**ĐẠI HỌC ĐÀ NẴNG**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM**

**KHOA TIN HỌC**

****

**ĐỒ ÁN MÔN HỌC**

**ĐỀ TÀI**

**XẾP TỐI ĐA QUÂN MÃ KHÔNG KHỐNG CHẾ NHAU TRÊN BÀN CỜ N X N**

**Sinh viên thực hiên: Trần Nguyễn Ngọc Khoa**

**Lớp: 22CNTT1**

**Giảng viên hướng dẫn: TS. Đoàn Duy Bình**

Đà Nẵng – 2024

**MỤC LỤC**

# MỞ ĐẦU

Trong lĩnh vực khoa học máy tính, bài toán sắp xếp tối đa quân mã trên bàn cờ n×n mà không quân mã nào khống chế nhau là một bài toán tổ hợp thú vị và mang tính thách thức cao. Bài toán không chỉ đòi hỏi tư duy logic mà còn mở ra nhiều ứng dụng thực tiễn trong tối ưu hóa tài nguyên, lý thuyết đồ thị và trí tuệ nhân tạo.

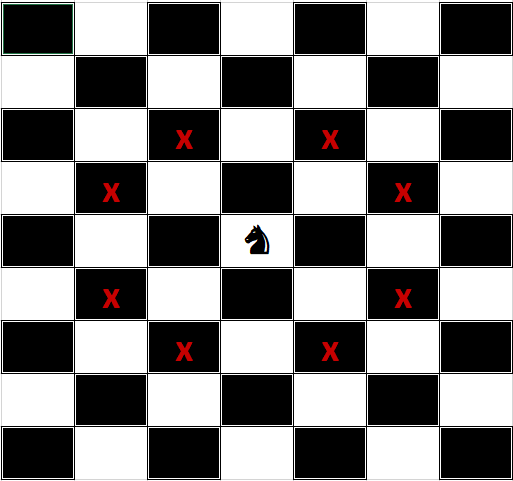
Mục tiêu của báo cáo là xây dựng phương pháp giải quyết bài toán sắp xếp tối ưu số lượng quân mã trên bàn cờ n×n sao cho không quân mã nào khống chế nhau. Báo cáo tập trung vào việc áp dụng các thuật toán tối ưu để giải quyết bài toán, đồng thời phân tích tính hiệu quả và tính khả thi của các phương pháp được đề xuất. Ngoài ra, các kết quả đạt được sẽ được đánh giá và trình bày thông qua quá trình mô phỏng và thực nghiệm, nhằm minh họa rõ ràng tính khả thi của các giải pháp đã triển khai.

Nội dung báo cáo được chia thành bốn phần chính. **Chương 1: Tổng quan về bài toán** giới thiệu lý thuyết cơ bản, ý nghĩa và các ứng dụng thực tiễn của bài toán sắp xếp tối ưu quân mã trên bàn cờ. **Chương 2: Phương pháp giải bài toán** trình bày chi tiết các phương pháp, thuật toán và công cụ được sử dụng để tìm kiếm giải pháp tối ưu. **Chương 3: Kết quả và đánh giá** cung cấp các kết quả đạt được, bao gồm các sắp xếp tối ưu, so sánh hiệu quả giữa các phương pháp và phân tích thời gian thực thi của từng giải pháp. **Chương 4: Kết luận và hướng phát triển** tóm tắt các kết quả chính, rút ra bài học kinh nghiệm từ quá trình thực hiện, và đề xuất các hướng phát triển trong tương lai. Báo cáo không chỉ nhằm giải quyết bài toán một cách hiệu quả mà còn minh họa tính ứng dụng của các thuật toán tối ưu tổ hợp trong công nghệ thông tin và nhiều lĩnh vực liên quan khác.

# CHƯƠNG 1: TỔNG QUAN VỀ BÀI TOÁN

1. Lịch sử và bối cảnh

Bài toán sắp xếp quân mã trên bàn cờ xuất phát từ các bài toán tổ hợp cổ điển trong lý thuyết cờ vua, một trò chơi phổ biến từ thời Trung Cổ. Bài toán này là một biến thể từ bài toán "Hành trình quân mã" (**Knight’s Tour**), trong đó quân mã phải đi qua tất cả các ô trên bàn cờ mà không lặp lại. Về sau, bài toán được mở rộng thành nhiều dạng khác nhau, bao gồm việc sắp xếp tối đa số quân mã mà không quân nào khống chế nhau, trở thành một bài toán tối ưu tổ hợp thú vị và có tính thách thức cao trong toán học và khoa học máy tính.

1. Cơ sở lý thuyết
2. Định nghĩa bài toán

Bài toán sắp xếp quân mã tối ưu trên bàn cờ n×n yêu cầu tìm cách đặt tối đa số lượng quân mã sao cho không quân mã nào khống chế nhau. Quân mã trong cờ vua di chuyển theo hình chữ L, tức là đi 2 ô theo một hướng và 1 ô theo hướng vuông góc, hoặc ngược lại. Trên bàn cờ, mỗi quân mã chỉ được di chuyển trong phạm vi này, tạo thành mối quan hệ khống chế giữa các ô.

1. Mô hình hóa bằng lý thuyết đồ thị

Bài toán này có thể được mô hình hóa bằng đồ thị, trong đó:

* Các ô trên bàn cờ là các đỉnh (vertices).
* Các nước đi hợp lệ của quân mã tạo thành các cạnh (edges) nối các đỉnh.

Tập hợp các quân mã trên bàn cờ tương ứng với một tập các đỉnh trong đồ thị. Yêu cầu "không quân mã nào khống chế nhau" đồng nghĩa với việc các đỉnh trong tập hợp này không được nối với nhau bằng cạnh, tức là tạo thành một tập đỉnh độc lập (Independent Set). Bài toán trở thành tìm tập đỉnh độc lập lớn nhất (Maximum Independent Set) của đồ thị này.

1. Phân loại bài toán

Bài toán sắp xếp tối ưu quân mã thuộc nhóm bài toán NP-khó trong lý thuyết tính toán, nghĩa là không có thuật toán hiệu quả nào được biết để giải bài toán trong thời gian đa thức cho mọi giá trị nnn. Điều này khiến việc giải bài toán với kích thước bàn cờ lớn (n>8) trở thành thách thức lớn, đòi hỏi các phương pháp tối ưu hóa hoặc xấp xỉ hiệu quả.

