cắt alpha-beta.

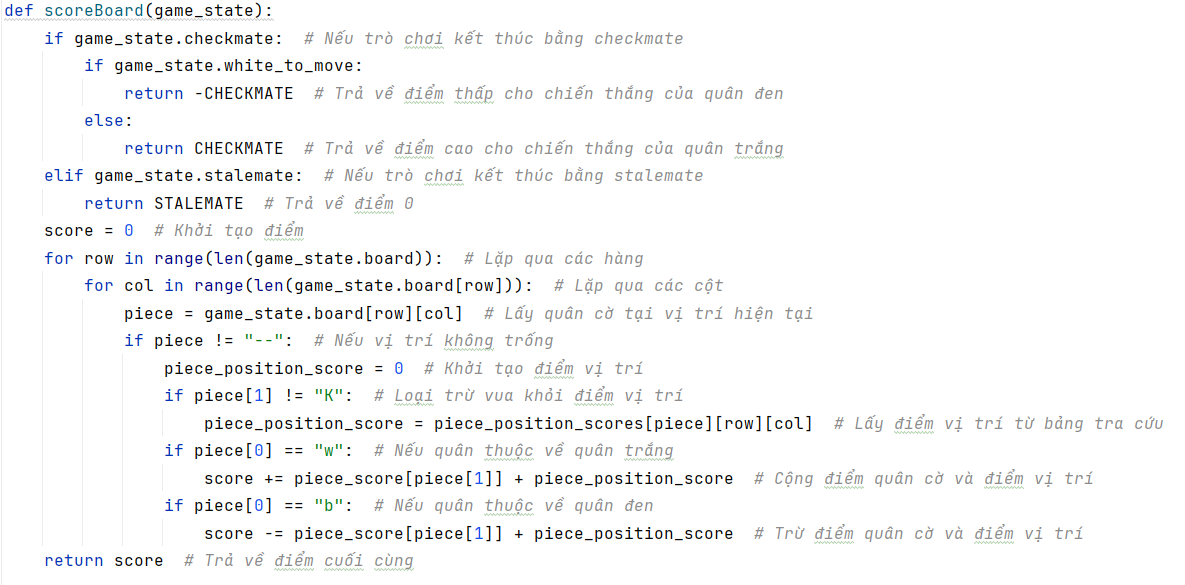
Hàm này khám phá các nước đi có thể và đánh giá chúng đệ quy để tìm ra nước đi tốt nhất.



Hàm Điểm Số Bảng Cờ (Bảng điểm)

Hàm này đánh giá trạng thái trò chơi hiện tại và chỉ định số cho nó.

Nó xem xét các yếu tố như tham chiếu tướng, bí ẩn, giá trị quân cờ và điểm số vị trí.



# CHƯƠNG 3: PHÁT TRIỂN VÀ TRIỂN KHAI ỨNG DỤNG

## 3.1 Cài đặt các thư viện cần thiết

Thư viện pygame:

Để cài đặt thư viện Pygame trên máy tính của bạn, bạn cần thực hiện các bước sau:

Mở Command Prompt hoặc Terminal tùy theo hệ điều hành bạn đang sử dụng (Windows hoặc macOS/Ubuntu/Linux).

Sử dụng pip (trình quản lý gói Python) để cài đặt Pygame. Để làm điều này, bạn có thể sử dụng lệnh sau:



Chờ cho đến khi quá trình cài đặt hoàn thành. Pygame sẽ tự động được tải về và cài đặt trên máy tính của bạn.

Để kiểm tra xem Pygame đã được cài đặt thành công hay chưa, bạn có thể mở Python và nhập các lệnh sau:



Nếu bạn thấy phiên bản của Pygame được in ra màn hình, điều đó có nghĩa là bạn đã cài đặt thành công Pygame.

## 3.2 Huấn luyện các nước đi hợp lệ

### 3.2.1 Huấn luyện cho con tốt

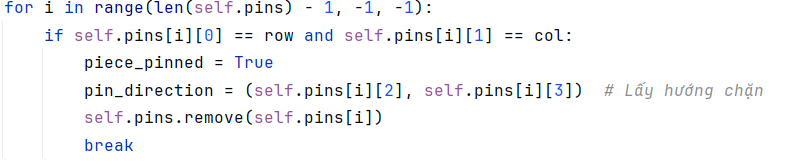
Đầu tiên xác định xem quân cờ có bị chặn không

