# BÀI BÁO CÁO THUẬT TOÁN DSA

*Nguyễn Đình Trung Kiên - 24120195*

Ngày 23 tháng 3 năm 2025

## 1 Fibonacci

### 1.1 Mô tả:

Từ một số nguyên dương n cho trước, in ra màn hình các phần tử của dãy Fibonacci từ 0 đến n – 1.

### 1.2 Hướng giải với đệ quy:

- Khi n <= 1 thì phần tử của dãy Fibonacci tương ứng là F(0) = 0 và F(1) = 1
- Khi n > 1 thì phần tử của dãy Fibonacci được tính với công thức F(n) = F(n – 1) + F(n – 2). Ta thấy sự gọi lại chính bản thân hàm Fibonacci nên ta sử dụng đệ quy để giải quyết bài toán trên.
- Trường hợp cơ sở - điều kiện dừng của đệ quy: khi n = 0 hoặc n = 1 thì trả về F(n) = n.
- Bước đệ quy: gọi lại chính hàm Fibonacci để tính 2 phần tử liền trước của dãy: F(n) = F(n – 1) + F(n – 2) với n > 1.
- Sử dụng vòng lặp từ 0 đến n – 1 để in ra các số Fibonacci từ 0 đến n – 1.

### 1.3 Độ phức tạp thuật toán:

- Ta thấy ở mỗi lần gọi đệ quy thì đều gọi thêm lại 2 lần hàm fibonacci, gọi liên tục từ n về 1 hoặc 0. Vì vậy cây đệ quy của thuật toán trên sẽ có 2n nhánh.
- Độ phức tạp thời gian của thuật toán: $O(2^n)$.

## 2 Factorial (giai thừa)

### 2.1 Mô tả:

Từ số nguyên dương n cho trước, tính ra và in ra màn hình giai thừa n! của n với công thức n! = n(n – 1)(n – 2)... 1.

### 2.2 Hướng giải với đệ quy:

- Để tính được giai thừa của một số nguyên dương n, ta cần nhân liên tục với sự giảm dần của số n về 1: n! = (n – 2)(n – 1)... 1. Để lặp được như trên cùng trường hợp cơ sở để dừng lại là khi n đã giảm về 1, ta sử dụng đệ quy trong hàm.
- Ta thấy trường hợp cơ sở của thuật toán gọi đệ quy: khi n = 0 hay n = 1 thì 1! = 0! = 1.
- Bước đệ quy: Ta thấy từ công thức n! = n(n – 1)(n – 2)... 1 tương đương với công thức n! = n nhân với giai thừa của n – 1. Vì vậy, ta xây dựng được công thức truy hồi: n! = n * (n – 1)!.

### 2.3 Độ phức tạp thuật toán:

- Ta thấy hàm sẽ duyệt qua một vòng lặp giảm từ n về 1 để tạo thành giai thừa của n, tổng cộng duyệt đúng n lần cho mọi số nguyên dương n. Vì vậy, ta có độ phức tạp của thuật toán là O(n).

## 3 Tạo các chuỗi nhị phân có chiều dài n.

### 3.1 Mô tả:

In ra màn hình tất cả các chuỗi chị phân (chỉ bao gồm số 0 và 1) có độ dài n.

### 3.2 Hướng giải với đệ quy:

- Ta thấy cách để tạo ra một chuỗi đệ quy có độ dài n là lần lượt thêm '1' và '0' vào cuối chuỗi có độ dài n – 1. Ví dụ:
  - Các chuỗi có độ dài 2 là: 01, 10, 11, 00. Tổng cộng 4 chuỗi.
  - Ta thêm '1' vào các chuỗi trên: 101, 110, 111, 100. Sau đó thêm '0': 001, 010, 011, 000. Tổng cộng 8 chuỗi.
- Từ cách nhìn trên, ta sử dụng đệ quy để tạo ra một chuỗi nhị phân có n ký tự từ một chuỗi rỗng được truyền vào hàm.
- Trường hợp cơ sở: khi n giảm từ n về 0 nghĩa là tất cả các ký tự của chuỗi đã được làm điền bởi '0' và '1', vì vậy ta in chuỗi ra màn hình và kết thúc đệ quy.
- Bước đệ quy: Từ chuỗi ban đầu là chuỗi rỗng, ta truyền vào chính hàm đó một chuỗi khác được tạo ra từ chuỗi đó rồi thêm vào '0' và '1', mỗi trường hợp ta gọi hàm đó 1 lần.

### 3.3 Độ phức tạp thuật toán:

- Khi giảm từ n về 0 để điền hết n ký tự bởi '0' và '1', ở mỗi bước ta gọi lại hàm 2 lần cho cả 2 trường hợp thêm '0' và '1' vào cuối chuỗi.
- Vì vậy, sau khi điền hết n ký tự cho tất cả các chuỗi, ta gọi tổng cộng 2n lần cho hàm, tương đương có tổng cộng 2n chuỗi được tạo ra.
- Từ đó ta suy ra độ phức tạp của thuật toán là O(2n).

