**TRƯỜNG ĐẠI HỌC ĐIỆN LỰC**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO CHUYÊN ĐỀ HỌC PHẦN**

**NHẬP MÔN TRÍ TUỆ NHÂN TẠO**

**ĐỀ TÀI:**

**ÁP DỤNG THUẬT TOÁN QUAY LUI VÉT CẠN GIẢI QUYẾT**

**BÀI TOÁN MÃ ĐI TUẦN**

|  |  |  |
| --- | --- | --- |
|  | |  |
| |  |  | | --- | --- | | **Sinh viên thực hiện** | **: NGUYỄN ĐÌNH MẠNH** | | **Giảng viên hướng dẫn** | **: PHẠM ĐỨC HỒNG** | | | **Ngành** | **: CÔNG NGHỆ THÔNG TIN** | | | **Chuyên ngành** | **: CÔNG NGHỆ PHẦN MỀM** | | | **Lớp** | **: D14CNPM8** | | | **Khóa** | **: 2019-2023** | | | |  |
|  |

***Hà Nội, tháng 12 năm 2021***

**TRƯỜNG ĐẠI HỌC ĐIỆN LỰC**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO CHUYÊN ĐỀ HỌC PHẦN**

**NHẬP MÔN TRÍ TUỆ NHÂN TẠO**

**ĐỀ TÀI:**

**ÁP DỤNG THUẬT TOÁN QUAY LUI VÉT CẠN GIẢI QUYẾT BÀI TOÁN MÃ ĐI TUẦN**

|  |  |  |
| --- | --- | --- |
|  | |  |
| |  |  | | --- | --- | | **Sinh viên thực hiện** | **: NGUYỄN ĐÌNH MẠNH** | | **Giảng viên hướng dẫn** | **: PHẠM ĐỨC HỒNG** | | | **Ngành** | **: CÔNG NGHỆ THÔNG TIN** | | | **Chuyên ngành** | **: CÔNG NGHỆ PHẦN MỀM** | | | **Lớp** | **: D14CNPM8** | | | **Khóa** | **: 2019-2023** | | | |  |
|  |

***Hà Nội, tháng 12 năm 2021***

**PHIẾU CHẤM ĐIỂM**

Sinh viên thực hiện:

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Họ và tên | Chữ ký | Nhiệm vụ |
| 1 | Nguyễn Đình Mạnh  Mã SV: 19810310580 |  |  |

Giảng viên chấm:

|  |  |  |
| --- | --- | --- |
| Họ và tên | Chữ ký | Ghi chú |
| Giảng viên chấm 1: |  |  |
| Giảng viên chấm 2: |  |  |

MỤC LỤC

[**DANH MỤC HÌNH ẢNH**](#_gjdgxs)[5](#_gjdgxs)

[**LỜI MỞ ĐẦU**](#_30j0zll) 6

[**CHƯƠNG 1: GIỚI THIỆU VỀ AI VÀ THUẬT TOÁN QUAY LUI VÉT CẠN**](#_3znysh7) 7

[**1.1.**](#_2et92p0) **GIỚI THIỆU VỀ TRÍ TUỆ NHÂN TẠO** 7

[**1.1.1.**](#_tyjcwt) **Trí tuệ nhân tạo là gì ?** 7

[**1.1.2.**](#_3dy6vkm) **Lịch sử** 8

[**1.1.3.**](#_1t3h5sf) **Lĩnh vực của AI** 9

[**1.2.**](#_4d34og8) **GIỚI THIỆU VỀ KỸ THUẬT QUAY LUI** 10

[**1.2.1.**](#_2s8eyo1) **Tư tưởng** 10

[**1.2.2.**](#_17dp8vu) **Heuristic** 11

[**1.2.3.**](#_3rdcrjn) **Phương pháp** 11

[**1.2.4.**](#_26in1rg) **Mô hình bài toán** 12

[**1.3.**](#_35nkun2) **KẾT LUẬN** 13

[**CHƯƠNG 2: BÀI TOÁN MÃ ĐI TUẦN** 15](#_1ksv4uv)

[**2.1 BÀI TOÁN** 15](#_44sinio)

[**2.1.1. Nước đi của quân Mã trên bàn cờ** 16](#_z337ya)

[**2.1.2. Xuất xứ bài toán** 17](#_3j2qqm3)

[**2.2 GIẢI QUYẾT BÀI TOÁN** 17](#_1y810tw)

[**2.2.1. Ý tưởng** 17](#_4i7ojhp)

[**2.2.2. Thiết kế thuật toán** 18](#_2xcytpi)

[**CHƯƠNG 3: CÀI ĐẶT, ĐÁNH GIÁ THỬ NGIỆM** 22](#_1ci93xb)

[**3.1 Cài đặt với C++** 22](#_3whwml4)

[**3.1.1 Code** 22](#_2bn6wsx)

[**3.1.2. Kết quả đạt được** 25](#_qsh70q)

[**3.1.3 Kết luận** 25](#_3as4poj)

[**KẾT LUẬN** 26](#_147n2zr)

**DANH MỤC HÌNH ẢNH**

[Hình 1. Cây mô tả thuật toán quay lui 9](#_23ckvvd)