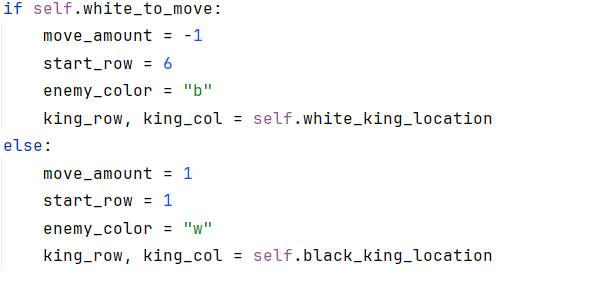
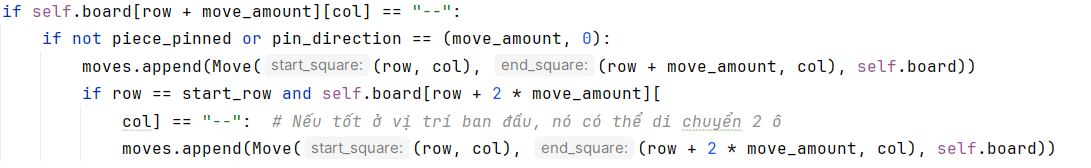
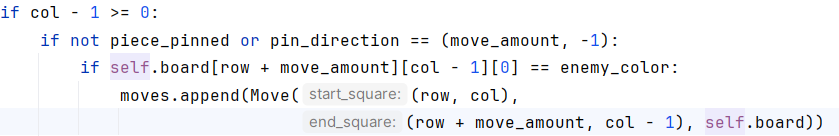
Nếu bị chặn sử dụng vòng for để:

Duyệt qua các hướng có thể của các quân cờ.

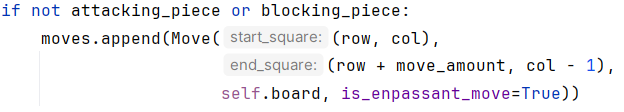
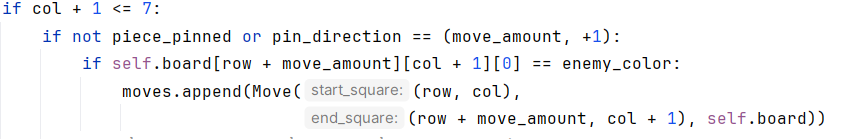
Xác định màu quân cờ của đối phương và của mình.

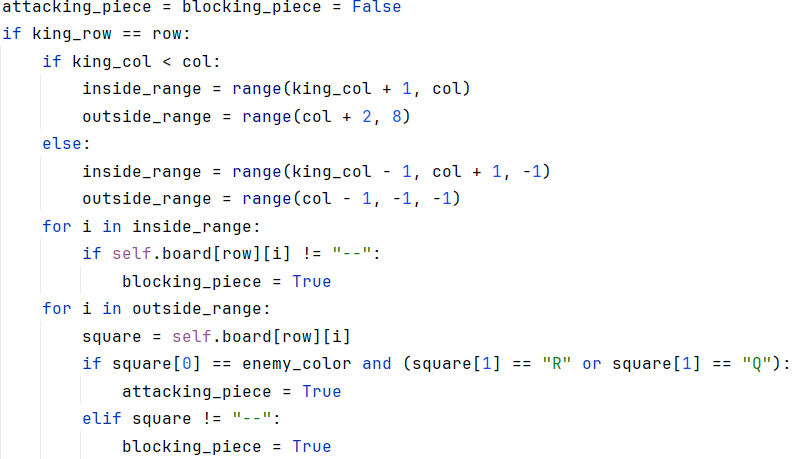
Kiểm tra các vị trí có thể bị chiếu từ các quân đối phương như: tướng, xe, ngựa, tốt và hậu.

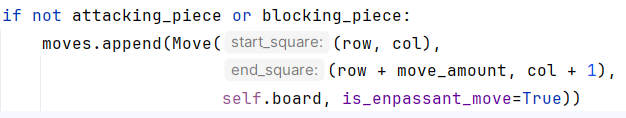


Xác định hướng và mức độ di chuyển của quân tốt dựa trên màu và vị trí  
  
Nếu ô trước tốt là trống, thì tốt có thể di chuyển về phía trước còn nếu ở ô ban đầu thì có thể duy chuyển 2 ô  
  
Nếu có quân địch ở chéo trái trước, tốt có thể tiến hành ăn quân đó  


Kiểm tra xem có thể di chuyển theo quy tắc "en passant" không  
  
Kiểm tra xem việc di chuyển theo quy tắc "en passant" có thể tạo ra tình huống chiếu không  
 

Nếu không có quân tấn công hoặc có quân chặn, thì thêm nước đi "en passant" vào danh sách  
  
Tương tự như trên, kiểm tra nước đi có thể ăn quân địch ở chéo phải trước đó  


Kiểm tra xem có thể di chuyển theo quy tắc "en passant" không  
 Kiểm tra xem việc di chuyển theo quy tắc "en passant" có thể tạo ra tình huống chiếu không  


Nếu không có quân tấn công hoặc có quân chặn, thì thêm nước đi "en passant" vào danh sách  


### 3.2.2 Huấn luyện con xe

Hàm getRookMoves được sử dụng để tạo ra các nước đi hợp lệ cho quân Xe (Rook) trong trò chơi cờ vua.

Đầu tiên, nó xác định các hướng di chuyển cho quân Xe. Quân Xe có thể di chuyển theo các hướng dọc (theo cột) hoặc theo các hướng ngang (theo dòng). Các hướng di chuyển được lưu trữ trong biến directions.



Nó xác định màu của quân Xe (trắng hoặc đen) dựa trên biến whileToMovie. Điều này quan trọng để biết quân Xe của người chơi hiện tại là màu gì.