1. Cơ sở công nghệ
2. Python

Python là một ngôn ngữ lập trình bậc cao, thông dịch, được tạo ra bởi Guido vanRossum và ra mắt lần đầu vào năm 1991. Python nổi tiếng với cú pháp rõ ràng, dễ đọc, và thiết kế tập trung vào khả năng mở rộng cũng như tính linh hoạt. Python thường được sử dụng trong nhiều lĩnh vực khác nhau.

1. Thư viện Tkinter

Tkinter là thư viện tiêu chuẩn của Python dùng để xây dựng giao diện đồ họa (GUI). Nó là một phần của Python kể từ phiên bản 1.1, và được tích hợp sẵn trong Python nên không cần cài đặt thêm.

1. Các phương pháp tiếp cận lý thuyết
2. Thuật toán Backtracking

Backtracking (tạm dịch: quay lui) là một chiến lược giải quyết bài toán dựa trên việc thử và loại bỏ các giải pháp không phù hợp. Thuật toán này xây dựng dần lời giải bằng cách mở rộng từng bước, và nếu phát hiện lời giải không khả thi, nó sẽ quay lại bước trước đó để thử lựa chọn khác.

1. Thuật toán Greedy

Thuật toán Greedy (tạm dịch: tham lam) là một phương pháp giải bài toán tối ưu hóa dựa trên việc lựa chọn các hành động tốt nhất tại mỗi bước (local optimum) với hy vọng rằng giải pháp toàn cục (global optimum) sẽ được tối ưu.

1. Thuật toán Hill Climbing

Hill Climbing là một thuật toán tối ưu hóa heuristic dựa trên phương pháp lặp, được sử dụng để tìm kiếm lời giải tốt nhất (hoặc gần tốt nhất) cho một bài toán. Thuật toán bắt đầu từ một trạng thái ban đầu, sau đó di chuyển tới trạng thái "láng giềng" tốt hơn, với mục tiêu đạt được đỉnh cao nhất trong không gian tìm kiếm (giá trị tối ưu).

1. Ứng dụng và tầm quan trọng

Bài toán này không chỉ mang ý nghĩa lý thuyết mà còn có nhiều ứng dụng thực tiễn trong lĩnh vực công nghệ thông tin. Các ứng dụng điển hình bao gồm sắp xếp cảm biến trong mạng IoT để tránh nhiễu lẫn nhau, lập kế hoạch đặt camera giám sát hoặc cảm biến sao cho không có vùng chồng lấn, hay phân bổ tài nguyên trong hệ thống máy tính và mạng lưới. Tuy nhiên, bài toán thuộc nhóm NP-khó, khiến việc giải quyết cho các bàn cờ kích thước lớn (n>8) vẫn là một thách thức lớn, đòi hỏi các thuật toán tối ưu hóa hiệu quả và tài nguyên tính toán đáng kể.

# CHƯƠNG 2: PHƯƠNG PHÁP GIẢI BÀI TOÁN

1. Thuật toán Backtracking
2. Mô tả thuật toán

Backtracking là phương pháp tìm kiếm tất cả các khả năng có thể của một bài toán thông qua việc thử nghiệm và quay lại nếu gặp phải trường hợp không thể tiếp tục. Trong bài toán này, thuật toán sẽ thử đặt quân mã vào các ô trên bàn cờ theo từng bước và quay lại nếu phát hiện ra rằng sắp xếp không hợp lệ.

1. Các bước thực hiện

**Bước 1:** bắt đầu từ ô đầu tiên trên bàn cờ

**Bước 2:** đặt quân mã vào ô đó và kiểm tra xem quân mã có bị khống chê bởi quân mã nào đã đặt không

**Bước 3:** nếu ô hợp lệ (không bị khống chế), tiếp tục đặt quân mã vào ô tiếp theo

**Bước 4:** nếu không tiếp tục (không có ô hợp lệ để đặt quân mã), quay lại ô trước đó và thử lại phương án khác.

**Bước 5:** tiếp tục cho đến khi tìm được sắp xếp tối ưu hoặc thử hết tất cả các khả năng

1. Mã giả

function Backtracking(board, n, current\_knights, max-knights, currentrow, currentcol):

max\_knights = max(max\_knights, current\_knights)

for row from currentrow to n-1:

for col from current\_col to n-1:

if board[row][col] == 0 AND is\_safe(board, row, col, n):

board[row][col] = 1

max\_knights = Backtracking(board, n, current\_knights + 1, max\_knights, row, col + 1)

// quay lui

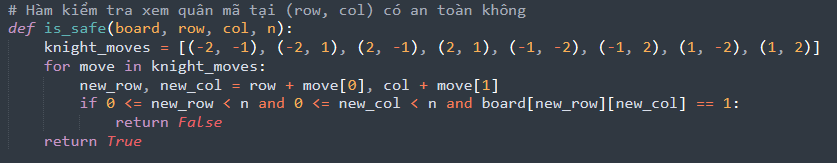
board[row][col] = 0

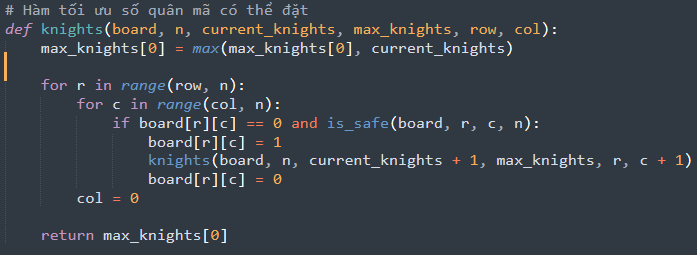
currentcol = 0

return max\_knights

1. Chương trình thuật toán

Theo kết quả tính toán từ những bài toán với n nhỏ thì số quân mã tối đa có thể đặt là:

 Kiểm tra quân mã có được đặt ở một vị trí hợp lệ hay không ta viết một hàm is\_safe nếu quân mã đặt ở một vị trí hợp lên ta trả về giá trị đúng(true), nếu không thì trả về giá trị sai(false), bằng cách kiểm tra từng vị trí mà quân mã đó có thể tấn công xem đã có quân mã nào đã được đặt vào vị trí đó hay chưa.