Hình 2. Vị trí xuất phát của quân Mã trên bàn cờ 12

[Hình 3. Nước đi của quân Mã 13](#_ihv636)

[Hình 4. Mã ở rìa cũng giống như đồ trang trí 14](#_32hioqz)

[Hình 5. Các nước đi có thể có của ngựa 16](#_1hmsyys)

[Hình 6. Kết quả đạt được với C++ 22](#_41mghml)

# **LỜI MỞ ĐẦU**

Bài toán “ Mã đi tuần” là một bài toán thú vị, tương đối kinh điển của thuật toán quay lui vét cạn – 1 thuật toán quan trọng và phổ biến trong Trí tuệ Nhân tạo. Mã đi tuần (hay hành trình của quân mã) là bài toán về việc di chuyển một quân mã trên bàn cờ vua ( 8 x 8). Quân mã được đặt ở một ô trên một bàn cờ trống nó phải di chuyển theo quy tắc của cờ vua để đi qua mỗi ô trên bàn cờ đúng một lần.

Mặc dù đề tài đã hoàn thành, nhưng chắc chắn vẫn không thể tránh khỏi những thiếu sót, vì vậy chúng em mong muốn nhận được các ý kiến đóng góp của các thầy cô để có thể hoàn thiện hơn nữa.

***Chúng em xin chân thành cảm ơn!***

# **CHƯƠNG 1: GIỚI THIỆU VỀ AI VÀ THUẬT TOÁN QUAY LUI VÉT CẠN**

## **GIỚI THIỆU VỀ TRÍ TUỆ NHÂN TẠO**

* + 1. **Trí tuệ nhân tạo là gì ?**

Để hiểu trí tuệ nhân tạo (artificial intelligence) là gì chúng ta bắt đầu với khái niệm sự bay nhân tạo (flying machines), tức là cái máy bay.

Đã từ lâu, loài người mong muốn làm ra một cái máy mà có thể di chuyển được trên không trung mà không phụ thuộc vào địa hình ở dưới mặt đất, hay nói cách khác là máy có thể bay được. Không có gì ngạc nhiên khi những ý tưởng đầu tiên làm máy bay là từ nghiên cứu cách con chim bay. Những chiếc máy biết bay được thiết kế theo nguyên lý “vỗ cánh” như con chim chỉ có thể bay được quãng đường rất ngắn và lịch sử hàng không thực sự sang một trang mới kể từ anh em nhà Wright thiết kế máy bay dựa trên các nguyên lý của khí động lực học (aerodynamics).

Các máy bay hiện nay, như đã thấy, có sức trở rất lớn và bay được quãng đường có thể vòng quanh thế giới. Nó không nhất thiết phải có nguyên lý bay của con chim nhưng vẫn bay được như chim (dáng vẻ), và còn tốt hơn chim.

Quay lại câu hỏi Trí tuệ nhân tạo là gì. Trí tuệ nhân tạo là trí thông minh của máy do con người tạo ra. Ngay từ khi chiếc máy tính điện tử đầu tiên ra đời, các nhà khoa học máy tính đã hướng đến phát hiển hệ thống máy tính (gồm cả phần cứng và phần mềm) sao cho nó có khả năng thông minh như loài người. Mặc dù cho đến nay, theo quan niệm của người viết, ước mơ này vẫn còn xa mới thành hiện thực, tuy vậy những thành tựu đạt được cũng không hề nhỏ: chúng ta đã làm được các hệ thống (phần mềm chơi cờ vua chạy trên siêu máy tinh GeneBlue) có thể thắng được vua cờ thế giới; chúng ta đã làm được các phần mềm có thể chứng minh được các bài toán hình học; v.v. Hay nói cách khác, trong một số lĩnh vực, máy tính có thể thực hiện tốt hơn hoặc tương đương con người (tất nhiên không phải tất cả các lĩnh vực). Đó chính là các hệ thống thông minh.

Có nhiều cách tiếp cận để làm ra trí thông minh của máy (hay là trí tuệ nhân tạo), chẳng hạn là nghiên cứu cách bộ não người sản sinh ra trí thông minh của loài người nhưthế nào rồi ta bắt chước nguyên lý đó, nhưng cũng có những cách khác sử dụng nguyên lý hoàn toàn khác với cách sản sinh ra trí thông minh của loài người mà vẫn làm ra cái máy thông minh như hoặc hơn người; cũng giống như máy bay hiện nay bay tốt hơn con chim do nó có cơ chế bay không phải là giống như cơ chế bay của con chim.

Như vậy, trí tuệ nhân tạo ở đây là nói đến khả năng của máy khi thực hiện các công việc mà con người thường phải xử lý; và khi dáng vẻ ứng xử hoặc kết quả thực hiện của máy là tốt hơn hoặc tương đương với con người thì ta gọi đó là máy thông minh hay máy đó có trí thông minh. Hay nói cách khác, đánh giá sự thông minh của máy không phải dựa trên nguyên lý nó thực hiện nhiệm vụ đó có giống cách con người thực hiện hay không mà dựa trên kết quả hoặc dáng vẻ ứng xử bên ngoài của nó có giống với kết quả hoặc dáng vẻ ứng xử của con người hay không.