Duyệt qua các hướng di chuyển: Sử dụng vòng lặp for để duyệt qua tất cả các hướng di chuyển có thể của quân Xe.

Duyệt qua ô trong hướng di chuyển: Sử dụng vòng lặp for i in range(1, 8) để duyệt qua các ô trên bàn cờ theo từng hướng. Biến i đại diện cho khoảng cách từ quân Xe đến ô đang xem xét.



Kiểm tra điều kiện cho từng ô:

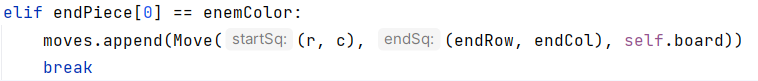
Kiểm tra xem ô đang xem xét có nằm trong bàn cờ không bằng cách sử dụng điều kiện if 0 <= endRow < 8 and 0 <= endCol < 8.



Kiểm tra xem ô đó có trống không ("--") không. Nếu trống, thì thêm nước đi từ vị trí hiện tại đến ô đó vào danh sách moves. Điều này cho phép quân Xe di chuyển đến ô trống mà không cần bắt quân cờ nào.



Kiểm tra xem ô đó có chứa quân đối phương không (endPiece[0] == enemColor). Nếu có, thêm nước đi từ vị trí hiện tại đến ô đó vào danh sách moves. Điều này cho phép quân Xe di chuyển đến ô đó và bắt quân đối phương.



Nếu ô đó chứa quân cùng màu với quân Xe hiện tại (endPiece[0] giống với màu của quân Xe), thì dừng vòng lặp vì quân Xe không thể đi qua quân cùng màu để tiếp tục di chuyển.

Dừng vòng lặp: Nếu ô đang xem xét ra khỏi bàn cờ (else: break), thì dừng vòng lặp vì không thể đi xa hơn trong hướng đó.

Hàm này tạo ra các nước đi hợp lệ cho quân Xe theo các hướng dọc và ngang trên bàn cờ, đồng thời kiểm tra xem có quân cờ nào ở đó để bắt hoặc có quân cùng màu không để ngăn quân Xe tiếp tục di chuyển.

### 3.2.3 Huấn luyện cho con mã

Hàm getKnightMoves được sử dụng để tạo ra danh sách các nước đi hợp lệ cho quân Ngựa (Knight) trong trò chơi cờ vua.

Xác định các bước di chuyển có thể cho quân Ngựa: Quân Ngựa có thể thực hiện các bước di chuyển đặc biệt, được xác định trong biến knightMoves. Các bước di chuyển này là sự kết hợp của các đơn vị [-2, -1], [-2, 1], [-1, -2], [-1, 2], [1, -2], [1, 2], [2, -1], và [2, 1] trên bàn cờ.



Xác định màu của quân Ngựa: Để biết quân Ngựa của người chơi hiện tại là màu gì, nó sử dụng biến whileToMovie. Nếu whileToMovie là True, thì quân Ngựa là màu trắng (allyColor = "w"), ngược lại nếu whileToMovie là False, thì quân Ngựa là màu đen (allyColor = "b").



Duyệt qua các bước di chuyển: Sử dụng vòng lặp for m in knightMoves để duyệt qua tất cả các bước di chuyển có thể của quân Ngựa.



Xác định ô đích: Biến endRow và endCol được tính dựa trên vị trí hiện tại của quân Ngựa và bước di chuyển đang xem xét.



Kiểm tra điều kiện cho từng ô:

Kiểm tra xem ô đang xem xét có nằm trong bàn cờ không bằng cách sử dụng điều kiện if 0 <= endRow < 8 and 0 <= endCol < 8.



Kiểm tra xem ô đó có trống không ("--") không. Nếu trống, thêm nước đi từ vị trí hiện tại đến ô đó vào danh sách moves. Điều này cho phép quân Ngựa di chuyển đến ô trống mà không cần bắt quân cờ nào.

Kiểm tra xem ô đó có chứa quân đối phương không (endPiece[0] != allyColor). Nếu có, thêm nước đi từ vị trí hiện tại đến ô đó vào danh sách moves. Điều này cho phép quân Ngựa di chuyển đến ô đó và bắt quân đối phương.



Kết quả là, hàm getKnightMoves tạo ra danh sách các nước đi hợp lệ cho quân Ngựa bằng cách áp dụng các bước di chuyển đặc biệt của nó và kiểm tra xem có quân cờ nào ở đó để bắt hoặc có quân cùng màu không để ngăn quân Ngựa tiếp tục di chuyển.

### 3.2.4 Huấn luyện cho con tượng

Hàm getBishopMoves được sử dụng để tạo ra danh sách các nước đi hợp lệ cho quân Tượng (Bishop) trong trò chơi cờ vua.

Xác định các hướng di chuyển cho quân Tượng: Quân Tượng có thể di chuyển theo các đường chéo trên bàn cờ. Dưới dạng các cặp số nguyên, biến directions biểu thị các hướng di chuyển. Ví dụ: directions = ((-1, -1), (-1, 1), (1, -1), (1, 1)) biểu thị cho các đường chéo đi lên trái, đi lên phải, đi xuống trái và đi xuống phải.



