

**ĐẠI HỌC BÁCH KHOA ĐÀ NẴNG  
KHOA CÔNG NGHỆ THÔNG TIN**

****

**ĐỒ ÁN**

**GIẢI THUẬT VÀ LẬP TRÌNH**

**ĐỀ TÀI**

**THUẬT TOÁN QUAY LUI**

**GVHD : PHAN THANH TAO**

**LỚP HP : 17Nh12A**

**SVTH :**

1. **PHẠM MẠNG DŨNG MSSV: 102170149**
2. **MAI VĂN THÀNH MSSV: 102170192**

**Đà Nẵng, 2019**

**Đa**

# **LỜI MỞ ĐẦU**

Ngày nay, Tin học ngày càng phát triển nhanh chóng và được ứng dụng rộng rãi trong mọi lĩnh vực đời sống xã hội, việc học và nắm bắt công nghệ mới đặc biệt là công nghệ thông tin ngày càng trở nên bức thiết. Đối với sinh viên trong ngành càng phải tích cực học tập, nắm vững mọi kiến thức về công nghệ thông tin, trong đó cấu trúc dữ liệu và giải thuật được xem là cơ sở, nền tảng đầu tiên.

Cấu trúc dữ liệu và Phân tích, thiết kế giải thuật là hai học phần rất quan trọng đối với người lập trình. Hai học phần này là cơ sở vũng chắc để giải quyết nhiều bài toán, đồng thời cung cấp cho chúng ra hiểu biết về tầm quan trọng của giải thuật, cũng như cách tổ chức dữ liệu để tối ưu hóa bài toán.

Sau một thời gian học tập và nghiên cứu ngôn ngữ lập trình C++ (C++ Programming Language), môn Cấu trúc dữ liệu, Phân tích và thiết kế giải thuật, để nắm bắt những kiến thức đã học tốt hơn, với ***Đồ Án Giải Thuật Và Lập Trình*** này nhóm chúng em chọn và thực hiện đề tài “**Phương Pháp Quay Lui**” (Backtracking)

Đây là đồ án đầu tiên mà chúng em được nhận, trong quá trình thực hiện đề tài, mặc dù đã ra sức nghiên cứu và cố gắng nhưng chắc chắn không tránh khỏi những thiếu sót. Chúng em rất mong nhận được những sự chỉ dẫn, ý kiến đóng góp của quý thầy cô để chúng em càng hoàn thiện hơn. Đồng thời chúng em xin gửi lời cảm ơn chân thành đến thầy Phan Thanh Tao đã giúp chúng em hoàn thành đề tài này.

Nhóm sinh viên thực hiện.

**MỤC LỤC**

[LỜI MỞ ĐẦU 2](#_Toc28804710)

[ĐỀ TÀI: THUẬT TOÁN QUAY LUI 4](#_Toc28804711)

[I. GIỚI THIỆU ĐỀ TÀI 4](#_Toc28804712)

[II. CƠ SỞ LÝ THUYẾT 4](#_Toc28804713)

[1. Ý tưởng 4](#_Toc28804714)

[2. Cơ sở lý thuyết 5](#_Toc28804715)

[3. Các bước thiết kế thuật toán 5](#_Toc28804716)

[III. BÀI TOÁN N HẬU HÒA BÌNH 5](#_Toc28804717)

[1. Phát biểu bài toán 5](#_Toc28804718)

[2. Các bước thiết kế thuật toán 5](#_Toc28804719)

[3. Thuật toán 5](#_Toc28804720)

[4. Độ phức tạp của thuật toán 6](#_Toc28804721)

[IV. BÀI TOÁN LIỆT KÊ CẤU HÌNH TỔ HỢP 6](#_Toc28804722)

[1. Phát biểu bài toán 6](#_Toc28804723)

[2. Các bước thiết kế thuật toán 7](#_Toc28804724)

[3. Thuật toán 7](#_Toc28804725)

[4. Độ phức tạp của thuật toán 7](#_Toc28804726)

[V. BÀI TOÁN MÃ ĐI TUẦN 8](#_Toc28804727)

[1. Phát biểu bài toán 8](#_Toc28804728)

[2. Các bước thiết kế thuật toán 8](#_Toc28804729)

[3. Thuật toán 8](#_Toc28804730)

[4. Độ phức tạp của thuật toán 9](#_Toc28804731)

[5. Cải tiến thuật toán với thuật toán Warnsdorff 9](#_Toc28804733)

[VI. CHƯƠNG TRÌNH 9](#_Toc28804734)

[1. Các thư viện dùng trong chương trình 9](#_Toc28804735)

[2. Các hàm sử dụng trong chương trình 9](#_Toc28804736)

[VII. KẾT QUẢ DEMO CHƯƠNG TRÌNH 11](#_Toc28804750)

[1. Giao diện chính 11](#_Toc28804751)

[2. Kết quả bài toán N Hậu hòa bình 12](#_Toc28804752)

[3. Kết quả bài toán Liệt kê cấu hình Tổ Hợp 12](#_Toc28804753)

[4. Kết quả bài toán Mã đi tuần 12](#_Toc28804754)

[5. Chương trình chi tiết 13](#_Toc28804755)

[VIII. }KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 17](#_Toc28804756)

[TÀI LIỆU THAM KHẢO: 17](#_Toc28804757)

