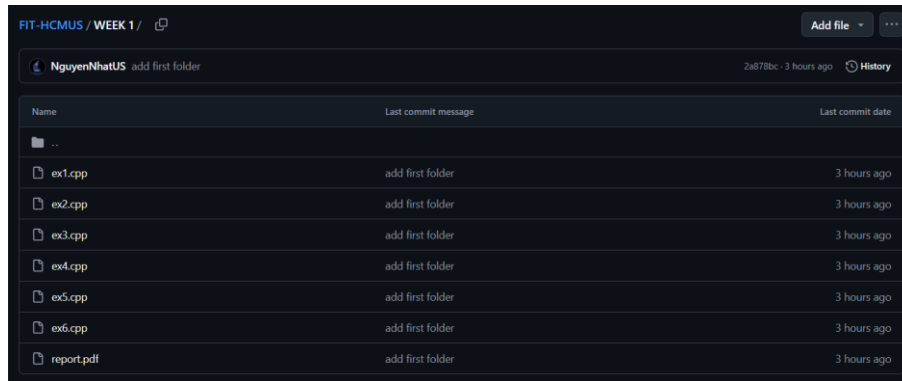


Nguyễn Lê Đức Nhật

24120403



The screenshot shows a file explorer interface with a commit history table. The table has three columns: Name, Last commit message, and Last commit date. The data is as follows:

Name	Last commit message	Last commit date
..		
ex1.cpp	add first folder	3 hours ago
ex2.cpp	add first folder	3 hours ago
ex3.cpp	add first folder	3 hours ago
ex4.cpp	add first folder	3 hours ago
ex5.cpp	add first folder	3 hours ago
ex6.cpp	add first folder	3 hours ago
report.pdf	add first folder	3 hours ago

Bài 1: Fibonacci

Base case: • Với $n = 0$ hoặc $n = 1$ thì return n . Vì $\text{fibonacci}[0] = 0, \text{fibonacci}[1] = 1$;

Recursive case: Nếu $n > 1$ thì gọi đệ quy $\text{fibonacci}[n - 1] + \text{fibonacci}[n - 2]$

// ($\text{fibonacci}[n] = \text{fibonacci}[n - 1] + \text{fibonacci}[n - 2]$)

Bài 2: Factorial

• **Base case:** Nếu $(n = 0 \parallel n = 1)$ thì return 1 vì $0! = 1! = 1$

• **Recursive case:** Nếu $n > 1$ thì gọi hàm đệ quy $n * \text{factorial}(n - 1)$ // vì $n! = n * (n - 1)!$

Bài 3: Generate Binary String

• **Base case:** Nếu $\text{str.size()} == n$ (tức chuỗi str đã có chiều dài n) thì ta in ra kết quả và return để hàm backtrack đến bước trước đó.

+ Char $c[2] = \{'0', '1'\}$ // vì đây là xâu nhị phân nên mỗi ký tự chỉ có thể 2 giá trị là 0 và 1.

Lặp 1 vòng for từ 0 đến 1, thêm kí tự `c[i]` vào xâu `str` sau đó gọi đệ quy (`n, str`)

Khi `str.size() == n`, tức là gặp base case thì hàm sẽ in ra xâu `str` có độ dài `n` sau đó backtrack để tiếp tục xử lí, `str.pop_back()` chính là bước backtrack

Bài 4: Tower of Hanoi

- **Base Case** : Khi chỉ có 1 đĩa (`n == 1`), ta di chuyển trực tiếp từ cột `from_rod` đến `to_rod`.
- **Recursive Case**: Nếu có `n` đĩa, ta chia bài toán thành 3 bước:
 1. **Di chuyển n-1 đĩa** từ `from_rod` sang `aux_rod` (cột trung gian) bằng cách sử dụng `to_rod` làm cột phụ.
 2. **Di chuyển đĩa lớn nhất (n)** trực tiếp từ `from_rod` sang `to_rod`.
 3. **Di chuyển n-1 đĩa** từ `aux_rod` sang `to_rod` bằng cách sử dụng `from_rod` làm cột trung gian.

Bài 5: is Sorted array

•**Base case**: `if(n == 1 || n == 0)` return true vì mảng có 1 hoặc 0 phần tử thì luôn được sắp xếp.

// check: nếu phần tử đầu tiên của mảng lớn hơn phần tử thứ hai thì return false

•**Recursive case**: gọi đệ quy (`arr + 1, n - 1`), lúc này tăng chỉ số của mảng lên 1 đơn vị và giảm số phần tử của mảng đi 1 đơn vị. (Xem như loại bỏ phần tử đầu tiên của mảng) và gọi đệ quy đến mảng từ phần tử thứ hai đến cuối.

Bài 6: N-Queens

•**Base case**: `if(i == N + 1)` lúc này đã sắp xếp được 8 con hậu lên bàn cờ => cộng 1 cách xếp

Với mỗi hàng `i`, kiểm tra xem đâu là cột thỏa mãn có thể đặt quân hậu đó, cột `j` thỏa mãn khi trên cột `j` chưa có bất kì con hậu nào khác, tương tự với đường chéo xuôi (`i - j + N`) và đường chéo ngược (`i + j - 1`). Nếu `j` thỏa thì ta đánh dấu tương ứng với cột `j`, chéo xuôi và chéo ngược sau đó gọi đệ quy đến hàng tiếp theo. Sau đó là bước backtrack để hủy bỏ đánh dấu cột `j` vừa thăm.