# 4 Bài toán tháp Hanoi.

## 4.1 Mô tả:

Cho trước 3 tháp để chuyển đổi các đĩa, n đĩa được đặt vào 1 trong 3 tháp, nhiệm vụ là cần chuyển n đĩa đó qua 1 tháp khác và tháp còn lại được sử dụng làm trung gian. Yêu cầu: không được chuyển hơn 1 đĩa trong 1 lần chuyển và không được đặt đĩa to hơn lên đĩa nhỏ hơn. In ra màn hình tất cả các bước chuyển.

## 4.2 Hướng giải với đệ quy:

- Ta thấy để chuyển được n đĩa với yêu cầu trên, ta cần chuyển n – 1 đĩa ở trên cùng của tháp qua tháp trung gian và chuyển đĩa to nhất (nằm dưới cùng) qua tháp cần chuyển đến. Chẳng hạn ta có 3 đĩa cần chuyển, ta không thể chuyển đĩa trên cùng qua tháp cần chuyển trước, vì như vậy sẽ không thể để đĩa to nhất qua được. Vì vậy ta cần chuyển các đĩa nhỏ đi để nhường chỗ cho đĩa to nhất qua được đến tháp cần chuyển.
- Từ ý tưởng trên, ta lập thành phương pháp với các bước:
  - Chuyển n – 1 đĩa trên cùng của tháp đầu tiên qua tháp trung gian.
  - Chuyển đĩa to nhất qua tháp cần chuyển.
  - Tiếp tục chuyển n – 1 đĩa từ tháp trung gian qua lại tháp cần chuyển.
- Ta thấy trong chính các bước để chuyển đĩa, ta lại sử dụng cách chuyển đĩa đó cho n – 1 đĩa trên cùng. Vì vậy, đệ quy là thuật toán tối ưu được sử dụng.
- Trường hợp cơ sở: Khi chỉ còn duy nhất 1 đĩa cần chuyển, ta có thể chuyển trực tiếp đĩa đó qua tháp đích đến.
- Bước đệ quy: Khi còn nhiều hơn 1 đĩa cần chuyển, ta cần chuyển n – 1 đĩa trên cùng qua tháp trung gian, rồi chuyển đĩa to nhất rồi lại chuyển n – 1 đĩa đó qua tháp đích đến. Với mỗi bước chuyển n – 1 đĩa, ta chuyển dần dần như các bước trêm đến khi chỉ còn 1 đĩa to nhất để chuyển trực tiếp, đó cũng là trường hợp cơ sở để dừng đệ quy.

## 4.3 Độ phức tạp thuật toán:

- Ở mỗi lần gọi đệ quy để chuyển n đĩa, ta gọi hàm 2 lần để chuyển n – 1 đĩa và gọi 1 câu lệnh cho bước chuyển đĩa to nhất. Vì vậy, để chuyển hết n đĩa ta có số lần $T(n) = 2T(n – 1) + 1$ với $T(1) = 1$.
- Giải hệ thức đệ quy trên ta được $T(n) = 2n - 1$. Từ đó ta có độ phức tạp của thuật toán là $O(2n)$.

# 5 Kiểm tra mảng tăng dần với đệ quy.

## 5.1 Mô tả:

Sử dụng đệ quy để kiểm tra một mảng được cho trước có được sắp xếp theo thứ tự hay không.

### 5.2 Hướng giải với đệ quy:

- Ta cần kiểm tra tất cả các cặp phần tử liền nhau trong mảng xem có đúng thứ tự hay không. Để tạo vòng lặp kiểm tra, ta sử dụng đệ quy gọi lại chính hàm đó đến khi đã kiểm tra hết mảng.
- Trường hợp cơ sở: Khi kiểm tra đến phần tử cuối cùng hoặc kiểm tra hết các cặp trong mảng, hoặc mảng chỉ có dưới 2 phần tử thì luôn luôn đã được sắp xếp theo thứ tự.
- Bước đệ quy: Thu hẹp dần mảng từ n xuống 1 hoặc 0 về phía đầu mảng theo từng bước, ở mỗi bước kiểm tra 2 phần từ cuối mảng.

### 5.3 Độ phức tạp thuật toán:

- Ta cần duyệt qua tất cả các cặp phần tử liền kề nhau trong mảng bằng cách thu hẹp mảng lại theo từng bước, mỗi bước một phần tử. Vì vậy số lần gọi hàm để duyệt là n – 1 lần.
- Từ đó suy ra độ phức tạp của thuật toán trên là O(n).