# **ĐỀ TÀI: THUẬT TOÁN QUAY LUI**

1. GIỚI THIỆU ĐỀ TÀI

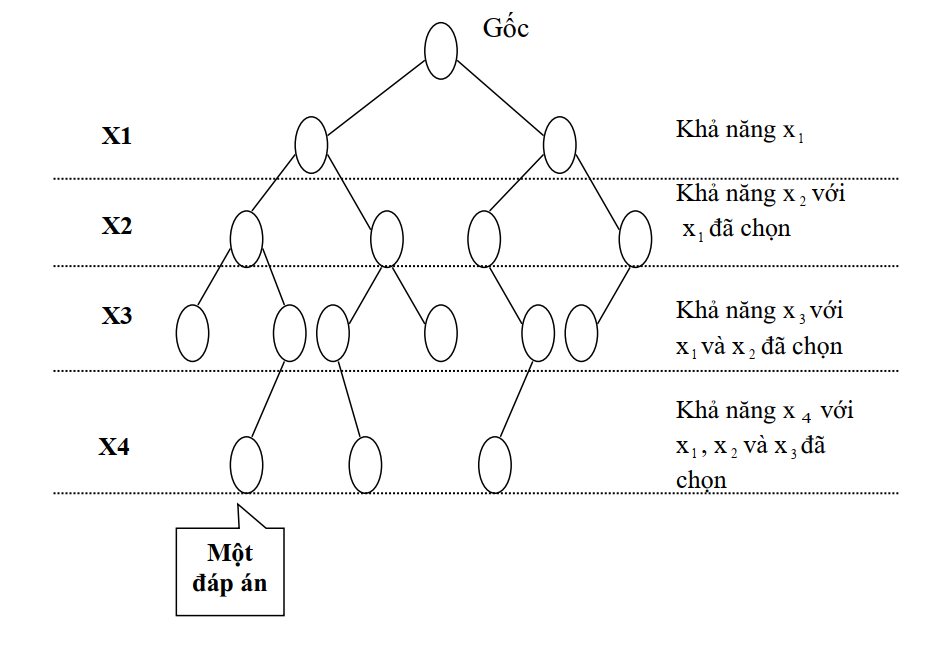
Có những bài toán yêu cầu phải liệt kê nghiệm theo một điều kiện cho trước nào đó. Ví dụ như một bài toán liệt kê tổ hợp cần được đảm bảo không được bỏ sót cũng như trùng lặp bất kỳ một trường hợp nào[1]. Phương pháp quay lui (backtracking) được sử dụng trong bài toán này nhằm thử tất cả các tổ hợp để tìm được một lời giải.

Đặc trưng của phương pháp quay lui là thử các trường hợp để tiến tới lời giải của bài toán. Việc thử thành công thì ghi nhận kết quả và tiến hành bước tiếp theo, còn ngược lại thì xóa bỏ các ghi nhận tại bước đó và quay về các lựa chọn còn lại. Thế mạnh của phương pháp này là nhiều cài đặt tránh được việc phải thử nhiều tổ hợp chưa hoàn chỉnh, và nhờ đó giảm thời gian chạy chương trình.[2]

Sau đây, trước hết chúng em xin trình bày mô hình chung của phương pháp quay lui. Sau đó trình bày một số bài toán điển hình sử dụng phương pháp quay lui. Cùng với đó là những chương trình cụ thể cho từng bài toán.

1. **CƠ SỞ LÝ THUYẾT**
2. **Ý tưởng**

Kĩ thuật quay lui (backtracking) như tên gọi của nó, là một quá trình phân tích đi xuống và quay lui trở lại theo con đường đã đi qua.Trong đó, việc tìm một lời giải đầy đủ được thực hiện bằng cách xây dựng các lời giải riêng phần sao cho chúng luôn thỏa mãn những điều kiện của bài toán. Giải thuật được áp dụng để kéo dài một lời giải riêng phần cho thành một lời giải trọn vẹn. Tuy nhiên, tại một bước nào đó, nếu có sự vi phạm điều kiện của bài toán, giải thuật sẽ quay ngược trở lại, bỏ đi sự lựa chọn mới nhất để thử với một khả năng cho phép khác [3]. Trong trường hợp không còn một hướng nào khác nữa thì thuật toán kết thúc. Thuật toán quay lui có thể được thể hiện theo sơ đồ cây tìm kiếm theo chiều sâu (Depth First Search – DFS) như hình dưới:



Từ hình vẽ, dẽ dàng nhận thấy:

- Ở một bài toán hiện tại (mỗi nốt), ta đi tìm lời giải cho bài toán đó. Ứng với lời giải, ta đi giải bài toán kế tiếp cho đến khi bài toán gốc trở nên đầy đủ.

- Lời giải của bài toán gốc thường là một lối đi từ gốc đến nốt cuối cùng (không có nốt con).

1. **Cơ sở lý thuyết**

Như đã đề cập phía trước, phương pháp quay lui về bản chất là quá trình tìm kiếm theo chiều sâu trên đồ thị không gian trạng thái của bài toán.