Các nhiệm vụ của con người thường xuyên phải thực hiện là: giải bài toán (tìm kiếm, chứng minh, lập luận), học, giao tiếp, thể hiện cảm xúc, thích nghi với môi trường xung quanh, v.v., và dựa trên kết quả thực hiện các nhiệm vụ đó để kết luận rằng một ai đó có là thông minh hay không. Môn học Trí tuệ nhân tạo nhằm cung cấp các phương pháp luận để làm ra hệ thống có khả năng thực hiện các nhiệm vụ đó: giải toán, học, giao tiếp, v.v. bất kể cách nó làm có như con người hay không mà là kết quả đạt được hoặc dáng vẻ bên ngoài như con người.

Trong môn học này, chúng ta sẽ tìm hiểu các phương pháp để làm cho máy tính biết cách giải bài toán, biết cách lập luận, biết cách học, v.v.

* + 1. **Lịch sử**

Vào năm 1943, Warren McCulioch và Walter Pitts bắt đầu thực hiện nghiên cứu ba cơ sở lý thuyết cơ bản: triết học cơ bản và chức năng của các noron thần kinh; phân tích các mệnh đề logic; và lý thuyết dự đoán của Turing. Các tác giả đã nghiên cứu đề xuât mô hình noron nhân tạo, mỗi noron đặc trưng bởi hai trạng thái “bật”, “tắt” và phát hiện mạng noron có khả năng học.

Thuật ngữ “Trí tuệ nhân tạo” (Artificial Intelligence - AI) được thiết lập bởi John McCarthy tại Hội thảo đầu tiên về chủ đề này vào mùa hè năm 1956. Đồng thời, ông cũng đề xuất ngôn ngữ lập trình Lisp – một trong những ngôn ngữ lập trình hàm tiêu biểu, được sử dụng trong lĩnh vực AI. Sau đó, Alan Turing đưa ra "Turing test" như là một phương pháp kiểm chứng hành vi thông minh.

Thập kỷ 60, 70 Joel Moses viết chương trình Macsyma - chương trình toán học sử dụng cơ sở tri thức đầu tiên thành công. Marvin Minsky và Seymour Papert đưa ra các chứng minh đầu tiên về giới hạn của các mạng nơ-ron đơn giản. Ngôn ngữ lập trình logic Prolog ra đời và được phát triển bởi Alain Colmerauer. Ted Shortliffe xây dựng thành công một số hệ chuyên gia đầu tiên trợ giúp chẩn đoán trong y học, các hệ thống này sử dụng ngôn ngữ luật để biểu diễn tri thức và suy diễn.  
 Vào đầu những năm 1980, những nghiên cứu thành công liên quan đến AI như các hệ chuyên gia (expert systems) – một dạng của chương trình AI mô phỏng tri thức và các kỹ năng phân tích của một hoặc nhiều chuyên gia con người.  
 Vào những năm 1990 và đầu thế kỷ 21, AI đã đạt được những thành tựu to lớn nhất, AI được áp dụng trong logic, khai phá dữ liệu, chẩn đoán y học và nhiều lĩnh vực ứng dụng khác trong công nghiệp. Sự thành công dựa vào nhiều yếu tố: tăng khả năng tính toán của máy tính, tập trung giải quyết các bài toán con cụ thể, xây dựng các mối quan hệ giữa AI và các lĩnh vực khác giải quyết các bài toán tương tự, và một sự chuyển giao mới của các nhà nghiên cứu cho các phương pháp toán học vững chắc và chuẩn khoa học chính xác.