 Khởi tạo hàm Backtracking để sắp xếp quân mã gồm các tham số board (mảng 2 chiều), placed\_knights (số quân mã đã được đặt), max\_knights (quân mã tối đa được đặt), current\_row, current\_col, hàm này sẽ bắt đầu duyệt qua tất cả các ô trên bàn cờ, kiểm tra xem vị trí có bị khống chế không và đặt quân mã, quân mã được đặt sẽ tăng placed\_knights lên 1, đệ quy hàm nếu quân mã được đặt còn không thì quân mã sẽ bị xóa và giảm placed\_knights đi 1.

1. Thuật toán Greedy
2. Mô tả thuật toán

Thuật toán Greedy là một phương pháp tìm kiếm tham lam, trong đó thuật toán sẽ chọn lựa bước đi tốt nhất tại mỗi bước mà không xem xét các bước tiếp theo. Mục tiêu là tối đa hóa số lượng quân mã được sắp xếp trên bàn cờ mà không bị khống chế nhau.

1. Các bước thực hiện

**Bước 1:** Xếp quân mã vào ô có ít quân mã nhất có thể khống chế

**Bước 2:** Tiếp tục chọn các ô còn lại để đặt quân mã sao cho không bị khống chế

**Bước 3:** quá trình dừng lại khi không còn ô nào có thể đặt quân mã mà không bị khống chế

1. Mã giả

Hàm greedy\_knight (n):

Khởi tao bàn cờ (board) kích thước nxn, với tất cả các ô có giá trị ban đầu là 0

Đặt biến đếm Placed\_knight = 0 (đại diện cho số quân mã đã đặt)

For row = 0 to n-1:

For col = 0 to n-1:

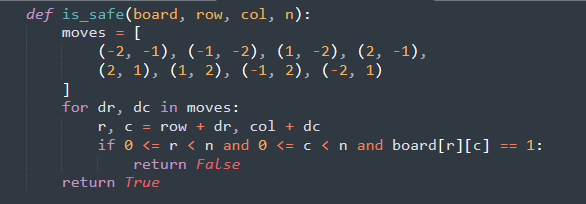
If vị trí (row, col) là hợp lệ để đặt quân mã:

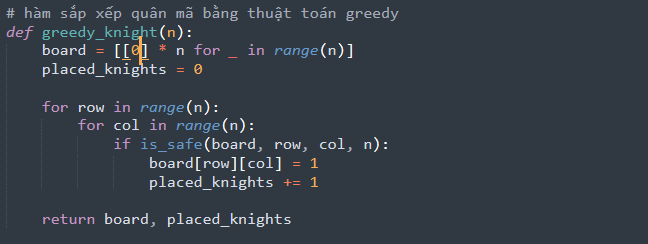
Đặt quân mã tại vị trí đó (board[row][col] = 1)

Tăng số quân mã đẵ dặt (Placed\_knight += 1)

Return board và Placed\_knight

1. Chương trình thuật toán

Khởi tạo hàm is\_safe để kiểm tra xem quân mã có được đặt ở vị trí hợp lệ không

Khởi tạo hàm sắp xếp quân mã bằng thuật toán greedy, bắt đầu khởi tạo bàn cờ bằng mảng 2 chiều với các ô có giá trị ban đầu bằng 0, cho biến đếm ban đầu bằng 0, kiểm tra từng vị trí trong bàn cờ nếu vị trí hợp lệ với hàm kiểm tra is\_safe thì đặt quân cờ tại vị trí đó và tăng biến đếm lên 1, lặp lại nó cho đến khi không còn vị trí nào hợp lệ có thể đặt quân mã nữa.

Hàm sẽ trả về giá trị bàn cờ (board) và số quân mã được đặt.

1. Thuật toán Hill Climbing
2. Mô tả thuật toán

Hill Climbing được coi là một trường hợp đặt biệt của thuật toán greedy, là một thuật toán tìm kiếm cục bộ, trong đó thuật toán luôn di chuyển về hướng có đánh giá tốt nhất trong không gian giải pháp. Mục tiêu của thuật toán là đạt được điểm cực trị cục bộ, nơi không thể cải thiện thêm nữa.

1. Các bước thực hiện

**Bước 1:** Khởi tạo và sắp xếp quân mã bất kỳ trên bàn cờ

**Bước 2:** Tính toán hàm đánh giá cảu sắp xếp hiện tại (số quân mã không bị khống chế)

**Bước 3:** Thử di chuyển một quân mã và tính toán lại hàm đánh giá. Chọn bước đi tốt nhất.

**Bước 4:** tiếp tục quá trình cho đến khi không thể cải thiện được nữa.

1. Mã giả

Hàm hill\_climbing(n):

Board ma trận nxn, chứa toàn số 0

Knights danh sách lưu vị trí các quân mã

Đặt max\_knights = 0 để lưu sô lượng quân mã tối đa tìm được

Khởi tạo best\_board để lưu bàn cờ tối ưu

Lặp lại 100 lần (giới hạn số lần):

Khởi tạo lại bàn cờ trống (board) và danh sách knight rỗng

Trong khi còn ô hợp lệ để dặt quân mã:

Tìm tất cả các ô hợp lên trên bàn cờ (candidates):

Một ô được coi là hợp lệ nó sẽ thỏa mãn hàm is\_safe

Nếu không còn ô hợp lệ:

Dừng vòng lặp

Chọn một ô ngẫu nhiên từ danh sách các ô hợp lệ (candidates)

Đặt quân mã vào ô đó (cập nhật board và danh sách knights)

Nếu số lượng quân mã trong danh sách knight lớn hơn max\_knights:

Cập nhật max\_knights = len(knights)

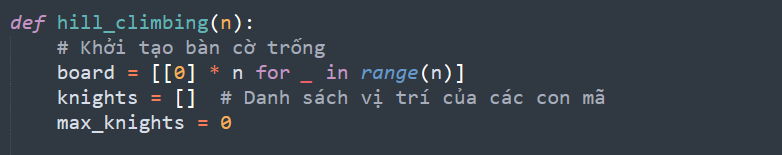
Sao chép bàn cờ hiện tại vào best\_board

Trả vê max\_knight và best\_board.

