Chinh phục bài toán 1472B - Fair Division

Tiếp cận theo hướng Quy hoạch động (DP)

Coach "Slide Learning CPP"

Ngày 19 tháng 1 năm 2026

Giới thiệu

Lời chào

Chào bạn! Rất vui được đồng hành cùng bạn chinh phục bài toán **1472B - Fair Division** trên Codeforces.

- Phương pháp: Quy hoạch động (Dynamic Programming - DP).
- Bản chất DP: "Xây dựng kết quả lớn từ những kết quả nhỏ đã biết".
- Lộ trình: Chia nhỏ vấn đề thành các Micro-chunks logic.

Bước 1: Phẫu thuật đề bài (Deconstruct)

Cốt lõi bài toán

- Bạn có n viên kẹo.
- Cân nặng mỗi viên: **1 unit** hoặc **2 units**.
- **Mục tiêu:** Chia toàn bộ kẹo thành 2 phần có tổng cân nặng bằng nhau.

Lộ trình tư duy

1. Xác định tổng trọng lượng cần đạt được cho mỗi người.
2. Dùng DP để kiểm tra khả năng tạo ra phần quà đó.

Bước 2: Mảnh ghép 1 - Mục tiêu của phép chia

Giả sử tổng cân nặng của tất cả viên kẹo là S. Để chia đều, mỗi người phải nhận chính xác $S/2$.

Bẫy logic (Trap)

Nếu S là một số lẻ (ví dụ $S = 5$), liệu chúng ta có bao giờ chia đều được không?

Thử thách tư duy

Nếu tôi có: 1 viên kẹo nặng 2 unit và 1 viên kẹo nặng 1 unit $\rightarrow S = 3$. Theo bạn, chúng ta có cần dùng DP để kiểm tra trường hợp này không, hay có thể kết luận ngay là "NO"? Tại sao?

Mảnh ghép 2 - Cập nhật trạng thái DP

Khi ta nhặt thêm một viên kẹo nặng $w = 2$, các trạng thái mới của mảng dp sẽ được cập nhật.

Cảnh báo: Bẫy Knapsack

Nếu duyệt từ đầu đến cuối (từ 0 đến S), bạn có thể vô tình dùng một viên kẹo nhiều lần!

Thử thách tư duy

Để tránh việc một viên kẹo vừa làm `dp[j]` thành `true` lại tiếp tục được dùng để cập nhật `dp[j+w]`, chúng ta nên:

- **A.** Duyệt từ S ngược về 0.
- **B.** Duyệt từ 0 tiến lên S.

Mảnh ghép 2 - Cập nhật trạng thái DP

Khi ta nhặt thêm một viên kẹo nặng $w = 2$, các trạng thái mới của mảng dp sẽ được cập nhật.

Cảnh báo: Bẫy Knapsack

Nếu duyệt từ đầu đến cuối (từ 0 đến S), bạn có thể vô tình dùng một viên kẹo nhiều lần!

Thử thách tư duy

Để tránh việc một viên kẹo vừa làm `dp[j]` thành `true` lại tiếp tục được dùng để cập nhật `dp[j+w]`, chúng ta nên:

- **A.** Duyệt từ S ngược về 0.
- **B.** Duyệt từ 0 tiến lên S.

Đáp án: A. Duyệt ngược là chìa khóa để mỗi vật phẩm chỉ được dùng đúng một lần.

Bước 3: Tổng kết Thuật toán (Mã giả)

```
Tong = tong tat ca vien  k o
Neu Tong % 2 != 0:
    In ra "NO"
Target = Tong / 2
dp[0] = true, tat ca dp[1...Target] = false

Voi moi vien  k o  nang 'w':
   Duyet j tu Target nguoc ve w:
      Neu dp[j-w] == true:
         dp[j] = true

Neu dp[Target] == true:
    In ra "YES"
Con lai:
    In ra "NO"

```

Listing 1: Thuật toán DP cho Fair Division

Mảnh ghép cuối: Tối ưu hóa logic

Kiểm tra trường hợp đặc biệt

Giả sử có 3 viên kẹo nặng 2 ($n = 3$, $S = 6$, $Target = 3$).

- `dp[0] = true`
- Viên 1 ($w = 2$): `dp[2] = true`
- Viên 2 ($w = 2$): `dp[4]` (vượt Target 3 nên bỏ qua)
- Kết quả: `dp[3]` vẫn là `false`.

Kết luận

Thuật toán DP trả về **NO** là chính xác, vì không thể tạo ra mức 3 từ các viên kẹo nặng 2. DP kiểm tra được cả "nguyên liệu"có phù hợp hay không.

Bước 4: Chốt hạ kiến thức

1. **Loại trừ:** Tổng lẻ luôn là "NO".
2. **Xây dựng:** Dùng mảng dp kiểm tra khả năng tạo mức *Target*.
3. **Duyệt ngược:** Đảm bảo tính chất của bài toán Knapsack 0/1.

Bạn muốn làm gì tiếp theo?

1. **Thử thách Code:** Tự viết mã C++ hoặc Python.
2. **Tối ưu hóa:** Khám phá cách giải bằng Toán học (Greedy).
3. **Sang bài mới:** Luyện tập bài DP khác trên Codeforces.