* + 1. **Lĩnh vực của AI**
* Lập luận, suy diễn tự động: Khái niệm lập luận (reasoning), và suy diễn (reference) được sử dụng rất phổ biến trong lĩnh vực AI. Lập luận là suy diễn logic, dùng để chỉ một tiến trình rút ra kết luận (tri thức mới) từ những giả thiết đã cho (được biểu diễn dưới dạng cơ sở tri thức). Như vậy, để thực hiện lập luận người ta cần có các phương pháp lưu trữ cơ sở tri thức và các thủ tục lập luận trên cơ sở tri thức đó.
* Biểu diễn tri thức: Muốn máy tính có thể lưu trữ và xử lý tri thức thì cần có các phương pháp biểu diễn tri thức. Các phương pháp biểu diễn tri thức ở đây bao gồm các ngôn ngữ biểu diễn và các kỹ thuật xử lý tri thức. Một ngôn ngữ biểu diễn tri thức được đánh giá là “tốt” nếu nó có tính biểu đạt cao và các tính hiệu quả của thuật toán lập luận trên ngôn ngữ đó. Tính biểu đạt của ngôn ngữ thể hiện khả năng biểu diễn một phạm vi rộng lớn các thông tin trong một miền ứng dụng. Tính hiệu quả của các thuật toán lập luận thể hiện chi phí về thời gian và không gian dành cho việc lập luận. Tuy nhiên, hai yếu tố này dường như đối nghịch nhau, tức là nếu ngôn ngữ có tínhbiểu đạt cao thì thuật toán lập luận trên đó sẽ có độ phức tạp lớn (tính hiệu quả thấp)và ngược lại (ngôn ngữ đơn giản, có tính biểu đạt thấp thì thuật toán lập luận trên đó sẽ có hiệu quả cao). Do đó, một thách thức lớn trong lĩnh vực AI là xây dựng các ngôn ngữ biểu diễn tri thức mà có thể cân bằng hai yếu tố này, tức là ngôn ngữ có tínhbiểu đạt đủ tốt (tùy theo từng ứng dụng) và có thể lập luận hiệu quả.
* Lập kế hoạch: khả năng suy ra các mục đích cần đạt được đối với các nhiệm vụ đưa ra, và xác định dãy các hành động cần thực hiện để đạt được mục đích đó.
* Học máy: là một lĩnh vực nghiên cứu của AI đang được phát triển mạnh mẽ và có nhiều ứng dụng trong các lĩnh vực khác nhau như khai phá dữ liệu, khám phá tri thức,…
* Xử lý ngôn ngữ tự nhiên: là một nhánh của AI, tập trung vào các ứng dụng trên ngôn ngữ của con người. Các ứng dụng trong nhận dạng tiếng nói, nhận dạng chữ viết, dịch tự động, tìm kiếm thông tin,…
* Hệ chuyên gia: cung cấp các hệ thống có khả năng suy luận để đưa ra những kết luận. Các hệ chuyên gia có khả năng xử lý lượng thông tin lớn và cung cấp các kết luận dựa trên những thông tin đó. Có rất nhiều hệ chuyên gia nổi tiếng như các hệ chuyên gia y học MYCIN, đoán nhận cấu trúc phân tử từ công thức hóa học DENDRAL, …
* Robotics
* …

## **GIỚI THIỆU VỀ KỸ THUẬT QUAY LUI**

Thuật toán quay lui vét cạn (Backtracking) là một kĩ thuật thiết kế giải thuật dựa trên đệ quy. Ý tưởng của quay lui là tìm lời giải từng bước, mỗi bước chọn một trong số các lựa chọn khả dĩ và đệ quy. Người đầu tiên đề ra thuật ngữ này (backtrack) là nhà toán học người Mỹ D. H. Lehmer vào những năm 1950.

* + 1. **Tư tưởng**

Nét đặc trưng của phương pháp quay lui là các bước hướng tới lời giải cuối cùng của bài toán đều được làm thử. Tại mỗi bước, nếu có một lựa chọn được chấp nhận thì ghi lại lựa chọn này và tiến hành các bước thử tiếp theo. Còn ngược lại không có lựa chọn nào thích hợp thì làm lại bước trước, xóa bỏ sự ghi nhận và quay về chu trình thử các lựa chọn còn lại.

Hành động trên được gọi là quay lui, thuật toán thể hiện phương pháp này gọi là quay lui.

Điểm quan trọng của thuật toán là phải ghi nhớ tại mỗi bước đi qua để tránh trùng lặp khi quay lui. Các thông tin này được lưu trữ trong một ngăn xếp, nên thuật toán thể hiện ý thiết kế một cách đệ quy.

* + 1. **Heuristic**

Người ta thường sử dụng một số phương pháp heuristic để tăng tốc cho quá trình quay lui. Do các biến có thể được xử lý theo thứ tự bất kỳ, việc thử các biến bị ràng buộc chặt nhất (nghĩa là các biến có ít lựa chọn về giá trị nhất) thường có hiệu quả do nó tỉa cây tìm kiếm từ sớm (cực đại hóa ảnh hưởng của lựa chọn sớm hiện hành).

Các cài đặt quay lui phức tạp thường sử dụng một hàm biên, hàm này kiểm tra xem từ lời giải chưa đầy đủ hiện tại có thể thu được một lời giải hay không, nghĩa là nếu đi tiếp theo hướng hiện tại thì liệu có ích hay không. Nhờ đó, một kiểm tra biên phát hiện ra các lời giải dở dang chắc chắn thất bại có thể nâng cao hiệu quả của tìm kiếm. Do hàm này hay được chạy, có thể tại mỗi bước, nên chi phí tính toán các biên cần tối thiểu, nếu không, hiệu quả toàn cục của thuật toán sẽ không được cải tiến. Các hàm kiểm tra biên được tạo theo kiểu gần như các hàm heuristic khác: nới lỏng một số điều kiện của bài toán.

* + 1. **Phương pháp**

Giả sử có cấu hình cần liệt kê gồm n phần tử. Bắt đầu từ phần tử thứ 1. Khi đó thuật toán quay lui sẽ thực hiện như sau :

Các bước trong việc liệt kê cấu hình dạng X[1...n]:

* Xét tất cả các giá trị X[1] có thể nhận, thử X[1] nhận các giá trị đó. Với mỗi giá trị của X[1] ta sẽ:
* Xét tất cả giá trị X[2] có thể nhận, lại thử X[2] cho các giá trị đó. Với mỗi giá trị X[2] lại xét khả năng giá trị của X[3]...tiếp tục như vậy cho tới bước:
* ...
* ...
* Xét tất cả giá trị X[n] có thể nhận, thử cho X[n] nhận lần lượt giá trị đó.

Thông báo cấu hình tìm được.