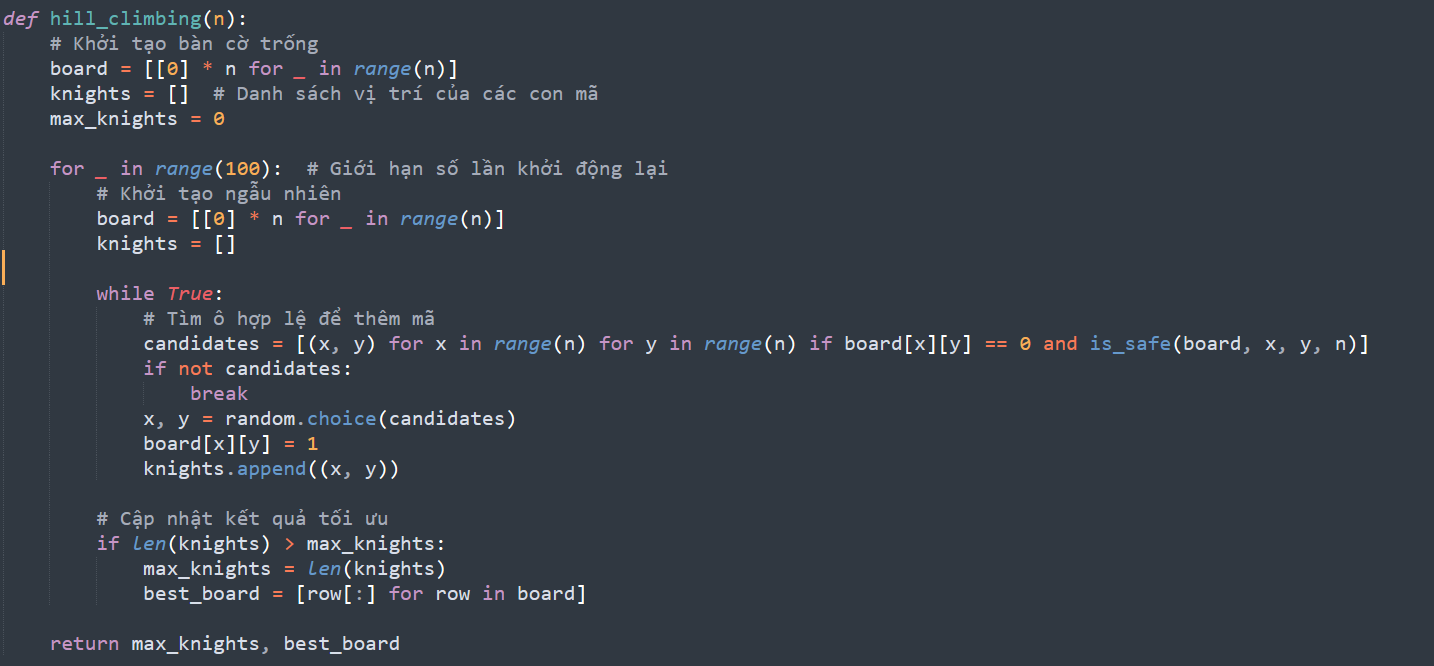
1. Chương trình thuật toán

Như mọi chương trình thuật toán khác, khởi tạo hàm is\_safe để kiểm tra vi trí đặt quân mã có hợp lệ không

Khởi tạo hàm hill\_climbing(n), bạn cờ bây giờ sẽ là một ma trận nxn với các ô bằng 0, khởi tạo một danh sách để chứa các vị trí mà quân mã được đặt và biến max\_knights chứa giá trị quân mã tối đa được đặt.



Thuật toán sẽ lặp lại 100 lần để có thế tìm được best\_board (bàn cờ có số quân mã lớn nhất), trong vòng lặp này chúng ta sẽ khởi tạo lại bàn cờ trống và danh sách rỗng, cứ mỗi lần bàn cờ đã không còn vị trí có thể đặt quân mã thì tự động sẽ khởi tạo lại bàn cờ và danh sách chứa vị trí quân mã, tiếp theo dùng vòng lặp while để đặt quân mã và cập nhật giá trị max\_knight và danh sách knights.

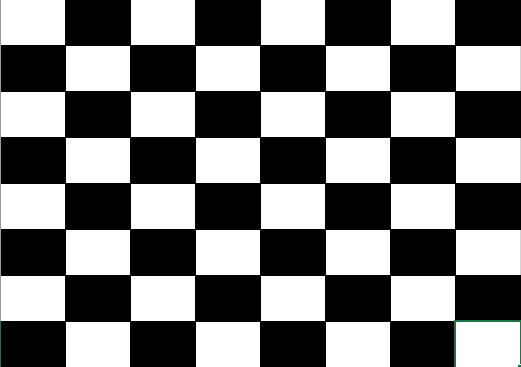


Trả về max\_knight và best\_board khi đã lặp đủ 100 lần là giá trị lớn nhất và tối ưu nhất trong 100 lần lặp.

1. Phương pháp sắp xếp tối ưu (nghiệm ra trong quá trình giải thuật toán)
2. Mô tả phương pháp

phương pháp này được nghiệm ra trong qua trình giải bài toán đó là, ta có các nước đi của quân mã là hình chữ L vậy quân mã có 8 phương án để duy chuyển nếu nằm ở vị trí lý tưởng như “hình 1”.

Theo như chúng ta đã biết thì bàn cờ vua sẽ gồm các ô vuông liên tiếp với nhau theo một ma trận nxn và có 2 màu nhất định là trắng và đen, từ đó ta sẽ nghiệm ra được điều đặt biệt là quân mã không thể tấn công quân mã khác ở ô cùng màu với nó có nghĩa là xung quanh quân mã chỉ cần đặt các quân mã khác ở ô cùng màu thì các quân mã không thể khống chế nhau được



Nhìn vào bàn cờ vua ta có thể thấy các ô cùng màu không nối tiếp nhau theo hàng và cột, mà nó liên tiếp nhau theo đường chéo các đường chéo cùng màu với nhau nên dù là bàn cờ n lớn đến đâu thì các đường chéo của bàn cờ cũng sẽ cùng một màu, nên việc đặt quân mã trên các đường chẻo để các quân mã trên bàn cờ không khống chế nhau là điều có thể xảy ra nhưng để biết kết quả có tối ưu không thì dựa vào cách tính số quân mã tối đa trên bàn cờ để thử thì biết được là kết quả tối ưu cho mọi bàn cờ.

