

Phân Tích Tư Duy Thuật Toán

Codeforces 460A - Vasya and Socks

Coach Tư Duy Thuật Toán

Ngày 13 tháng 1 năm 2026

Lộ Trình Tư Duy

- 1 Bước 1: Tiếp Nhận & Phẫu Thuật (Briefing)
- 2 Bước 2: Vòng Lặp Tư Duy - Chunk 1 (Cơ chế)
- 3 Bước 2: Vòng Lặp Tư Duy - Chunk 2 (Mô phỏng)
- 4 Bước 2: Vòng Lặp Tư Duy - Chunk 3 (Thứ tự)
- 5 Bước 3: Tổng kết Code Mô phỏng
- 6 Tư duy Toán học $O(1)$
- 7 Giải thích sâu (Deep Dive)

Tóm tắt đề bài (Ngôn ngữ con người)

Hãy liệt kê lớp vỏ bọc "Vasya" và "đôi tất" để nhìn thấy lỗi toán học.

- **Vốn khởi nghiệp:** Bạn có tài nguyên ban đầu là n .
- **Luật tiêu dùng:** Mỗi ngày trôi qua, tiêu hao **1** đơn vị.
- **Luật hồi phục (Bonus):** Cứ sau mỗi m ngày, bạn được thưởng thêm **1** đơn vị vào kho.
- **Mục tiêu:** Tìm xem sau bao nhiêu ngày thì "phá sản" (tài nguyên về 0).

Lộ trình tư duy

- 1 **Cơ chế dòng chảy:** Hiểu cách tài nguyên giảm và tăng.
- 2 **Mô phỏng (Simulation):** Chạy thử quy trình.
- 3 **Toán học $O(1)$:** Tìm công thức tối ưu.

Chunk 1: Logic & Ấn dụ hóa

Logic (Ấn dụ hóa)

Hãy tưởng tượng n là mức pin điện thoại.

- Mỗi ngày pin tụt 1 vạch.
- Cứ đến ngày thứ $m, 2m, 3m \dots$ bạn cắm sạc dự phòng kích thêm được 1 vạch pin.

Vấn đề: Cái "vạch pin" được sạc thêm đó, nó lại giúp điện thoại sống thêm được 1 ngày nữa. Và biết đâu trong cái ngày sống thêm đó, nó lại chạm mốc để được sạc tiếp?

Chunk 1: Bẫy & Thử thách tư duy

Bẫy (Trap)

Sai lầm phổ biến là dùng phép chia đơn thuần (Tổng chia m). **Lý do:** Bạn quên mất rằng cái đôi tất được tặng thêm cũng tính vào quy trình để nhận đôi tất tiếp theo (Lãi suất kép).

Thử thách tư duy (Mental Check)

Input: $n = 4$ (4 đôi gốc), $m = 2$ (2 ngày thưởng 1).

- Ngày 1: Dùng 1. Còn 3.
- Ngày 2: Dùng 1. Còn 2. **Sự kiện:** Qua 2 ngày \rightarrow Thưởng +1. Thực tế còn $2 + 1 = 3$.

Chunk 1: Kết quả thử thách

Câu trả lời của bạn: $4 + 2 + 1 = 7$ ngày.

Phân tích trực giác

Phép tính này cho thấy bạn đã nắm được bản chất:

- 4 đôi gốc giúp sống qua ngày 2 và 4 (được thưởng 2 đôi).
- 2 đôi thưởng đó giúp sống qua ngày 6 (được thưởng thêm 1 đôi nữa).
- 1 đôi cuối cùng giúp sống nốt ngày 7.

Chunk 2: Chiến thuật mô phỏng (Simulation)

Chúng ta dạy máy tính cách "Sống qua từng ngày".

Logic (Cơ chế vận hành)

Hai biến số:

- 1 **Đồng hồ (Days):** Bắt đầu từ 0. Tăng dần.
- 2 **Kho tất (Socks):** Bắt đầu là n . Giảm dần.

Quy trình mỗi sáng:

- Kho tất giảm 1.
- Đồng hồ tăng 1.
- **KIỂM TRA THƯỞNG:** Nhìn vào Đồng hồ.

Chunk 2: Bẫy Logic & Modulo

Bẫy (Trap)

Rất nhiều bạn kiểm tra thưởng dựa trên số tất còn lại \rightarrow SAI. **Đúng:** Phải kiểm tra dựa trên **số ngày đã trôi qua**.

Kiểm tra tư duy

Để biết ngày nào được thưởng ($m = 3 \rightarrow$ ngày 3, 6, 9...), ta dùng phép toán gì?

\rightarrow **Phép Chia lấy dư (Modulo %).**

Nếu $\text{số_ngày \% } m == 0$ thì Thưởng!

Chunk 3: Thứ tự sinh tồn

Thứ tự hành động quyết định sống hay chết!

Kịch bản trong ngày

- 1 Sáng ngủ dậy, rút 1 đôi tất (n giảm 1).
- 2 Ngày trôi qua ($days$ tăng 1).
- 3 Chiều tối, nếu là ngày thưởng ($days \% m == 0$) \rightarrow Nhận thêm 1 đôi (n tăng 1).