Xác định màu của quân Tượng: Để biết quân Tượng của người chơi hiện tại là màu gì, nó sử dụng biến whileToMovie. Nếu whileToMovie là True, thì quân Tượng là màu trắng (enemyColor = "b"), ngược lại nếu whileToMovie là False, thì quân Tượng là màu đen (enemyColor = "w").



Duyệt qua các hướng di chuyển: Sử dụng vòng lặp for để duyệt qua tất cả các hướng di chuyển có thể của quân Tượng.



Duyệt qua ô trong hướng di chuyển: Sử dụng vòng lặp for i in range(1, 8) để duyệt qua các ô trên bàn cờ theo từng hướng. Biến i đại diện cho khoảng cách từ quân Tượng đến ô đang xem xét.

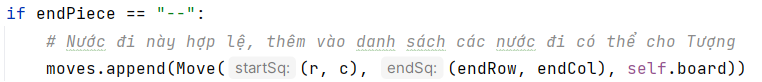


Kiểm tra điều kiện cho từng ô:

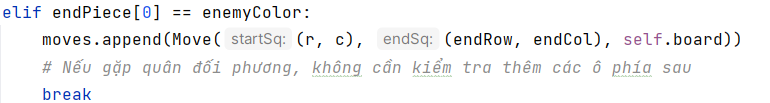
Kiểm tra xem ô đang xem xét có nằm trong bàn cờ không bằng cách sử dụng điều kiện if 0 <= endRow < 8 and 0 <= endCol < 8.



Kiểm tra xem ô đó có trống không ("--") không. Nếu trống, thì thêm nước đi từ vị trí hiện tại đến ô đó vào danh sách moves. Điều này cho phép quân Tượng di chuyển đến ô trống mà không cần bắt quân cờ nào.



Kiểm tra xem ô đó có chứa quân đối phương không (endPiece[0] == enemyColor). Nếu có, thêm nước đi từ vị trí hiện tại đến ô đó vào danh sách moves. Điều này cho phép quân Tượng di chuyển đến ô đó và bắt quân đối phương.



Nếu ô đó chứa quân cùng màu với quân Tượng hiện tại (endPiece[0] giống với màu của quân Tượng), thì dừng vòng lặp vì quân Tượng không thể đi qua quân cùng màu để tiếp tục di chuyển.

Dừng vòng lặp: Nếu ô đang xem xét ra khỏi bàn cờ (else: break), thì dừng vòng lặp vì không thể đi xa hơn trong hướng đó.

Kết quả là, hàm getBishopMoves tạo ra danh sách các nước đi hợp lệ cho quân Tượng theo các đường chéo trên bàn cờ, đồng thời kiểm tra xem có quân cờ nào ở đó để bắt hoặc có quân cùng màu không để ngăn quân Tượng tiếp tục di chuyển.

### 3.2.5 Huấn luyện cho con vua

Hàm getKingMoves được sử dụng để tạo ra danh sách các nước đi hợp lệ cho quân Vua (King) trong trò chơi cờ vua.

Xác định các nước đi có thể cho quân Vua: Quân Vua có thể di chuyển đến tất cả các ô lân cận (8 hướng di chuyển) xung quanh nó.



Xác định màu của quân Vua: Để biết quân Vua của người chơi hiện tại là màu gì, nó sử dụng biến whileToMovie. Nếu whileToMovie là True, thì quân Vua là màu trắng (allyColor = "w"), ngược lại nếu whileToMovie là False, thì quân Vua là màu đen (allyColor = "b").



Duyệt qua các hướng di chuyển: Sử dụng vòng lặp for i in range(8) để duyệt qua tất cả các hướng di chuyển có thể của quân Vua.



Xác định ô đích: Biến endRow và endCol được tính dựa trên vị trí hiện tại của quân Vua và hướng di chuyển đang xem xét.



Kiểm tra điều kiện cho từng ô:

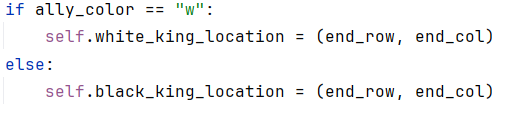
Kiểm tra xem ô đang xem xét có nằm trong bàn cờ không bằng cách sử dụng điều kiện if 0 <= endRow <= 7 and 0 <= endCol <= 7.



Kiểm tra xem ô đó có trống không ("--") không. Nếu trống, thêm nước đi từ vị trí hiện tại đến ô đó vào danh sách moves. Điều này cho phép quân Vua di chuyển đến ô trống mà không cần bắt quân cờ nào.

Kiểm tra xem ô đó có chứa quân đối phương không (endPiece[0] != allyColor).