* Nút gốc tương ứng với trạng thái đầu (trước khi việc tìm kiếm giải pháp bắt đầu).
* Mỗi nút tương ứng với một giải pháp từng phần có triển vọng.
* Các nút lá tương ứng với giải pháp từng phần không có triển vọng hoặc giải pháp cuối cùng.[1]

Xét hoặt động của chương trình trên cây nhị phân tìm kiếm quay lui ta thấy tại mỗi bước thử chọn một nút xi, nút này sẽ gọi đệ quy để tìm tiếp xi+1 có nghĩa là quá trình sẽ duyệt sâu xuống phía dưới đến tận nút lá [4]. Cụ thể, nếu có một lựa chọn được chấp nhận thì ghi nhận 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. Các thông tin này thường được lưu trữ vào một ngăn xếp-Stack (vào sau ra trước) nhằm đảm bảo ghi nhớ tại mỗi bước đi qua để tránh trùng lặp khi quay lui.

Thuật toán quay lui thường được cài đặt theo lối đệ quy, trong đó mỗi thể hiện của hàm lấy thêm một biến và lần lượt gán tất cả các giá trị có thể cho biến đó [2], trong hàm sẽ cố gắng tìm lời giải cho bài toán con đó, nếu tìm thấy thì gọi hàm đệ quy để giải bài toán con tiếp theo hoặc là đưa ra đáp án bài toán lớn nếu đã đầy đủ lời giải, nếu không tìm thấy thì chương trình sẽ trở về điểm gọi hàm đó.

1. **Các bước thiết kế thuật toán**

* Chọn cách biểu diễn giải pháp.
* Xây dựng các tập A1, A2, …, An và xếp thứ tự các phần tử của chúng được xử lý.
* Xây dựng các điều kiện từ ràng buộc của bài toán để xác định một giải pháp từng phần là có triển vọng không, gọi là điều kiện tiếp tục.
* Chọn tiêu chí để xác định một giải pháp từng phần có là giải pháp cuối cùng không.[1]

1. **BÀI TOÁN N HẬU HÒA BÌNH**
2. **Phát biểu bài toán**

Tìm tất cả các khả năng xếp n con hậu trên một bàn cờ có n x n ô sao cho các con hậu không tấn công nhau, nghĩa là:

* Mỗi hàng chỉ chứa một con hậu
* Mỗi cột chỉ chứa một con hậu
* Mỗi đường chéo chỉ chứa một con hậu

Tiêu biểu với trường hợp tám quân hậu. Nếu không phân biệt các lời giải là ảnh của nhau qua phép đối xứng, phép quay bàn cờ thì chúng chỉ có 12 lời giải đơn vị [5]. Trong đồ án này, chúng em xin được xuất ra 1 kết quả trong nhiều kết quả có thể có của bài toán.

1. **Các bước thiết kế thuật toán**

* B1: Biểu diễn giải pháp: Giả sử con hậu k được đặt trên hàng k, như thế đối với mỗi con hậu chỉ cần mô tả cột chứa nó. Khi đó giải pháp được biểu diễn bởi véc-tơ S = (s1, s2, … , sn) với sk là cột mà con hậu k được đặt trên đó.
* B2: Xây dựng tập đầu vào A1, A2, … , An và thứ tự các phần tử: Ak = {1, … , n} với mọi k các phần tử sẽ được xử lý tăng dần.
* B3: Điều kiện tiếp tục: Giải pháp từng phần S = (s1, s2, …, sn) phải thỏa mãn ràng buộc bài toán (mỗi hàng, mỗi cột và mỗi đường chéo chỉ chưa đúng một con hậu).
* B4: Tiêu chí để xác định một giải pháp từng phần có là giải pháp cuối cùng: k = n. [1]

1. **Thuật toán**

**Algorithm:** N\_Hau

**Input:** Số n

**Output:** Tất cả các cách xếp hậu

Hàm đệ quy **OutputNHau**(k)

**Begin:**

**if** (k – 1 = n) **then** print (s[1…n]) // In cách xếp hậu

**else**

**for** i ← 1 **to** n **do**

s[k] ← i

**if** (TienVong(s[1..k])) t**hen**

OutputHau (k+1)

**endif**

**endfor**

**End**

Hàm **TrienVong** (s[1...k])

**Begin:**

**for** i ← 1 **to** k – 1 **do**

**if** (s[k] = s[i] **or** | i-k| = | s[i] – s[k] |) **then**

**return** false

**endif**

**endfor**

**return** true

**End [1]**

1. **Độ phức tạp của thuật toán**

Độ phức tạp về thời gian (Time Complexity)

Gọi T(n) là hàm thời gian của thuật toán, với n là dữ liệu đầu vào. Với bài toán, với mỗi lần gọi đệ quy thì một vị trí được chọn trên bàn cờ tương ứng hàng i, cột j và đánh dấu hàng i đã được chọn. Ta tiếp tục thực hiện việc chọn j cột trên n-i hàng còn lại.

Ta có T(n) = n\*T(n-1) + C1 với T(0) = C2; (C1, C2 là các hằng số)

Công thức được rút gọn về độ phức tạp về thời gian là T(n) = n\*(n-1)\*(n-2).. = O(n!)

1. **BÀI TOÁN LIỆT KÊ CẤU HÌNH TỔ HỢP**
2. **Phát biểu bài toán**

**Tổ hợp** chập k các phần tử của A (0 ≤ k ≤n) là một tập con k phần tử (0 ≤ k ≤n) của tập A.

Ví dụ: với k =3, n =4 ta có các tổ hợp sau: (1,2,3) , (1,2,4), (1,3,4), (1,2,4).