1. Các bước thực hiện

**Bước 1:** Khởi tạo bàn cờ

**Bước 2:** xác định các ô có thể đặt quân mã..

**Bước 3:** đặt quân mã tại các ô đã xác định.

**Bước 4:** đếm số lượng quân mã và in kết quả.

1. Mã giả

Hàm place\_knight(n)

Tạo bàn cờ và danh sách vị trí quân mã đã đặt

For i=0 to n

For j=0 to n

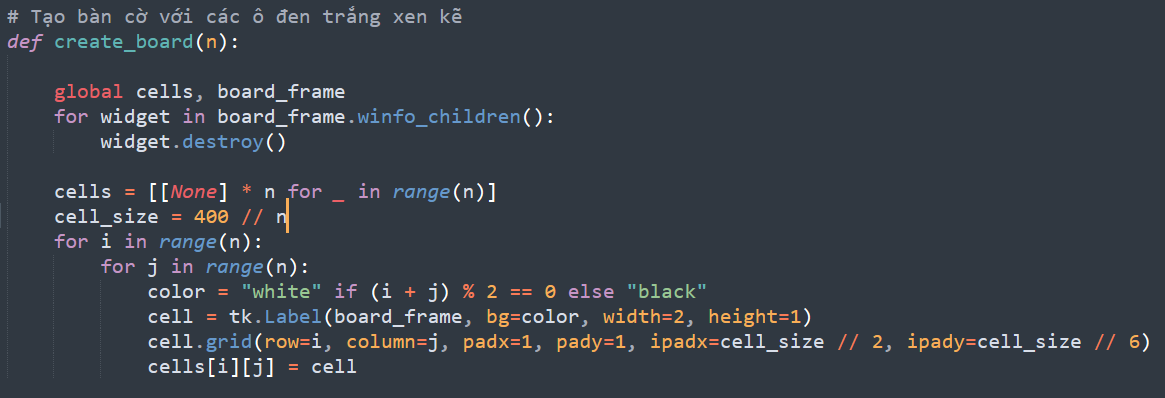
If (i +j ) mod 2 == 0

Thêm vị trí đặt quân mã vào danh sách

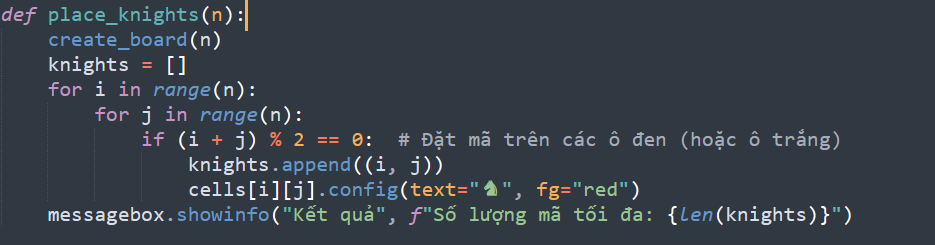
Đặt quân mã vào ô đó

Trả về số lượng quân mã tối đa bằng cách lấy độ dành của danh sách

1. Chương trình thuật toán

Khởi tạo hàm đẻ tạo bàn cờ với các ô trắng và đen xen kẽ bằng cách kiểm tra các ô bàn cờ nếu (i+j) mod 2 == 0 thì tô trắng ngược lại tô trắng

Tiếp đến khởi tạo hàm tính toán để đặt quân mã trên các ô cùng màu



Cuối cùng chúng ta hiển thị bàn cờ và các quân mã đã được đặt trên giao diện.

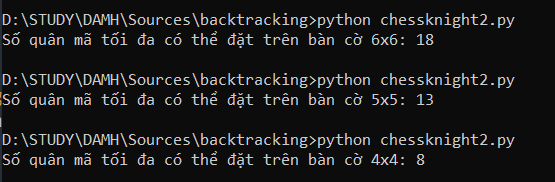
# CHƯƠNG 3: KẾT QUẢ VÀ ĐÁNH GIÁ

1. Kết quả thực nghiệm
2. Mô tả cấu hình thực nghiệm

Ngôn ngữ lập trình sử dụng là python với kích thước bàn cờ từ n>3, các thông số sử dụng trong chương trình thuật toán gồm: kích thước bàn cờ (n), bàn cờ (board), danh sách lưu trữ vị trí đặt quân cờ (knights), giá trị quân cờ tối đa được đặt (max\_knights),...