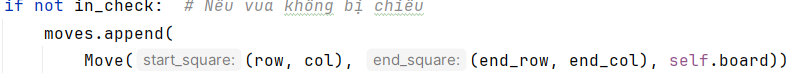
Lưu vị trí vua tạm thời



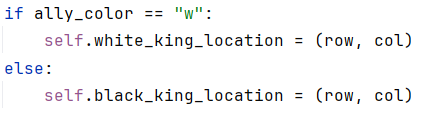
Kiểm tra vua có bị chiếu hay không



Nếu vua mà không bị chiếu thì lưu đó là nước đi hợp lệ



Khôi phục vị trí ban đầu của vua



Kết quả là, hàm getKingMoves tạo ra danh sách các nước đi hợp lệ cho quân Vua trong tất cả các hướng xung quanh quân Vua, đồng thời kiểm tra xem có quân cờ nào ở đó để bắt hoặc có quân cùng màu không để ngăn quân Vua tiếp tục di chuyển.

### 3.2.6 Huấn luyện cho con hậu

Hàm getQueenMoves được sử dụng để tạo danh sách các nước đi hợp lệ cho quân Hậu (Queen) trong trò chơi cờ vua bằng cách kết hợp các bước di chuyển của quân Xe (Rook) và quân Tượng (Bishop).

Gọi hàm getRookMoves(r, c, moves): Hàm này thêm tất cả các nước đi hợp lệ cho quân Xe từ vị trí (r, c) vào danh sách moves.



Gọi hàm getBishopMoves(r, c, moves): Hàm này thêm tất cả các nước đi hợp lệ cho quân Tượng từ vị trí (r, c) vào danh sách moves.



Bằng cách gọi hai hàm con này, getQueenMoves tổng hợp danh sách các nước đi hợp lệ từ cả hai loại quân Xe và quân Tượng và thêm chúng vào danh sách moves. Kết quả là, moves sẽ chứa tất cả các nước đi hợp lệ cho quân Hậu từ vị trí (r, c).

### 3.2.7 Huấn luyện con vua nhập thành

Đầu tiên kiểm tra quân xe bị ăn



Vô hiệu hóa quyền castling bên trái cho trắng nếu quân xe trắng bị ăn từ cột bên trái

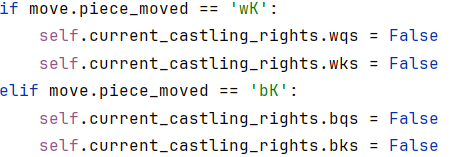


Vô hiệu hóa quyền castling bên phải cho trắng nếu quân xe trắng bị ăn từ cột bên phải



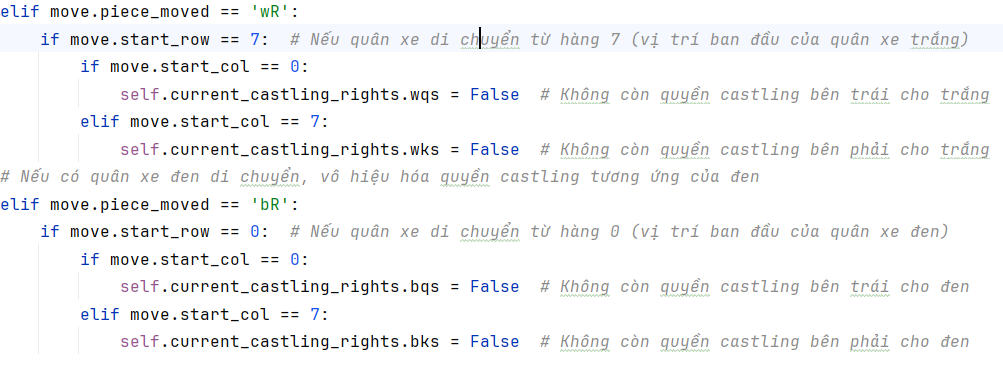
Ở đây, nếu quân xe trắng bị ăn, ta kiểm tra xem nó bị ăn từ cột nào. Nếu nó bị ăn từ cột 0 (bên trái), quyền castling bên trái của trắng (wqs) sẽ bị vô hiệu hóa. Tương tự, nếu nó bị ăn từ cột 7 (bên phải), quyền castling bên phải của trắng (wks) sẽ bị vô hiệu hóa.

Kiểm tra vua di chuyển



Ở đoạn này, nếu vua trắng hoặc vua đen di chuyển, tất cả quyền castling của bên tương ứng sẽ bị vô hiệu hóa.

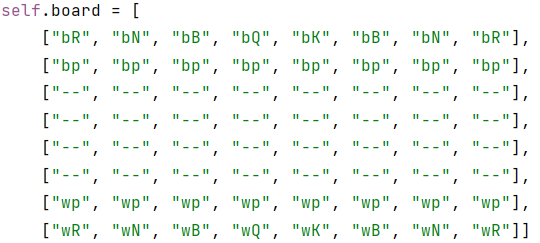
Kiểm tra quân xe di chuyển



Ở đây, nếu quân xe trắng hoặc đen di chuyển từ vị trí ban đầu của nó, quyền castling tương ứng của bên đó sẽ bị vô hiệu hóa.

Như vậy, hàm updateCastleRights này giúp duy trì các quyền castling trong trò chơi cờ vua, đảm bảo rằng người chơi không thể thực hiện castling sau khi có những thay đổi trong bảng cờ.