Trong toán học, công thức tính số tổ hợp là với (0 ≤k ≤n) [6]

Tuy nhiên, trong bài toán này chúng ta cần liệt kê tất cả các cấu hình tổ hợp.

1. **Các bước thiết kế thuật toán**

Để xây dựng được thuật toán quay lui ta cần biết:

* Dạng của cấu hình cần tìm
* Tất cả các khả năng của x[i]
* Điều kiện chấp nhận khả năng j

Áp dụng mô hình của thuật toán quay lui vào bài toán liệt kê các tổ hợp chập k của n phần tử. Lần lượt xét các yếu tố cơ bản của thuật toán quay lui trong bài toán sinh tổ hợp ta có:

* B1: Xác định dạng cấu hình cần tìm: đáp số có thể biểu diễn dưới dạng (x1, x2,…,xn) với x[i-1] < x[i]
* B2: Tất cả các khả năng của x[i]: Giá trị của x[i] là từ x[i-1] + 1 cho đến n-k+i (Cấu hình khởi tạo x[0] =0) .
* B3: Điều kiện “chấp nhận khả năng j” : Một tổ hợp chập k của n phải thỏa 0 ≤ k ≤n.

1. **Thuật toán**

**Algorithm**: TimToHop

**Input:** k, n

**Output:** Tất cả các cấu hình tổ hợp

Hàm đệ quy **BackTracking** (int i)

**Begin:**

**for** j ← a[i-1] + 1 **to** n-k+i **do** //Xet cac kha nang cua j

a[i] = j; //Ghi nhận một khả năng của j

**if** (i==k) OutputToHop ();

**else** BackTracking(i+1);

**end for**

**End**

Hàm **OutputToHop** ()

**Begin:**

**for** i ← 1 **to** k **do**

In ra a[i];

**endfor**

**End**

1. **Độ phức tạp của thuật toán**

Với vấn đề của bài toán đặt ra là tìm tổ hợp chập k của n phần tử, ta dễ dàng có công thức để tính số lượng tổ hợp được sinh ra như sau:

cong thuc to hop

Với k là một hằng số không đổi, k ≤ n – k

Tương tự với k > n – k, ta có độ phức tạp là O(nn-k).

Tổng quát ta có độ phức tạp về thời gian là O(nmin{k, n-k}), và với giới hạn trên của min{n, n-k} là n/2, thì có thể viết lại là O(nn/2).

1. **BÀI TOÁN MÃ ĐI TUẦN**
2. **Phát biểu bài toán**

Trên bàn cờ nxn ô, hãy tìm cách đi quân mã qua nxn ô sao cho mỗi ô chỉ đi qua một lần (xuất phát từ một ô bất kì nào đó). Tên bài toán này gọi là “Mã đi tuần” .

Trên bàn cờ 8x8 có 26.534.728.821.064 cách giải (tổng cả hành trình đóng – quân mã đi hết bàn cờ và quay lại vị trí xuất phát, và hành trình mở - quân mã đi hết bàn cờ). Và số cách giải là 9.862 với bàn cờ 6x6 [7]. Trong đồ án này, chúng em xin được xuất ra 1 kết quả bất kì của hành trình mở.

1. **Các bước thiết kế thuật toán**

* B1: Xây dựng nước đi cho quân mã
* Gọi x, y là độ dài bước đi trên các trục Oxy. Một bước đi hợp lệ của quân mã sẽ như sau: |x| + |y| = 3
* Khi ở một vị trí bất kì quân mã có 8 đường có thể di chuyển. Chưa xét đén tính hợp lệ, các bước đi đó là: (-2,-1), (-2,1), (-1,-2), (-1,2), (1,-2), (1,2), (2,-1), (2,1)
* Ta sẽ tạo mảng dd[] và dc [] để chứa các giá trị trên
* B2: Kiểm tra tính hợp lệ của bước đi
* Ta sẽ dùng mảng 2 chiều A[n\*n] để lưu vị trí của từng ô trong bàn cờ. Tất cả đều mang giá trị 0 (quân mã chưa đi qua)
* Gọi x, y là vị trí hiện tại của quân mã, vị trí tiếp theo sẽ có dạng x + X[i], y + Y[i]. Một vị trí hợp lệ khi 0 ≤ x + X[i] ≤ n -1 và 0 ≤ y + Y[i] ≤ n-1
* Nếu bước đi đó là đi đúng thì ta sẽ lưu thứ tự của bước đi đó vào mảng A[x+X[i], y+Y[i]]
* B3: Đường đi kết thúc khi quân mã đã đi đủ nxn bước [8].