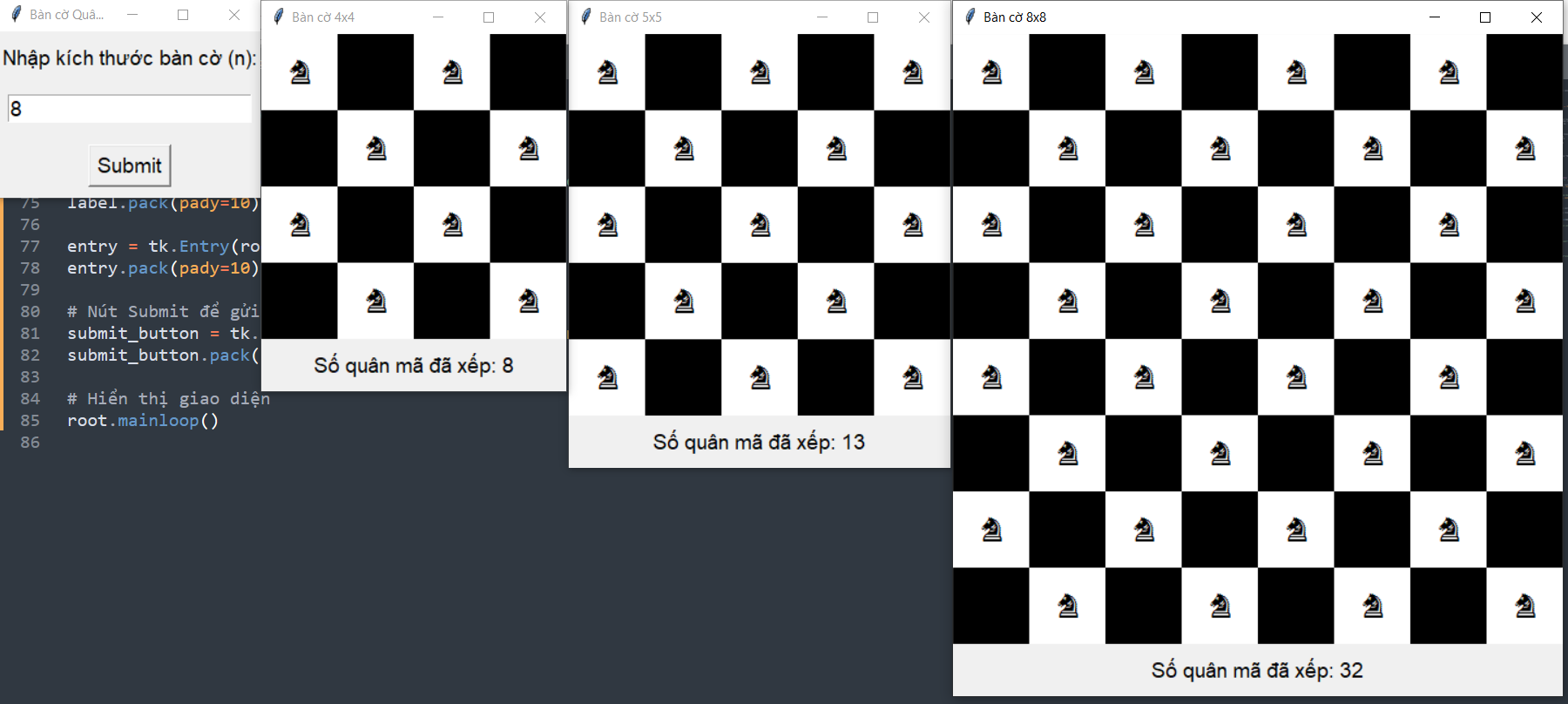
1. Kết quả chạy thử
2. Thuật toán Backtracking

Đối với chương trình thuật toán backtracking thì kết quả của bàn cờ với n từ 3->6 thì thuật toán đã đưa ra được kết quả số quân cờ tối đa chính xác, nhưng đến khi n>6 thì thời gian thực hiện lâu dẫn đến chương trình bị kẹt lại và không đưa ra được kết quả.



Lý do này có thể là do thuật toán backtracking chứa lời gọi đệ quy ở mộ bàn cờ n lớn hơn dẫn đến tràn ngăn xếp khiến chương trình bị kẹt lại không thể xử lý tiếp.

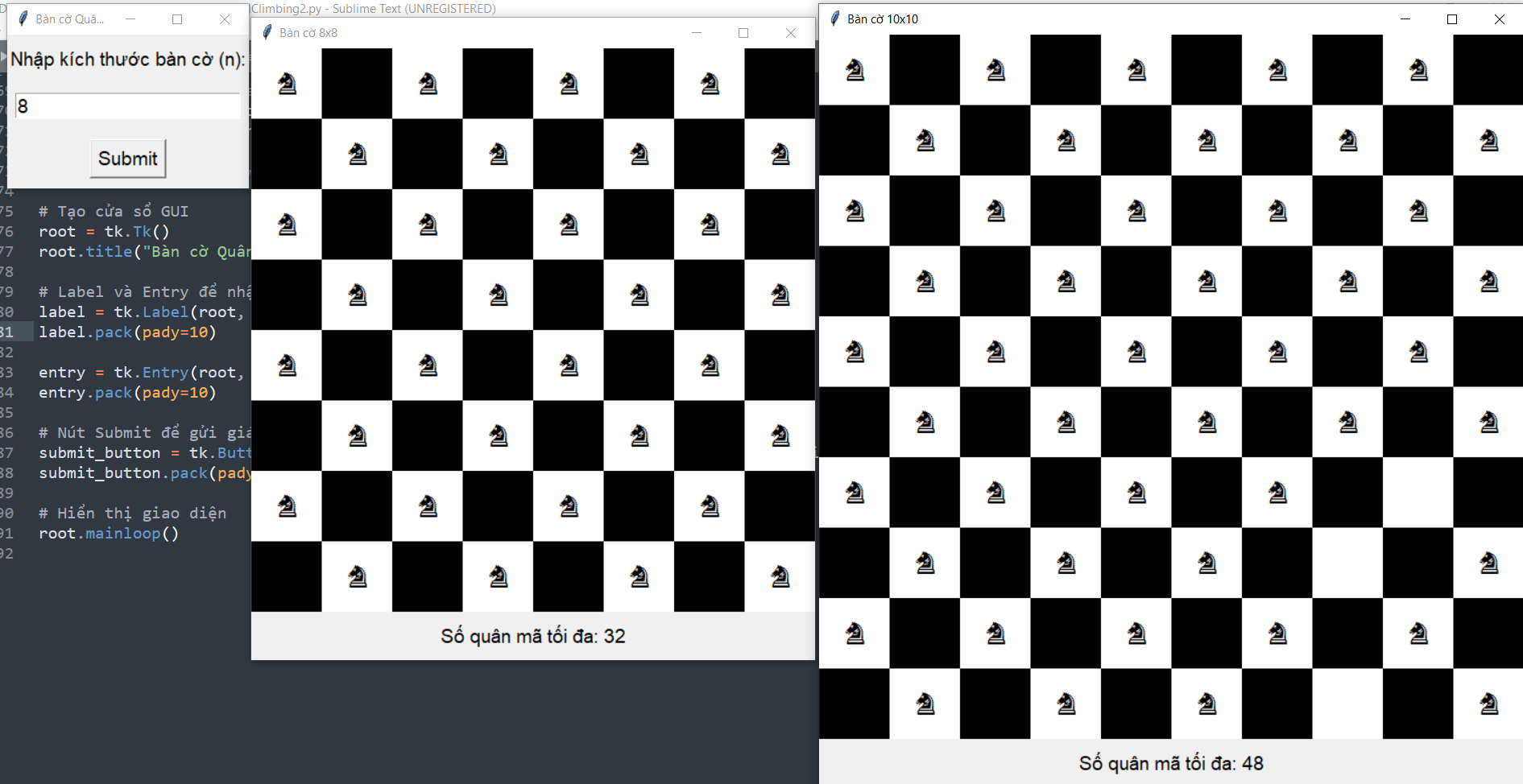
1. Thuật toán Greedy

Với chương trình thuật toán greedy thì kêt quả lúc này được tối ưu hơn khi chương trình có thể chạy đến một sô n rất lớn, nhưng khi n càng lớn thì kết quả đưa ra chỉ tương đối không có sự chính xác tuyệt đối.