Có thể hiểu đơn giản như sau, Thuật toán quay lui thiết kế cấu hình n phần tử bằng cách cho phần tử đầu tiên nhận giá trị có thể. Với mỗi giá trị đó lại tiếp tục liệt kê tiếp cấu hình n – 1 từ phần tử thứ 2. Bản chất của quay lui là một quá trình tìm kiếm theo chiều sâu(Depth-First Search).

* + 1. **Mô hình bài toán**

Thuật toán quay lui có thể được mô tả bằng đoạn mã giả (pseudocode) sau:

*// Cho phần tử x[i] nhận các giá trị mà nó có thể nhận*

Try( giá trị i)

for ( mọi giá trị V mà có thể gán cho phần tử đang xét x[i] )

thử gán x[i] = v ;

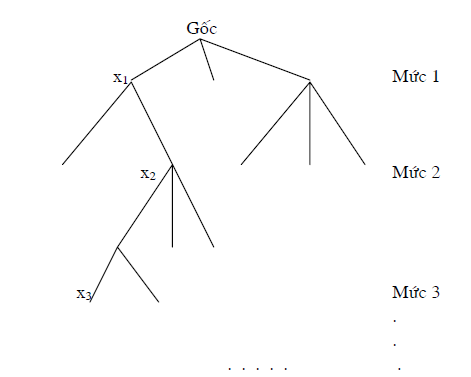
if (x[i] là phần tử kết thúc cấu hình) in ra cấu hình else

<ghi nhận việc đã gán x[i] = v nếu cần thiêt>

Try(i+1);

<bỏ ghi nhận việc gán x[i] = v nếu cần thiết>

Ta có thể trình bày quá trình tìm kiếm lời giải quả thuật toán quay lui bằng cây sau:



***Hình 1****: Cây mô tả thuật toán quay lui*

Vấn đề khó nhất khi thiết kế thuật toán dạng quay lui đó là tìm ra tập các lựa chọn có thể trong mỗi bước. Tập lựa chọn này sẽ ảnh hưởng đến tính chính xác cũng như độ phức tạp (thời gian cũng như bộ nhớ) của thuật toán quay lui. Để minh hoạ, ta xét một vài ví dụ về thiết kế thuật toán theo kĩ thuật quay lui:

## **KẾT LUẬN**

- Ưu điểm: Việc quay lui là thử tất cả các tổ hợp để tìm được một lời giải. Thế mạnh của phương pháp này là nhiều cài đặt tránh được việc phải thử nhiều trường hợp chưa hoàn chỉnh, nhờ đó giảm thời gian chạy.

- Nhược điểm: Trong trường hợp xấu nhất độ phức tạp của quay lui vẫn là cấp số mũ. Vì nó mắc phải các nhược điểm sau:

+ Rơi vào tình trạng "thrashing": qúa trình tìm kiếm cứ gặp phải bế tắc với cùng một nguyên nhân.

+ Thực hiện các công việc dư thừa: Mỗi lần chúng ta quay lui, chúng ta cần phải đánh giá lại lời giải trong khi đôi lúc điều đó không cần thiết.

+ Không sớm phát hiện được các khả năng bị bế tắc trong tương lai. Quay lui chuẩn, không có cơ chế nhìn về tương lai để nhận biết đc nhánh tìm kiếm sẽ đi vào bế tắc.

# **CHƯƠNG 2: BÀI TOÁN MÃ ĐI TUẦN**

## **2.1 BÀI TOÁN**

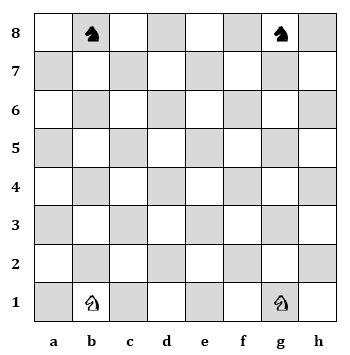
Mã đi tuần (hay hành trình của quân mã) là bài toán về việc di chuyển một quân mã trên bàn cờ vua (8 x 8). Quân mã được đặt ở một ô trên một bàn cờ trống (có toạ độ bất kì), nó phải di chuyển theo quy tắc của cờ vua để đi qua tất cả các ô của bàn cờ, mỗi ô đi qua đúng một lần.

Nếu một quân mã đi hết 64 vị trí và tại vị trí cuối cùng có thể di chuyển đến vị

trí bắt đầu thông qua một nước cờ thì đó gọi là một hành trình đóng

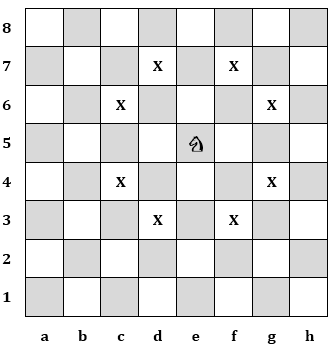
Có những hành trình, trong đó quân mã sau khi đi hết tất cả 64 ô của bàn cờ và từ ô cuối của hành trình không thể đi về ô xuất phát chỉ bằng một nước đi. Những hành trình như vậy được gọi là hành trình mở. Có rất nhiều lời giải cho bài toán này, chính xác là 26.534.728.821.064 lời giải trong đó quân mã có thể kết thúc tại chính ô mà nó khởi đầu.