1. **Thuật toán**

**Algorithm:** MaDT

**Input:** n, vị trí ban đầu của quân mã

**Output:** Đường đi của quân mã thỏa yêu cầu đề bài

Hàm đệ quy **NuocDi(x,y)** // đi từ ô (x,y)

**Begin:**

**Khởi gán** dd[8] = {-2,-1,1,2,2,1,-1,-2}

dc[8] = {1,2,2,1,-1,-2,-2,-1}

**Và** dem = 0 // số bước đi

++ dem

A[x][y] = dem // đánh dấu bước đi

**for** i ← 0 **to** 7 **do**

**if** (dem == n\*n) //Kiểm tra xem quân mã đã đi hết bàn cờ chưa

In lời giải

**endif**

u = x + dd[i], v = y + dc[i] // Nếu chưa tạo vị trí x, y mới

**if**(u>=0 && u<n && v>=0 && v<n && A[u][v] == 0) //Kiểm tra tính hợp lệ

Gọi NuocDi (u,v)

**endif**

**endfor**

-- dem; //Nếu không tìm được thì trả lại các giá trị ban đầu

A[x][y] = 0;

**End** [8]

1. **Độ phức tạp của thuật toán**

Độ phức tạp về thời gian (Time Complexity)

Gọi T(n) là hàm thời gian của thuật toán, với n là dữ liệu đầu vào. Với bài toán, tại mỗi bước trên bàn cờ, có 8 vị trí tiếp theo mà quân mã có thể đi. Và phải được thực hiện trên tất cả vị trí tên bàn cờ trừ vị trí cuối cùng (n-1)

Từ đó, ta có T(n) = 8\*T(n-1) + C1 với T(0) = C2; (C1, C2 là các hằng số)

Công thức được rút gọn về độ phức tạp về thời gian là T(n) = O(8n^2-1) = O(8n^2)

Nhận ra rằng độ phức tạp của thuật toán quay lui trong bài toán mã đi tuần là vô cùng lớn với dữ liệu có kích cỡ n lớn.Thuật toán Warnsdorff được kết hợp với phương pháp quay lui nhằm giải quyết những hạn chế trên.

1. **Cải tiến thuật toán với thuật toán Warnsdorff**
   1. Ý tưởng

Thay vì duyệt tất cả các hướng đi có thể trong bàn cờ từ vị trí đang xét, ta sẽ ưu tiên hướng đi đến vị trí tiếp theo mà tại đó, số hướng đi của nó là thấp nhất.

* 1. Thiết kế thuật toán

Với x, y là độ dài bước đi trên các trục Oxy, và A[x][y] là vị trí đang xét hiện tại. Ta thực hiện tìm kiếm bước đi tiếp theo tối ưu:

* Duyệt lần lượt tất cả các hướng đi hợp lệ.
* Tại mỗi hướng đi, lại tiến hành đếm số bước đi tiếp theo có thể có của nó.
* Trả lại hướng đi có số khả năng ít nhất.

1. **CHƯƠNG TRÌNH**
2. **Các thư viện dùng trong chương trình**

- Iostream: Định nghĩa các đối tượng để đọc và viết ra các dòng tiêu chuẩn (standard stream) Đây thường là tiêu đề duy nhất cần phải bao gồm để tiến hành xuất và nhập dữ liệu từ C++.

- Fstream: Định nghĩa các lớp hỗ trợ các phép toán cho dòng xuất nhập (iostream) trên các chuỗi chứa trong các tập tin bên ngoài.

- Iomanip: Định nghĩa nhiều phép điều chỉnh mà nó lấy vào một đối số. [9]

1. **Các hàm sử dụng trong chương trình**

### 2.1. Hàm N\_Hau:

- Sử dụng hàm fstream để đọc dữ liệu là số quân hậu trên bàn cờ từ file đầu vào InputNHau.txt.

- Truy xuất đến file đầu ra để các hàm được gọi phía sau giải quyết bài toán và ghi kết quả vào file đầu ra.

- Gọi hàm OutputNHau thực hiện bài toán.

### 2.2. Hàm OutputNHau(int k):

- In các vị trí xếp hậu thỏa mãn yêu cầu ra màn hình chương trình.

- Đồng thời sử dụng thư viện fstream để ghi và lưu trữ các vị trí xếp hậu vào file đầu tra OutputNHau.txt.

- Khi ( k – 1) = n (với k nhận giá trị truyền vào từ hàm N\_Hau là 1) thì thực hiện in ra kết quả.

- Ngược lại, với i nhận giá trị từ 1 đến n, s[k] nhận giá trị bằng i, gọi hàm Trienvong(k) để kiểm tra điều kiện. Nếu triển vọng trả về giá trị 1 thì gọi đệ quy lại hàm OutputNHau và tiếp tục thực hiện.

### 2.3. Hàm Trienvong(int k):

- Xây dựng hàm Trienvong(int k) để kiểm tra xem vị trí đặt quân hậu tiếp theo có thỏa mãn yêu cầu đề bài không.

- Với i chạy từ 1 đến nhỏ hơn k, nếu s[k] = s[i] hoặc trị tuyệt đối của(i-k) bằng trị tuyệt đối của (s[i] - s[k])) nghĩa là vị trí đặt quân hậu vi phạm yêu cầu (quân hậu vừa đặt có thể tấn công 1 trong những quân hậu trước đó) hàm sẽ trả về 0.

- Ngược lại trả về 1.

### 2.4. Hàm TimToHop:

- Sử dụng hàm fstream để đọc dữ liệu là giá trị của k và n từ file đầu vào InputToHop.txt.

- Truy xuất đến file đầu ra để các hàm được gọi phía sau giải quyết bài toán và ghi kết quả vào file đầu ra.

- Gọi hàm ToHop thực hiện bài toán.

### 2.5. Hàm ToHop

- Xây dựng hàm ToHop để kiểm tra tính hợp lệ của dữ liệu đầu vào.