Một phần lý do là vì khi kích cỡ bàn cờ càng lớn phạm vi tìm kiếm của thuật toán sẽ rộng hơn, các nước đi khống chế của quân mã sẽ tăng lên điều này khiến cho thuật toán greedy không thể tìm ra giải pháp tối ưu trong thời gian phức tạp hơn, cũng như càng nhiều quân mã được đặt, sẽ càng có nhiều ảnh hưởng tới các ô còn lại và thuật toán greedy sẽ bỏ qua nhiều sự lựa chọn tìm năng để tối ưu hóa quân mã.

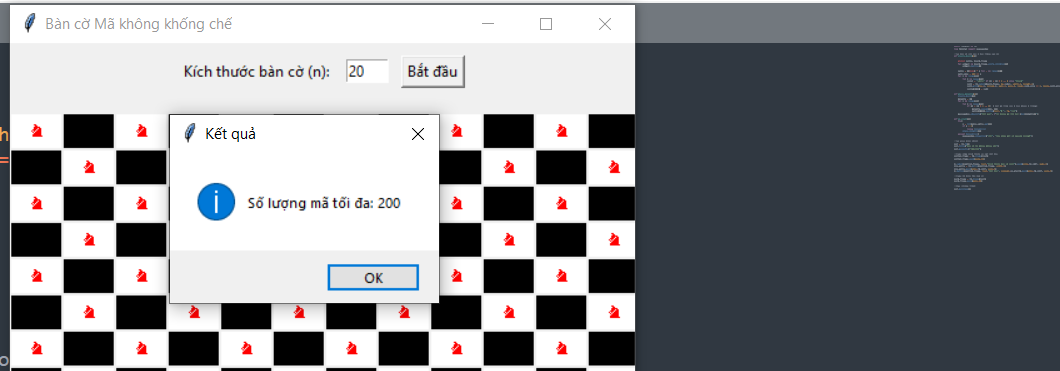
1. Thuật toán Hill Climbing

Đối với thuật toán hill climbing thì cũng giống như thuật toán greedy, nhưng với thuật toán hill climbing số quân mã dần được tối ưu hơn thuật toán greedy vì sự hoạt động của lặp lại thuật toán để tìm best\_board cho thấy hill\_climbing là phiên bản nâng cấp của thuật toán greedy.



1. Phương pháp sắp xếp tối ưu

Về phương pháp sắp xếp tối ưu đã nghiệm được trong quá trình giải bài toán thì đây là phương pháp đơn giản nhất không cần sử dụng thuật toán phức tạp, mà chỉ cần sắp xếp quân mã theo các cột chéo cùng màu với nhau thì số quân mã tối đa được đặt trên bàn cờ là giá trị tuyệt đối.



Dù cho con số n có lớn lên tới hàng trăm thì cách sắp xếp này vẫn đưa ra được kết quả chính xác nhất.

1. Đánh giá các thuật toán
2. So sánh hiệu suất

|  |  |  |  |
| --- | --- | --- | --- |
| Thuật toán | Hiệu quả | Tính khả thi | Thời gian thực thi |
| Backtracking | Kết quả có độ chính xác cao | Hoạt động kém với kích thước bàn cờ lớn | Thời gian dài hơn với kích thước bàn cờ lớn |
| Greedy | Kết quả có độ chính xác tương đối | Hoạt động tốt với bàn cờ lớn | Thời gian ngắn |
| Hill Climbing | Kết quả có độ chính xác tương đối | Hoạt động tốt với bàn cờ lớn | Thời gian ngắn |

1. Phân tích ưu và nhược điểm thực tế
2. Thuật toán Backtracking

* Ưu điểm:
* Đảm bảo thử mọi khả năng có thể tìm ra cách sắp xếp tối ưu, do đó kết quả là chính xác
* Phương pháp backtracking đơn giản, dễ hiểu, dễ triển khai.
* Nhược điểm:
* Hiệu suất chậm với bàn cở có kích thước n>8, do phải thử tất cả các tổ hợp có thể. Vì mỗi ô có 8 nước đi nên độ phức tạp tăng.
* Thuật toán sử dụng đệ quy nên khi số lượng ô lớn có thể gây ra lỗi tràn ngăn xếp

1. Thuật toán Greedy

* Ưu điểm:
* Thuật toán greedy đơn giản dễ triển khai
* Hiệu quả về thời gian, greedy phù hợp với bài toán n rất lớn do không yêu cầu quá nhiều tài nguyên tính toán
* Trong hầu hết các trường hợp greedy đưa ra được số lượng quân mã gần với số lượng tối đa với kích thước bàn cờ lớn.
* Nhược điểm:
* Khi bàn cờ lớn phạm vi tìm kiếm của thuật toán tăng dẫn đến kết quả khó được tối ưu
* Thứ tự duyệt của bàn cờ cũng ảnh hưởng đến kết quả

1. Thuật toán Hill Climbing

* Ưu điểm:
* Là phương pháp trực quan, dễ hiểu, bằng cách chọn cấu hình bàn cờ tối ưu nhất là nâng cấp của thuật toán greedy
* Tốc độ xử lý bài toán nhanh
* Nhược điểm:
* Không đảm bảo được tính tối ưu toàn cục
* Khi bàn cờ với n>10 không gian trạng thái trở nên phức tạp hơn và việc tìm kiếm có thể không tối ưu
* Kết quả không ổn định thay đổi tùy thuộc vào vị trí khởi tạo

1. Đánh giá tổng quan

ở những bàn cờ nhỏ nên dùng thuật toán backtracking để kết quả được chính xác và nhanh chóng.



Đối với những bàn cờ phức tạp hơn ta nên dùng thuật toán greedy để giải bài toán vì độ chính xác và thời gian thực hiện ngắn hơn.