## 3.3 Giao diện trò chơi



Dòng mã này định nghĩa một biến self.board là một danh sách hai chiều để lưu trữ trạng thái của bảng cờ vua. Mỗi phần tử trong danh sách self.board đại diện cho một ô trên bảng cờ và chứa thông tin về quân cờ hiện tại đặt ở ô đó.

Danh sách self.board có kích thước 8x8, tương ứng với 8 hàng và 8 cột trên bảng cờ vua.

Mỗi quân cờ được biểu diễn bằng hai ký tự. Ví dụ, "wp" đại diện cho quân tốt trắng, "bR" đại diện cho quân xe đen, "wK" đại diện cho quân vua trắng, vv.

Dòng 1 và 8 của self.board đại diện cho hàng đầu và hàng cuối của bảng cờ vua, và chứa quân cờ của một bên (trắng và đen) theo thứ tự "wR", "wN", "wB", "wQ", "wK", "wB", "wN", "wR" cho bên trắng và "bR", "bN", "bB", "bQ", "bK", "bB", "bN", "bR" cho bên đen.

Dòng 2 và 7 của self.board đại diện cho hàng các quân tốt của mỗi bên, với "wp" cho quân tốt trắng và "bp" cho quân tốt đen.

Dòng 3 đến 6 của self.board đại diện cho các hàng ẩn, không có quân cờ (được biểu diễn bằng "--").

Việc tạo giao diện trò chơi cờ vua là một dự án phức tạp và đòi hỏi nhiều kiến thức về lập trình giao diện đồ họa (GUI). Dưới đây một giao diện đơn giản cho trò chơi cờ vua bằng Python dựa theo đoan code trên:



Hình 3.3. 1: Hỉnh ảnh giao diện cờ vua

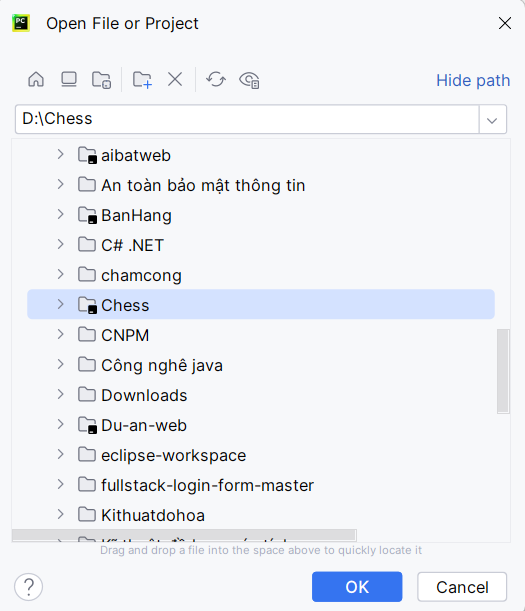
## 3.4 Các bước khởi chạy chương trình

Bước 1: Mở ứng dụng Pycharm:



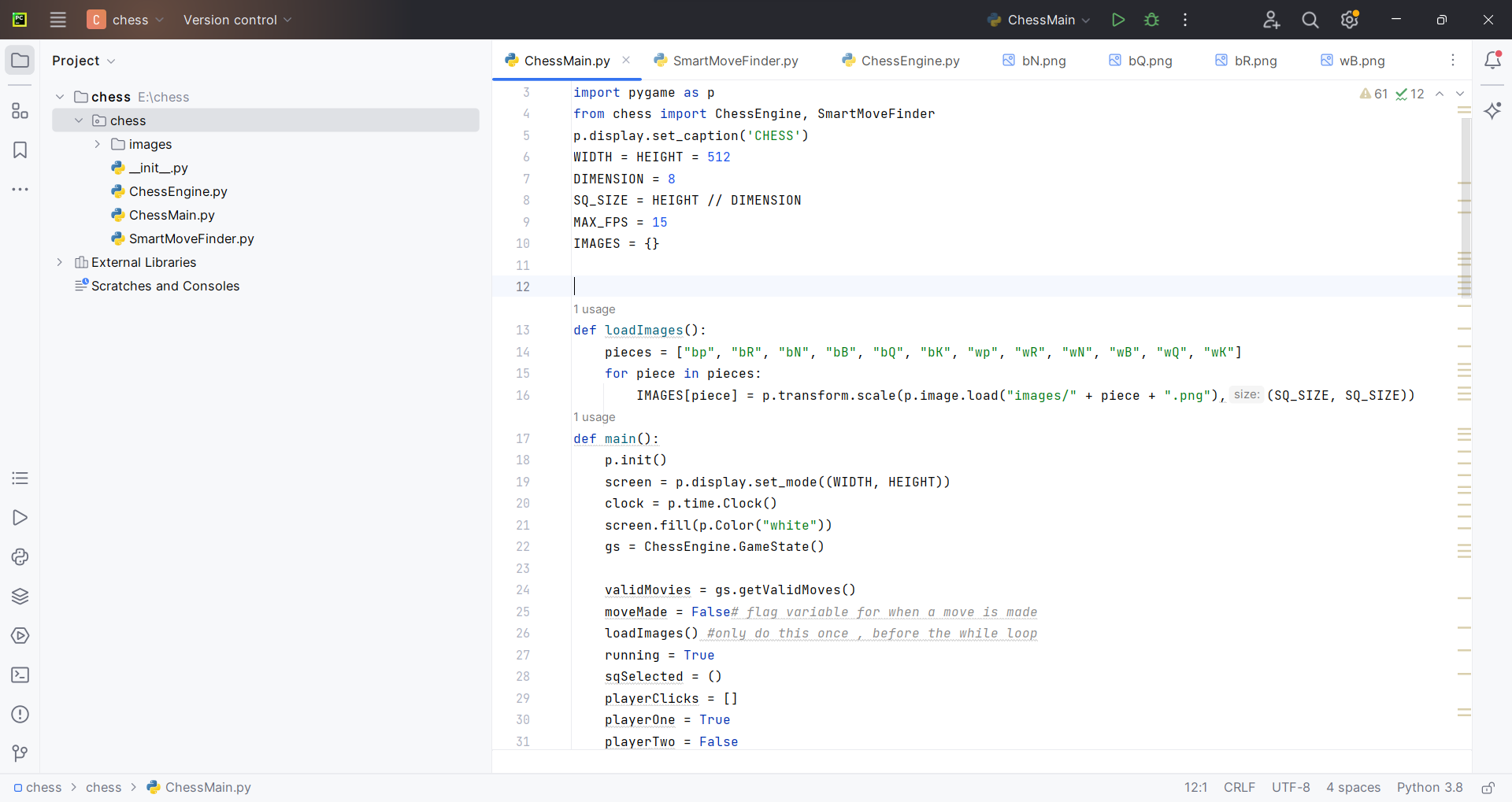
Hình 3.3. 2: Hình ảnh phần mềm pycharm

Bước 2: Mở tìm file lưu chương trình:



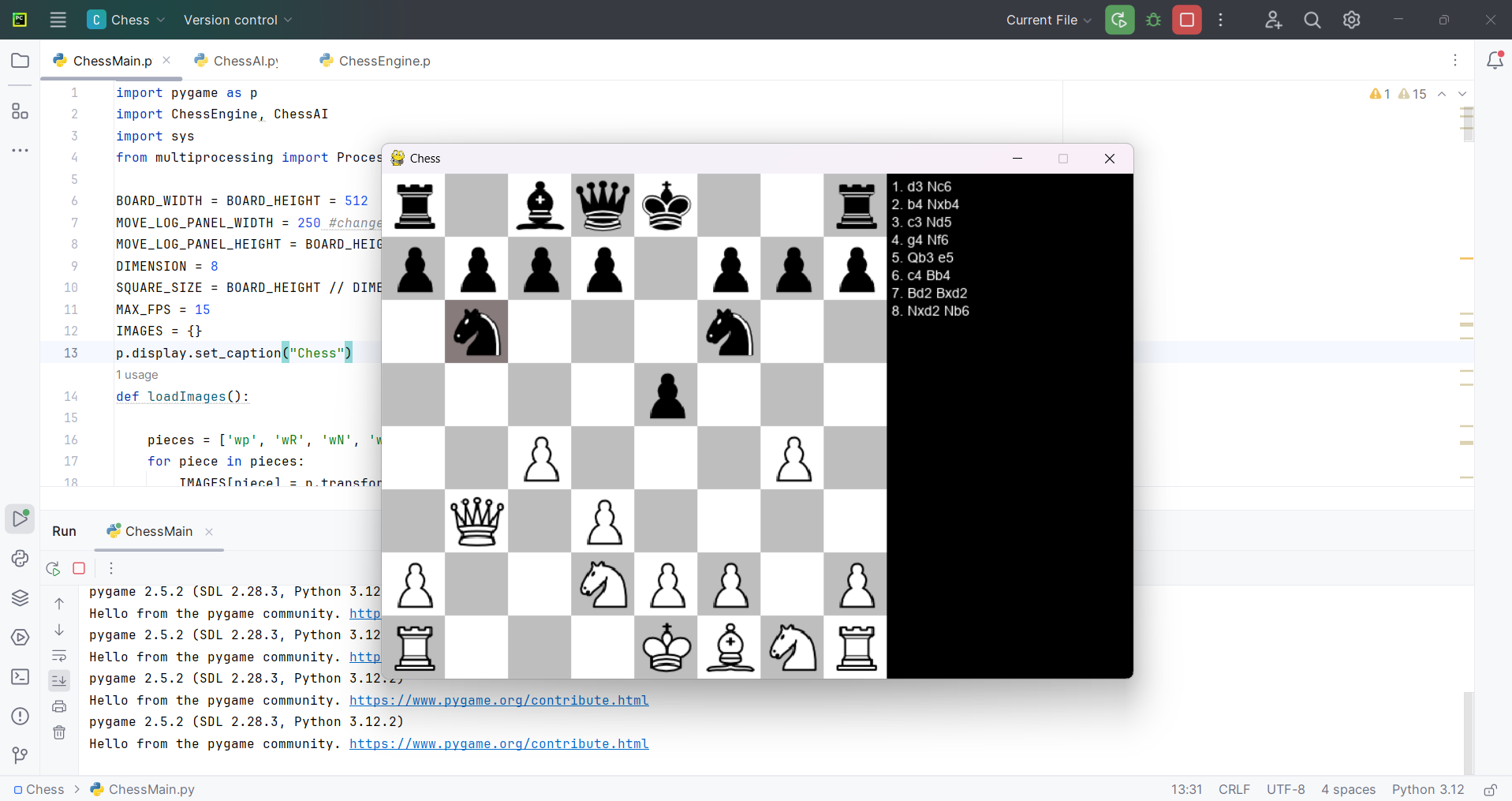
Hình 3.3. 3: Hình ảnh lưu file chương trình