## **2.1.1. Nước đi của quân Mã trên bàn cờ**



***Hình 2:****Vị trí xuất phát của quân Mã trên bàn cờ*

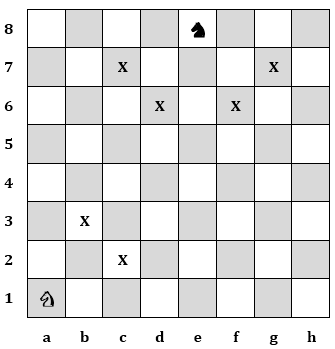
Trong cờ vua, quân Mã là quân có cách đi phức tạp nhất. Xét một quân Mã đang đứng trên bàn cờ và tất cả các hình chữ nhật 2 x 3 nhận ô mà quân Mã đó đang đứng làm đỉnh. Quân Mã đó có thể đi tới các đỉnh khác màu với đỉnh nó đang đứng của bất kì hình chữ nhật 2 x 3 nào, miễn là đỉnh đó không nằm ngay cạnh đỉnh nó đang đứng.



***Hình 3:****Nước đi của quân Mã*

Quân Mã có thể nhảy qua tất cả các quân khác để đến ô nó muốn, miễn là ô đó chưa bị ai chiếm giữ. Nói nôm na là quân Mã không bị cản. Khác với cờ tướng, nơi mà quân Mã có thể bị cản nếu có quân nào đứng ngay trước mặt nó, trong cờ vua, nước đi của quân Mã không có tính chất này.

Khi một quân Mã đứng ở cạnh bàn cờ, số nước đi có thể của nó sẽ bị thu hẹp xuống còn nhiều nhất là một nửa số nước đi ban đầu. Đặc biệt, nếu nó đứng ở một trong bốn góc bàn cờ, nó chỉ đi được tối đa hai nước. Câu nói “Mã ở rìa cũng giống như đồ trang trí” từ đây mà ra.



***Hình 4:****Mã ở rìa cũng giống như đồ trang trí*

**2.1.2. Xuất xứ bài toán**

Bài toán quân Mã đi tuần là một dạng của bài toán tổng quát hơn là bài toán tìm đường đi Hamilton trong l‎ý thuyết đồ thị, là một bài toán NP-đầy đủ. Bài toán tìm hành trình đóng của quân mã là một bài toán cụ thể của bài toán tìm chu trình hamiltonian.

Nhiều biến thể của chủ đề này được các nhà toán học nghiên cứu, trong đó có nhà toán học Euler. Các biến đổi có thể theo các hướng:

* Thay đổi kích thước bàn cờ.
* Biến thành trò chơi hai người theo tư tưởng này.

- Giảm nhẹ các yêu cầu trên đường đi của quân Mã.

## **2.2 GIẢI QUYẾT BÀI TOÁN**

### **2.2.1. Ý tưởng**

Có nhiều cách giải cho bài toán ngựa đi tuần. Nhưng ở đây ta sử dụng thuật toán quay lui để giải bài toán này. Xuất phát từ 1 ô, gọi số nước đi là t = 1, ta cho quân mã thử đi tiếp 1 ô (có 8 nước đi có thể), nếu ô đi tiếp này chưa đi qua thì chọn làm bước đi tiếp theo. Tại mỗi nước đi kiểm tra xem tổng số nước đi bằng n x n chưa, nếu bằng thì mã đã đi qua tất cả các ô ⇒ dừng (do chỉ cần tìm một giải pháp). Trường hợp ngược lại, gọi đệ quy để chọn nước đi tiếp theo. Ngoài ra, nếu tại một bước tìm đường đi, nếu không tìm được đường đi tiếp thì thuật toán sẽ quay lui lại nước đi trước và tìm đường đi khác.

### **2.2.2. Thiết kế thuật toán**

Ta có sơ đồ phác thảo sau:

Try(i) ≡

for ( j = 1 🡪 k)

If(xi chấp nhận được khả năng k)

{

Xác định xi theo khả năng k;

Ghi nhận trạng thái mới;

if( I < n2 )

Try(i + 1);

else

Ghi nhận nghiệm;

Trả lại trạng thái cũ cho bài toán;

}

Các thao tác để giải bài toán ngựa đi tuần:

* + - Biểu diễn bàn cờ.
    - Các khả năng lựa chọn cho xi.
    - Cách thức xác định xi theo j.
    - Ghi nhận nghiệm.