- Nếu dữ liệu đầu vào thỏa 0 ≤ k ≤ n thì khởi tạo mảng s có giá trị đầu tiên s[0] bằng 0 và gọi hàm BackTracking tiếp tục thực hiện.

- Ngược lại, dừng bài toán do dữ liệu đầu vào sai.

### 2.6. Hàm BackTracking(int i):

- Xây dựng hàm BackTracking(int i) để tìm ra các cấu hình tổ hợp thỏa yêu cầu đề bài đưa ra.

- Xét các giá trị của j từ s[i-1] + 1 đến n – k + i, với i nhận giá trị ban đầu là 1 từ hàm ToHop truyền vào.

- Mỗi một phần tử s[i] của mảng s sẽ lưu một giá trị của j.

- Nếu i = k, nghĩa là cấu hình đã đủ phần từ thì gọi hàm OutputToHop để thực hiện việc in ra kết quả.

- Ngược lại nếu i chưa bằng k thì gọi đệ quy lại hàm BackTracking(i+1) để tiếp tục thực hiện.

### 2.7. Hàm OutputToHop

- In các cấu hình tổ hợp ra màn hình chương trình.

- Đồng thời sử dụng thư viện fstream để ghi và lưu trữ các cấu hình tổ hợp vào file đầu tra OutputToHop.txt.

### 2.8. Hàm MaDT

- Sử dụng hàm fstream để đọc dữ liệu là kích thước bàn cờ và vị trí ban đầu của quân mã từ file đầu vào InputMaDT.txt

- Truy xuất đến file đầu ra để các hàm được gọi phía sau giải quyết bài toán và ghi kết quả vào file đầu ra.

- Gọi hàm NuocDi thực hiện bài toán.

### 2.9.Hàm NuocDi(int x, int y, int &flag):

- Xây dựng hàm NuocDi(int x, int y, int &flag) để tìm ra nước đi của quân mã thỏa yêu cầu đề bài đưa ra.

- Sử dụng biến d để tăng hoặc giảm bước đi của quân mã đồng thời dùng hàm A[x][y] = d để đánh dấu bước đi.

- Nếu d = n\*n, tức là quân mã đã đi hết bàn cờ thì dừng lại, gọi hàm OutputMaDT để in ra kết quả đồng thời gán giá trị cho cờ flag = 1.

- Ngược lại nếu quân mã chưa đi hết bàn cờ thì gọi hàm KiemTraUuTien để chọn hướng đi tiếp theo tối ưu nhất, tạo nước đi mới và gọi đệ quy lại chính hàm NuocDi.

- Kết thúc hàm NuocDi nếu cờ flag có giá trị bằng 0 tức là không tìm được kết quả cho bài toán thì hàm MaDT sẽ hiển thị ra thông báo ‘Khong tim thay duong di”.

### 2.10.Hàm OutputMaDT :

- In các bước đi của quân mã ra màn hình chương trình.

- Đồng thời sử dụng thư viện fstream để ghi và lưu trữ các bước đi này vào file đầu tra OutputMaDT.txt.

### 2.11.Hàm DemKhaNang (int a[][max], int x, int y)

- Trả về số nước đi có thể có tiếp theo từ vị trí quân mã đang đứng hiện tại.

### 2.12.Hàm KiemTraUuTien (int a[][max], int x, int y)

- Tạo nước đi mới theo một hướng có thể có bất kì, kiểm tra điều kiện nước đi có hợp lệ không.

- Nếu nước đi hợp lệ, gọi hàm DemKhaNang để đếm số nước đi có thể có nếu xuất phát theo hướng đi đó.

- Trả về hướng đi có số bước đi tiếp theo ít nhất.

### 2.13.Hàm main

- Hàm main() của chương trình chúng em trình bày theo giao diện bảng chọn với lựa chọn:

1. Bài toán N Hậu hòa bình

2. Bài toán Liệt kê cấu hình Tổ Hợp

3. Bài toán Mã đi tuần

0. Thoát

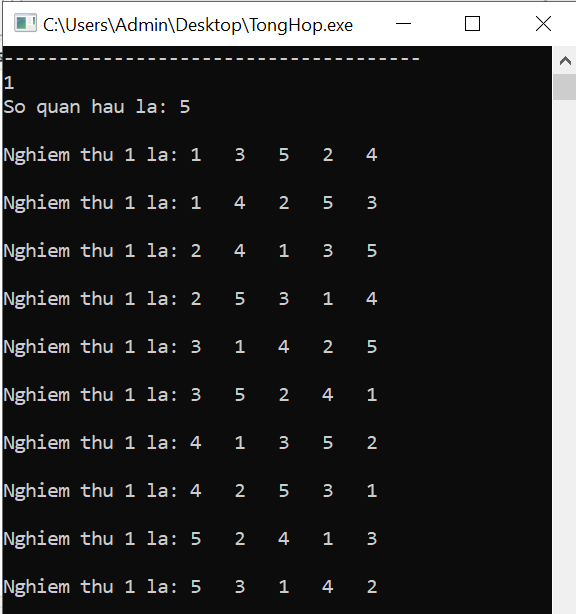
1. **KẾT QUẢ DEMO CHƯƠNG TRÌNH**

## Giao diện chính

Ảnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động

## 2. Kết quả bài toán N Hậu hòa bình



## `3. Kết quả bài toán Liệt kê cấu hình Tổ Hợp

Ảnh có chứa ảnh chụp màn hình

Mô tả được tạo tự động

## 4. Kết quả bài toán Mã đi tuần

Ảnh có chứa thiết bị điện tử

Mô tả được tạo tự động

## 5. Chương trình chi tiết

#include <fstream>

#include <iostream>

#include <iomanip>

#define nmax 1000

#define abs(x) ((x)<0 ? (-(x)):(x))