## 6 Bài tháp N-Queens.

### 6.1 Mô tả:

Tính số cách sắp xếp n quân hậu vào bàn cờ n x n ô theo luật. In ra màn hình số cách sắp xếp đó.

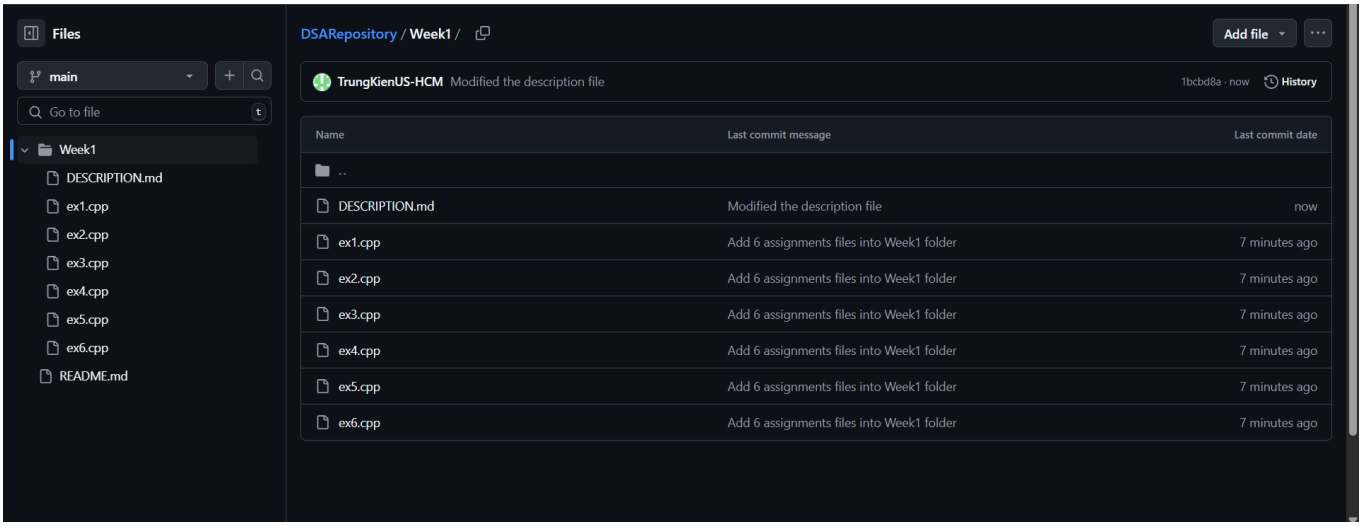
### 6.2 Hướng giải với đệ quy:

- Để xác định xem 1 ô trên bàn cờ là có khả thi để đặt 1 quân hậu hay không, ta cần kiểm tra các ô liền kề. Vì vậy cần viết 1 hàm để kiểm tra các ô trên cùng cột và các ô trên cùng đường xéo chính, phụ (nhưng chỉ cần kiểm tra từ ô đó trở lên vì đang kiểm tra lần lượt từ trên xuống để đặt quân hậu, vì vậy chắc chắn các ô dưới chưa được đặt).
- Sử dụng thuật toán backtrack (quay lui) để giải bài toán trên:
  - Lần lượt kiểm tra các phần tử trên hàng cần kiểm tra.
  - Ở mỗi vị trí hợp lệ để đặt quân hậu tiếp theo, ta đệ quy đến hàng kế tiếp để tìm ra cách đặt toàn bộ n quân hậu vào n hàng. Đến khi đặt được quân hậu thứ n ở hàng n – 1, một cách đặt thành công được ghi nhận lại.
  - Lặp lại các bước trên cho đến khi đã duyệt hết tất cả các phần tử ở hàng đầu tiên (tất cả các hướng đặt quân hậu đều đã được duyệt qua).
  - Ở mỗi bước đệ quy để tìm hướng đi, nếu đã duyệt qua cả hàng mà không tìm được hướng đi hợp lệ, thuật toán sẽ quay lại hàng trước đó và tìm hướng đi khác, ở ô tiếp theo.

### 6.3 Độ phức tạp thuật toán:

- Ở hàng đầu tiên, ta xét qua N vị trí của hàng để đặt quân hậu.
- Đệ quy đến hàng tiếp theo, ta còn lại N – 1 vị trí cần xét để đặt quân hậu (loại trừ vị trí ở cùng cột với quân hậu đã được đặt ở hàng đầu tiên).
- Tiếp tục đến khi duyệt qua hết N hàng, ta có số cần duyệt: T(N) = N * (N – 1) * (N – 2)... * 1 = N!.
- Từ đó suy ra độ phức tạp thuật toán trên là O(N!).

## 7 Hình ảnh minh chứng việc up file bài tập lên Github:




Hình 1: Minh chứng