Cấu trúc dữ liệu : Ta biểu diễn bàn cờ bằng một ma trận vuông cấp n:

int h[n][n];

Ta có qui ước sau:

h[x][y] = 0 ≡ Ô <x,y> ngựa chưa đi qua;

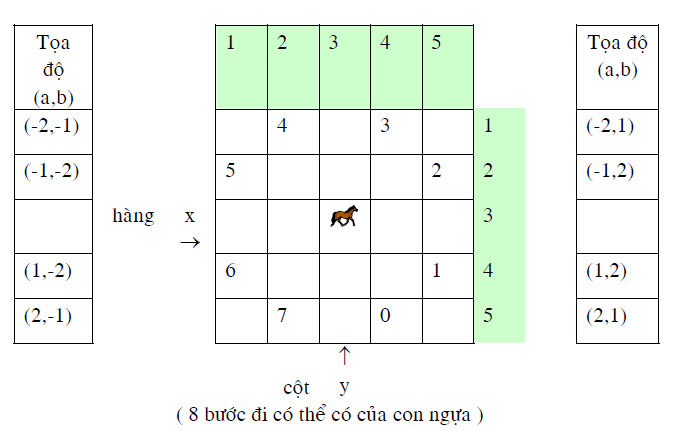
h[x][y] = I ≡ Ô <x,y> ngựa đã đi qua ở bước i ( 1≤ i ≤ n2 )

Với tọa độ bắt đầu là <x,y> như trong hình vẽ, ta sẽ có tất cả 8 ô <u,v> mà con ngựa có thể đi đến và được đánh số từ 0 đến 7.

Để có u, v từ x, y thì ta dùng 2 mảng a và b lưu trữ các sai biệt về tọa độ. Nếu dùng chỉ số k để đánh số “Bước đi kế” thì chi tiết đó được thể hiện bởi :

u= x + a[k]; v= y + b[k]; 0≤ k≤ 7;

\* Các khả năng lựa chọn xi: chính là các nước đi của ngựa mà xi  có thể chấp nhận được.



***Hình 5:*** *Các nước đi có thể có của ngựa*

Một phương pháp đơn giản để có được u, v từ x, y là ta dùng 2 mảng a và b lưu trữ các sai biệt về toạ độ. Nếu chỉ dùng một chỉ số k để đánh số “bước đi kế tiếp” thì chi tiết đó được thể hiện bởi:

u = x + a[k]; v = y + b[k]; 0 ≤ k ≤ 7;

Điều kiện chấp nhận được:

Ô mới phải thuộc bàn cờ (1 ≤ u ≤ n và 1 ≤ v ≤ n) và chưa đi qua ô đó, nghĩa là h[u, v] = 0;

Để ghi nhận nước đi hợp lệ ở bước I, ta gán h[u][v] = i; và để huỷ một nước đi thì ta gán h[u][v] = 0;

Ma trận h ghi nhận kết quả nghiệm. Nếu có <x, y> sao cho h<x, y> = 0 thì đó không phải lời giải có bài toán, ngược lại h chứa đường đi của ngựa.

Vậy thuật toán có thể mô tả như sau:

Input n; //Kích thước bàn cờ

x, y; Toạ độ xuất phát ở bưới i

Output h;

Mô tả:

Try (i, x, y) ≡

for(k = 0; k <= 7; k ++)

{

u = x + a[k];

v = y + b[k];

if(1 <= u && v <=n && h[u][v] == 0)

{

h[u][v] = i;

if(i < n \* n)

Try(i + 1, u, v);

else

xuat\_h(); // In ma trận h

}

h[u][v] = 0;

}

# **CHƯƠNG 3: CÀI ĐẶT, ĐÁNH GIÁ THỬ NGIỆM**

## **3.1 Cài đặt với C++**

### **3.1.1 Code**

**Bước 1**: Khai báo thư viện và các giá trị ban đầu

#include <iostream>

#define max 15

using namespace std;

**Bước 2**: Khai báo prototype

void nhap();

int quaylui(int,int,int);

void xuat();

**Bước 3**: Khai báo biến toàn cục, viết hàm nhập, xuất

int n, x, y; //Kích thước bàn cờ, ô xuất phát (x, y)

int a[8]= {2,1,-1,-2,-2,-1,1,2};

int b[8]= {1,2,2,1,-1,-2,-2,-1};

int flag[max+1][max+1];

void nhap()

{

//Nhập kích thước bàn cờ và ô xuất phát

cout << "Nhap vao kich thuoc ban co n: ";

cin >> n;

cout <<"Nhap vao toạ độ ô xuất phát (x, y): ";

cin >> x >> y;

//Khởi tạo

for(int i = 1; i <= n; i++)

for(int j = 1; j <= n; j++)

flag[i][j] = 0; //Moi ô đều tự do

flag[x][y] = 1; //Bắt đầu từ ô (x, y)

}

//In ra ma trận hành trình

void xuat()

{

for(int i = 1; i <= n; i++)

{

for(int j = 1; j <= n; j++)

cout << flag[i][j] << "\t";

cout << "\n";

}

}

**Bước 4**: Hàm Quay lui: Đặt quân mã vào ô tiếp theo ở bước thứ i

int quaylui(int i, int x, int y)

{

for(int j = 0; j < 8; j++)

{

//Đặt quân mã vào ô (u, v)

int u, v;

u = x + a[j];

v = y + b[j];

if ((u > 0 && u <= n) && (v > 0 && v <= n) && flag[u][v] == 0)

{

flag[u][v] = i; //ô (u, v) là ô tiếp theo

if (i == (n \* n))

{

xuat();

exit(0); //Chỉ đưa ra 1 hành trình

}

else

{

quaylui(i + 1, u, v);

flag[u][v] = 0;

}

}

}

return 0;