using namespace std;

int choice, n, m, d = 0, k, a, b;

int s[nmax];

int A[nmax][nmax] = { 0 }; //Khoi tao mang gia tri 0

int dd[] = { -2,-2,-1,-1, 1, 1, 2, 2 };

int dc[] = { -1, 1,-2, 2,-2, 2,-1, 1 };

ifstream filein;

ofstream fileout;

int N\_Hau();

void OutputNHau(int k);

int Trienvong(int k);

int TimToHop();

void ToHop();

void BackTracking(int i);

void OutputToHop();

int MaDT();

void NuocDi(int x, int y, int& flag);

void OutputMaDT();

int DemKhaNang(int a[][nmax], int x, int y);

int KiemTraUuTien(int a[][nmax], int i, int j);

main(){

cout << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "| DO AN GIAI THUAT & LAP TRINH |" << endl;

cout << "| DE TAI: THUAT TOAN QUAY LUI |" << endl;

cout << "| Giao Vien Huong Dan: Phan Thanh Tao |" << endl;

cout << "| Sinh Vien Thuc Hien: |" << endl;

cout << "| 1. Pham Manh Dung Lop: 17T3 MSSV: 102170149 |" << endl;

cout << "| 2. Mai Van Thanh Lop: 17T3 MSSV: 102170192 |" << endl;

cout << "| Lop HP: 17.Nh12A |" << endl;

cout << "|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|" << endl;

do{

cout << "\n\n Chao ban! Moi lua chon" << endl;

cout << "--------------------------------------" << endl;

cout << "1.Bai toan N Hau hoa binh" << endl;

cout << "2.Bai toan To Hop" << endl;

cout << "3.Bai toan Ma Di Tuan" << endl;

cout << "0.Ket thuc" << endl;

cout << "--------------------------------------" << endl;

cin >> choice;

switch (choice)

{

case 1: N\_Hau();

break;

case 2: TimToHop();

break;

case 3: MaDT();

continue;

}

} while (choice != 0);

}

int N\_Hau(){

// Mo va doc du lieu tu file InputNHau.txt

filein.open("InputNHau.txt");

filein >> n;

cout << "So quan hau la: " << n << endl;

filein.close(); //Dong file da mo

// Mo file ghi ket qua OutputNHau.txt

fileout.open("OutputNHau.txt");

OutputNHau(1); // Xep Hau

fileout.close(); //dong file da mo

return 0;

}

void OutputNHau(int k){

int i;

if (k - 1 == n)

{

d++;

fileout << endl << "Nghiem thu " << d << " la: ";

cout << endl << "Nghiem thu " << d << " la: ";

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

{

fileout << s[i] << " ";

cout << s[i] << " ";

}

fileout << endl;

cout << endl;

}

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

{

s[k] = i;

if (Trienvong(k)) OutputNHau(k + 1);

}

d = 0;

}

int Trienvong(int k){

int i;

for (i = 1; i < k; i++)

if (s[k] == s[i] || abs(i - k) == abs(s[i] - s[k])) return 0;

return 1;

}

int TimToHop(){

// Mo va doc du lieu tu file InputToHop.txt

filein.open("InputToHop.txt");

filein >> k >> n;

cout << "Gia tri cua k va n: " << k << " " << n << endl;

filein.close(); //Dong file da mo

// Mo file ghi ket qua OutputToHop.txt

fileout.open("OutputToHop.txt");

ToHop(); //Tim to hop

fileout.close(); //dong file da mo

return 0;

}

void ToHop(){ //Liet ke cac to hop

if (k >= 0 && k <= n)

{

s[0] = 0; //khoi tao gia tri a[0]

cout << "Cac to hop la: \n";

BackTracking(1);

}

else

{ cout << "Loi: Khong thoa dieu kien 0<=k<=n" << endl;}

}

void BackTracking(int i){

for (int j = s[i - 1] + 1; j <= n - k + i; j++) // xet cac kha nang cua j

{

s[i] = j; // ghi nhan mot gia tri cua j

if (i == k) // neu cau hinh da du k phan tu

{

// in mot cau hinh ra ngoai

OutputToHop();

}

else

{

BackTracking(i + 1);} // Quay Lui

}

}

void OutputToHop(){ // ham de in 1 cau hinh

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

{

cout << s[i] << "";

fileout << s[i] << "";

}

cout << endl;

fileout << endl;

}

int MaDT(){

// Mo va doc du lieu tu file InputMaDT.txt

filein.open("InputMaDT.txt");

filein >> n >> a >> b;

filein.close(); //Dong file da mo

// Mo file ghi ket qua OutputMaDT.txt

fileout.open("OutputMaDT.txt");

int flag = 0;

NuocDi(a, b, flag); // Ham nuoc di cua quan ma

if (!flag)

{

//Neu khong tim duoc duong di thi se thong bao

cout << "Khong tim thay duong di";

fileout << "Khong tim thay duong di";