Bước 3: Lựa chọn file ChessMain và bấm vào run để chạy chương trình:



Hình 3.3. 4: Hình ảnh lựa chọn file ChessMain và bấm vào run để chạy chương trình

Bước 4: Hoàn tất hiển thị chương trình



Hình 3.3. 5: Hình ảnh hoàn tất hiển thị chương trình

# KẾT LUẬN

Trong suốt quá trình nghiên cứu và xây dựng trò chơi cờ vua dựa trên thuật toán Minimax và thuật toán tham lam, chúng tôi đã trải qua một hành trình thú vị và hấp dẫn. Đề tài này đã đặt ra một số mục tiêu và đã đạt được những kết quả quan trọng, và dưới đây là một số điểm chính trong phần kết luận:

Hiểu rõ hơn về Minimax và thuật toán tham lam: Trong quá trình nghiên cứu, chúng tôi đã nắm vững kiến thức về cả thuật toán Minimax và thuật toán tham lam. Điều này đã giúp chúng tôi áp dụng hiệu quả các thuật toán này trong việc phát triển trò chơi cờ vua.

Xây dựng trò chơi cờ vua: Chúng tôi đã thành công trong việc xây dựng một trò chơi cờ vua sử dụng thuật toán Minimax để tối ưu hóa nước đi của máy tính. Trò chơi này đã đạt được mức độ chơi tương đối cao và cung cấp trải nghiệm thú vị cho người chơi.

Khám phá tiềm năng của trí tuệ nhân tạo trong trò chơi cờ vua: Đề tài này đã khám phá và chứng minh rằng trí tuệ nhân tạo có thể được ứng dụng rộng rãi trong lĩnh vực trò chơi cờ vua. Việc sử dụng thuật toán Minimax và thuật toán tham lam đã cung cấp cơ hội để tạo ra các đối thủ máy tính thông minh và thách thức.

Hướng phát triển tương lai: Mặc dù đã đạt được một số thành tựu quan trọng, còn rất nhiều khả năng phát triển trong lĩnh vực này. Các nghiên cứu sau này có thể tập trung vào việc tối ưu hóa thuật toán Minimax và thuật toán tham lam, cũng như phát triển trò chơi cờ vua có tính năng và giao diện người dùng phức tạp hơn.

Trong tổng kết, việc nghiên cứu và xây dựng trò chơi cờ vua dựa trên thuật toán Minimax và thuật toán tham lam đã đánh dấu một bước quan trọng trong việc kết hợp trí tuệ nhân tạo và giải trí. Chúng tôi hi vọng rằng đề tài này đã cung cấp một cái nhìn sâu hơn về tiềm năng của công nghệ trong việc cải thiện trải nghiệm chơi cờ vua và khuyến khích sự phát triển trong lĩnh vực này trong tương lai.

# TÀI LIỆU THAM KHẢO

[1] How to Play Chess [Online], Từ: <https://www.chess.com/learn-how-to-play-chess>

[2] Naive Pattern Searching Algorithm in Data Structure [Online], Từ: <https://www.tutorialspoint.com/data_structures_algorithms/naive_pattern_searching_algorithm.htm>

[3] Greedy Algorithms [Online], Từ: <https://www.geeksforgeeks.org/greedy-algorithms/?ref=shm>

[4] Mini-Max Algorithm in Artificial Intelligence [Online], Từ: https://www.javatpoint.com/mini-max-algorithm-in-ai

[5] Alpha-Beta Pruning [Online], Từ: https://www.javatpoint.com/ai-alpha-beta-pruning

[6] PGS.TS Từ Minh Phương, Nhập môn trí tuệ nhân tạo, Học viện bưu chính viễn thông

[7] Thuật toán tham lam - Greedy Algorithm [Online], Từ: <https://viblo.asia/p/thuat-toan-tham-lam-greedy-algorithm-XQZGxozlvwA>

[8] Thuật toán Mini-Max trong Trí tuệ nhân tạo [Online], Từ: <https://websitehcm.com/thuat-toan-mini-max-trong-tri-tue-nhan-tao/>

[9] Thuật toán NegaMax - Biến thể tối giản của MiniMax [Online], Từ: <https://viblo.asia/p/thuat-toan-negamax-bien-the-toi-gian-cua-minimax-gDVK2QEm5Lj>

[10] Giải Thuật Cắt Tỉa Alpha-beta [Online], Từ: https://www.iostream.co/article/giai-thuat-cat-tia-alpha-beta-Wu7F1