Điều kiện lặp: Tiếp tục miễn là $n > 0$.

Chunk 3: Tình huống "Ngàn cân treo sợi tóc"

Thử thách

Input: $n = 2, m = 2$.

- Ngày 1: Dừng 1 \rightarrow Còn 1.
- Ngày 2: Dừng 1 \rightarrow Còn 0 (Kho sạch bách!).

Câu hỏi: Tại thời điểm này, Game Over hay Sống tiếp?

Chunk 3: Tình huống "Ngàn cân treo sợi tóc"

Thử thách

Input: $n = 2, m = 2$.

- Ngày 1: Dừng 1 \rightarrow Còn 1.
- Ngày 2: Dừng 1 \rightarrow Còn 0 (Kho sạch bách!).

Câu hỏi: Tại thời điểm này, Game Over hay Sống tiếp?

Giải đáp

Đáp án: Sống tiếp!

Vì là ngày 2 (chia hết cho 2), vào buổi tối bạn được thưởng 1 đôi. Kho từ 0 lên 1.
 \rightarrow *Thứ tự thực hiện lệnh (Order of Execution)* đã cứu bạn.

Bản thiết kế (Pseudocode)

Tổng hợp logic thành mã giả:

```
B T      U   :  
  N h p  n, m  
    t    b i n      m      n g y (days) = 0  
  
TRONG KHI (n > 0):  <--  C h n g  n o c n  t t  t h  c n  
                      s n g  
    1.  G i m  n  i  1      (S n g  n g  d y  d n g)  
    2.  T n g  days l n 1  (T h i  gian tr i qua)  
  
    3.  K I M  TRA  T H      NG :  
        N U  (days % m == 0):  
            T n g  n  l n 1  (C h i u  t i  n h n  qu )  
  
K T  T H  C  V  NG  L P :  
In ra  s      days
```

Nâng cấp tư duy: Từ Mô phỏng sang $O(1)$

Bạn đã nhận ra quy luật cấp số nhân ($n + n/m + \dots$). Hãy chuyển sang tư duy "**Chi phí thực**".

Phân tích Lỗ - Lãi

Trong một chu kỳ m ngày:

- Bạn mất: m đôi.
- Bạn nhận lại: 1 đôi.

→ **Chi phí thực**: Để sống qua m ngày (nhận 1 lần thưởng), kho tất thực sự chỉ hụt đi:

$$m - 1 \quad (\text{đôi})$$

Công thức "Thần Thánh"

Bẫy "Đôi tất cuối cùng"

Công thức $\frac{n}{m-1}$ bị sai ở biên vì bạn không thể tiêu đôi tất cuối cùng để chờ thưởng được.

Giải pháp: Cắt đi 1 đôi an toàn, chỉ tính thưởng trên $(n-1)$ đôi.

$$\text{Số tất thưởng} = \left\lfloor \frac{n-1}{m-1} \right\rfloor$$

Tổng số ngày = Vốn gốc + Số tất thưởng:

$$\text{Kết quả} = n + \left\lfloor \frac{n-1}{m-1} \right\rfloor$$

Code $O(1)$ - C++ & Python

C++ Solution:

```
#include <iostream>
using namespace std;

int main() {
    int n, m;
    cin >> n >> m;
    // Công thức  $O(1)$ 
    cout << n + (n - 1) / (m - 1);
    return 0;
}
```

Python Solution:

```
n, m = map(int, input().split())

# Phep chia lay nguyen //
print(n + (n - 1) // (m - 1))
```

Tại sao công thức khớp với chuỗi cộng dồn?

Câu hỏi: Tại sao chuỗi $50 + 25 + 12 + \dots$ lại bằng $\frac{n-1}{m-1}$?

Truy tìm "Vụn bánh mì"

Phép chia số nguyên bỏ qua phần dư.

- $100/2 = 50$
- $25/2 = 12$ (Dư 1 \rightarrow Cất đi)
- $3/2 = 1$ (Dư 1 \rightarrow Cất đi)

Các phần dư này cộng gộp lại sẽ đủ để đổi thêm lần thưởng nữa. Công thức toán học tính luôn cả các phần dư "vô hình" này.

Chứng minh toán học (Cấp số nhân)

Tổng số tất cả là tổng của chuỗi vô hạn:

$$S = \frac{n}{m} + \frac{n}{m^2} + \frac{n}{m^3} + \dots$$

Áp dụng công thức tổng cấp số nhân lùi vô hạn $S = \frac{a}{1-r}$:

- Số hạng đầu $a = n/m$
- Công bội $r = 1/m$

$$S = \frac{\frac{n}{m}}{1 - \frac{1}{m}} = \frac{\frac{n}{m}}{\frac{m-1}{m}} = \frac{n}{m-1}$$

Do phải giữ lại 1 đôi cuối cùng (không thể chia nhỏ), ta dùng $n-1$.

Chúc mừng!

Bạn đã đi từ tư duy **Mô phỏng (Simulation)** đến tư duy **Đại số (Algebra)**.

- Simulation: Dễ hiểu, dễ cài đặt.
- Math $O(1)$: Hiệu suất tối đa, nhìn thấu bản chất.