}

fileout.close(); //Dong file da mo

}

void NuocDi(int x, int y, int& flag){

++d; //Tang gia tri buoc di

A[x][y] = d; //Danh dau da di

//Kiem tra xem quan ma da di het ban co chua

if (d == n \* n)

{

cout << "Cac buoc di la: \n";

fileout << "Cac buoc di la: \n";

OutputMaDT();

flag = 1;

}

if (KiemTraUuTien(A, x, y) != -1 && flag != 1) {

//Neu chua di het ban co thi tao nuoc di moi

//Tao vi tri x, y moi

int pos = KiemTraUuTien(A, x, y);

int u = x + dd[pos], v = y + dc[pos];

NuocDi(u, v, flag);

}

//Neu khong tim duoc buoc di thi tra lai cac gia tri ban dau

--d;

A[x][y] = 0;

}

void OutputMaDT(){

for (int i = 0; i < n; i++)

{

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

{

cout << setw(4) << A[i][j];

fileout << setw(4) << A[i][j];

}

cout << endl;

fileout << "\n";

}

cout << endl;

fileout << "\n";

}

int DemKhaNang(int a[][nmax], int x, int y){

int count = 0;

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

{

int u = x + dd[i], v = y + dc[i];

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

count++;

}

return count;

}

int KiemTraUuTien(int a[][nmax], int x, int y){

int min = 9, b, min\_i = -1;

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

{

if (a[x][y] == 63)

{

int u = x + dd[i], v = y + dc[i];

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

return i;

}

else

{

int u = x + dd[i], v = y + dc[i];

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

{

b = DemKhaNang(a, u, v);

if (b < min && b != 0)

{

min = b;

min\_i = i;

}

}

}

}

return min\_i;

1. }**KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

"N hậu hòa bình", "Liệt kê cấu hình tổ hợp" hay "Mã đi tuần" là những ví dụ điển hình về ứng dung của thuật toán quay lui.

Thuật toán quay lui cho phép ta tìm được kết quả của bài toán chỉ bằng những đoạn code ngắn, nhưng bên cạnh đó độ phức tạp của đệ quy là khá lớn. Với những cấu trúc có kích cỡ lớn thì việc sử dụng thuật toán quay lui và đệ quy tỏ ra khá chậm và chiếm dung lượng lớn bộ nhớ. Việc kết hợp thuật toán quay lui với những thuật toán khác để giải quyết nhanh những bài toán phức tạp là một bước tiến trong kỹ thuật lập trình.

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 sớm (cực đại hóa ảnh hưởng của lựa chọn sớm hiện hành)[2]. Trong đồ án này, chúng em đã sử dụng thuật toán Warnsdorff như một phương pháp heuristic của bài toán Mã đi tuần.

Qua quá trình nghiên cứu, thực hiện để hoàn thiện đề tài này, nhóm chúng em thấy được tầm quan trọng của cũng như những ưu điểm của thuật toán quay lui mang lại. Chúng em sẽ cố gắng để ngày càng hoàn thiện khả năng tư duy cũng như kỹ năng lập trình để ngày càng tiến bộ hơn trên lĩnh vực Công nghệ thông tin.

Một lần nữa, chúng em xin cảm ơn quý thầy cô đã giúp chúng em hoàn thiện đề tài này.

# TÀI LIỆU THAM KHẢO:

[1] Phân tích và thiết kế giải thuật, Nguyễn Thanh Bình – Phạm Minh Tuấn – Đặng Thiên Bình, NXB Giáo Dục Việt Nam.

[2] Quay lui. Lấy từ liên kết: [https://vi.wikipedia.org/wiki/Quay\_lui\_(khoa\_học\_máy\_tính)](https://vi.wikipedia.org/wiki/Quay_lui_(khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh))

[3] Cấu trúc dữ liệu và giải thuật, ĐH Bách Khoa Hồ Chí Minh.

[4] Giải thuật và lập trình, Lê Minh Hoàng, Đại học Sư phạm Hà Nôi, 1999-2002.

[5] Bài toán tám hậu. Lấy từ liên kết: [https://vi.wikipedia.org/wiki/Bài\_toán\_tám\_quân\_hậu](https://vi.wikipedia.org/wiki/B%C3%A0i_to%C3%A1n_t%C3%A1m_qu%C3%A2n_h%E1%BA%ADu)

[6] Tổ hợp. Lấy từ liên kết: [https://vi.wikipedia.org/wiki/Tổ\_hợp\_(toán\_học)](https://vi.wikipedia.org/wiki/T%E1%BB%95_h%E1%BB%A3p_(to%C3%A1n_h%E1%BB%8Dc))

[7] Knight’s tour. Lấy từ liên kết: <https://en.wikipedia.org/wiki/Knight%27s_tour>

[8] Kỹ thuật lập trình, Trần Đan Như – Nguyễn Thanh Phương – Đinh Bá Tiến – Trần Minh Triết – Đặng Bình Phương, NXB Khoa Học và Kỹ Thuật.

[9] Thư viện chuẩn C++. Lấy từ liên kết: [https://vi.wikipedia.org/wiki/Thư\_viện\_chuẩn\_C%2B%2B](https://vi.wikipedia.org/wiki/Th%C6%B0_vi%E1%BB%87n_chu%E1%BA%A9n_C%2B%2B)