Các yếu tố chính ảnh hưởng đến hiệu suất bài toán này chủ yếu là phụ thuộc vào giá trị n kích thước bàn cờ.

Thuật toán greedy sẽ là thuật toán được chọn ưu tiên để giải bài toán này vì thuật toán có thể kết hợp với các thuật toán khác để giải những bàn cờ phức tạp hoặc biến thể khác của bài toán.

1. Kết luận từ kết quả

Dựa vào các kết quả thực nghiệm và mô phỏng, ta có thể rút ra những kết luận sau:

1. Hiệu quả của các thuật toán

Backtracking cho thấy khả năng tìm kiếm chính xác các cấu hình tối ưu cho bàn cờ nhỏ (kích thước n≤7). Tuy nhiên, do độ phức tạp tăng theo cấp số nhân, thuật toán này không thể xử lý các bàn cờ lớn hơn trong giới hạn thời gian và bộ nhớ.So sánh giữa greedy và hill climbing

Greedy và HillClimbing đều thể hiện hiệu suất tốt với số lượng quân mã tối ưu tăng ổn định khi kích thước bàn cờ tăng. Hai thuật toán này có khả năng mở rộng tốt hơn cho các bài toán lớn.

1. Hạn chế của backtracking

Dễ bị giới hạn bởi tài nguyên tính toán, đặc biệt khi kích thước bàn cờ tăng, dẫn đến việc không thể tiếp tục giải quyết các bài toán n≥8.

1. ứng dụng của kết quả

Các thuật toán tối ưu tổ hợp như Greedy và Hill Climbing cung cấp giải pháp khả thi và nhanh chóng cho các bài toán tối ưu phức tạp trong thực tế.

Phương pháp phân tích và đánh giá từ bài toán này có thể áp dụng rộng rãi trong các lĩnh vực như phân bổ tài nguyên, lập lịch trình, và giải quyết bài toán đồ thị.

# CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. kết luận

Qua quá trình nghiên cứu và thực nghiệm, bài toán sắp xếp tối đa quân mã trên bàn cờ n×n đã đạt được các kết quả quan trọng sau:

Triển khai thành công các thuật toán Backtracking, Greedy, và HillClimbing, đồng thời thực hiện phân tích, so sánh hiệu suất của các thuật toán này trên bàn cờ với kích thước từ n=4 đến n=10.

Backtracking đạt kết quả chính xác với các bàn cờ nhỏ nhưng không thể mở rộng do hạn chế về thời gian và bộ nhớ.

Greedy và Hill Climbing đều tìm được số lượng quân mã tối ưu trên các bàn cờ lớn hơn, với hiệu suất vượt trội khi kích thước tăng.

Nghiệm ra được phương pháp tối ưu nhất của thuật toán làm cho bài toán dễ dàng triển khai chương trình tốt hơn mà không cần sử dụng nhiều đến các thuật toán đã phân tích.

Phân tích chi tiết đã làm rõ sự đánh đổi giữa độ chính xác, thời gian thực thi, và khả năng mở rộng của từng thuật toán, cung cấp góc nhìn toàn diện về việc áp dụng các thuật toán này trong thực tế.

Kết quả nghiên cứu cho thấy khả năng ứng dụng của các thuật toán tối ưu tổ hợp trong các bài toán tương tự như lập lịch trình, phân bổ tài nguyên, và giải quyết các bài toán tối ưu đồ thị khác.

1. Hướng phát triển
2. Cải tiến thuật toán

Có thể kết hợp các thuật toán greedy và hill climbing để tạo ra phương pháp lai để giảm nguy cơ kết quả không được tuyệt đối

Tối ưu hóa backtracking bằng cách áp dụng các chiến lược cắt tỉa mạnh hơn (BranchandBound).

1. ứng dụng thực tiễn

có các ứng dụng quan trọng trong AI về ngôn ngữ học máy, ứng dụng trong việc chọn tập các đặc trưng (feature selection), đảm bảo các đặc trưng được chọn không phụ thuộc lẫn nhau.

Trong lập trình game có thể ứng dụng lập trình AI cho quân mã trong cờ vua để di chuyển quân mã mà không bị khống chế bởi các quân khác, hoặc là sắp xếp đội hình để tránh bị xung đột tài nguyên của mỗi nhân vật.

Trong an toàn thông tin có thể ứng dụng để sắp xếp các nút mạng sao cho không có nút nào bị quá tải hoặc xung đột lẫn nhau, thiết kế cấu trúc mạng hiệu quả nhằm giảm thiểu giao thoa hoặc xung đột dữ liệu.

Bài toán còn được ứng dụng rất nhiều trong các ứng dụng quản lý phân bổ tài nguyên, đồ họa máy tính, thiết kế tự động,…

# TÀI LIỆU THAM KHẢO

1. [Bài toán N quân hậu và giải pháp đệ quy](https://codegym.vn/blog/moi-ngay-1-thuat-toan-bai-toan-n-quan-hau-va-giai-phap-de-quy/)
2. [Bài toán Hiệp Sĩ – Mã đi tuần (Knight’s Tour Problem)](https://codegym.vn/blog/thuat-toan/moi-ngay-1-thuat-toan-bai-toan-hiep-si-ma-di-tuan-knights-tour-problem/)
3. [Thuật toán tham lam - GeeksforGeeks](https://www.geeksforgeeks.org/greedy-algorithms/)
4. [Stack Overflow - Where Developers Learn, Share, & Build Careers](https://stackoverflow.com/)
5. Python 3.x và các thư viện: Numpy, Matplotlib để mô phỏng và trực quan hóa dữ liệu.
6. [Towards Data Science](https://towardsdatascience.com/)
7. [Hill Climbing in Artificial Intelligence - GeeksforGeeks](https://www.geeksforgeeks.org/introduction-hill-climbing-artificial-intelligence/)
8. [ChatGPT](https://chatgpt.com/)
9. [BÀI TOÁN “QUÂN MÃ ĐI TUẦN”](https://spiderum.com/bai-dang/BAI-TOAN-QUAN-MA-DI-TUAN-VA-NHUNG-DIEU-THU-VI-AN-SAU-NO-8ms)

# PHỤ LỤC