}

**Bước 5**: Chương trình chính

int main()

{

nhap();

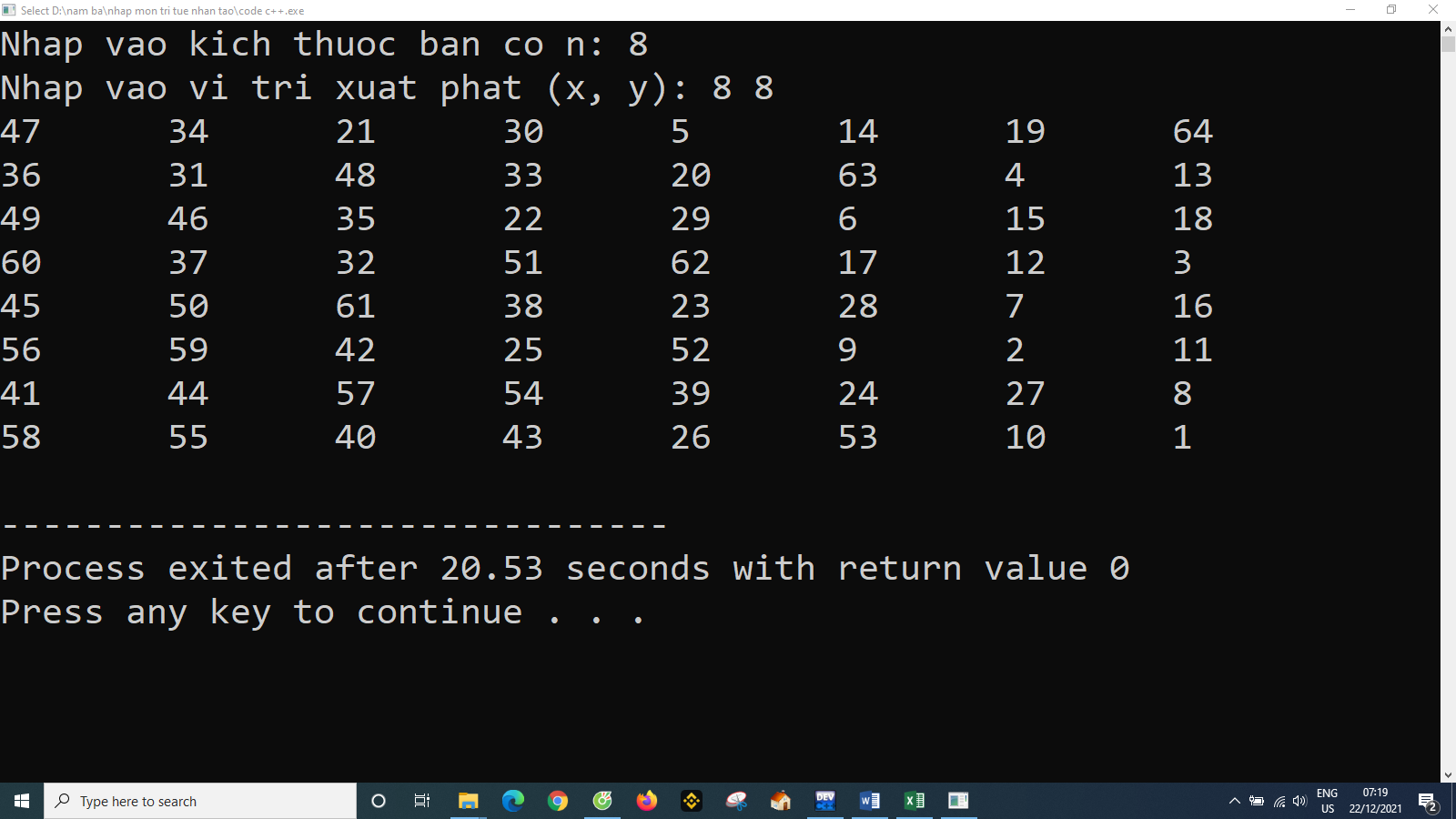
quaylui(2, x, y);

return 0;

}

### **3.1.2. Kết quả đạt được**

Với n = 8; x = 8; y = 8:



***Hình 6:*** *Kết quả đạt được với C++*

### **3.1.3 Kết luận**

**- Ưu điểm:** Cài đặt thuật toán nhanh , dễ hiểu dễ cài đặt và ngắn gọn.

**- Nhược điểm:** Chạy lâu do đệ quy nhiều lần, không có giao diện và khó hình dung.

# **KẾT LUẬN**

Với kiến thức hiện có của mình, chúng em đã hoàn thành các yêu cầu ở trên khi tiến hành thực hiện đề tài “**ÁP DỤNG THUẬT TOÁN QUAY LUI VÉT CẠN GIẢI QUYẾT BÀI TOÁN MÃ ĐI TUẦN**”. Tuy nhiên, trong quá trình làm sẽ không tránh khỏi những thiếu sót, hoặc cũng sẽ có những chỗ còn vướng mắc, chính vì vậy, chúng em mong được sự góp ý giúp đỡ của thầy giáo, để bài này được hoàn thiện hơn !

***Chúng em xin chân thành cảm ơn!***