LỜI GIẢI ĐỀ LUYỆN 03

(Dành cho đội tuyển quốc gia)

Bài 1. Các đoạn con đẹp nhất

Phân tích bài toán

Trước hết, hãy xem *một mảng đẹp* B có dạng như thế nào. Với mọi cặp phần tử trong B, một phần tử phải chia hết cho phần tử còn lại. Vì thứ tự của các phần tử không ảnh hưởng đến tính chất này, ta có thể giả sử:

$$B_1 \leq B_2 \leq B_3 \leq \ldots \leq B_k.$$

Khi đó, điều kiện "mỗi cặp chia hết" tương đương với:

$$\forall\, 1 \leq i < j \leq k:\ B_i \mid B_j.$$

Đặc biệt, B_i phải chia hết cho B_{i+1} với mọi i.

Ngược lại, nếu dãy B thỏa mãn rằng mỗi phần tử chia hết cho phần tử kế tiếp, thì rõ ràng nó cũng là một mảng đẹp. Do đó, ta có thể hiểu rằng **một mảng đẹp là một dãy (khi sắp tăng dần) trong đó mỗi phần tử chia hết cho phần tử kế tiếp**. Hay nói cách khác, đây là một *chuỗi chia hết* (divisibility chain).

Chiến lược tối ưu

Quay lại bài toán ban đầu, ta cần tìm **điểm số lớn nhất có thể** của một đoạn đẹp của A, sau khi sắp xếp lại các phần tử. Vì được phép hoán vị A, vấn đề chỉ còn là chọn ra tập phần tử sẽ nằm trong đoạn con B, rồi sắp xếp chúng sao cho mỗi phần tử chia hết cho phần tử kế tiếp.

Điểm số được định nghĩa là:

$$\text{score}(B) = \min(B) \cdot |B|.$$

Nếu $\min(B)$ đã cố định, ta chỉ cần tối đa hoá $|B|$, tức là tìm chuỗi chia hết dài nhất bắt đầu từ phần tử nhỏ nhất đó.

Quy hoạch động

Đặt $\text{freq}[x]$ là số lần xuất hiện của x trong mảng A. Nếu ta đã chọn phần tử x, thì tốt nhất nên chọn **tất cả** các bản sao của nó.

Gọi:

$$\text{dp}[x] = \text{độ dài chuỗi chia hết dài nhất mà phần tử nhỏ nhất là } x.$$

Khi đó, ta có công thức:

$$\text{dp}[x] = \text{freq}[x] + \max_{i>1} \text{dp}[i \cdot x],$$

với điều kiện $i \cdot x \leq M$, trong đó $M = \max(A)$.

Giá trị kết quả là:

$$\boxed{\max_{1 \le x \le M} (x \cdot \text{dp}[x])}.$$

Độ phức tạp

Vì mỗi x chỉ duyệt qua các bội của nó, nên tổng số phép duyệt là:

$$M + \frac{M}{2} + \frac{M}{3} + \cdots + 1 = \mathcal{O}(M \log M).$$

Thêm vào đó, việc đếm tần suất có độ phức tạp $\mathcal{O}(N)$. Do đó, tổng thời gian cho mỗi test là:

$$\boxed{\mathcal{O}(N + M \log M)},$$

trong đó $M = \max(A)$. Vì tổng M trên tất cả các test không vượt quá 2×10^5, thuật toán này hoàn toàn đáp ứng giới hạn thời gian.

Bài 2. Công tắc và bóng đèn

Phân tích bài toán

Gọi chỉ số i ($1 \le i \le N$) là **tốt** nếu $A_i = B_i$, và là **xấu** nếu ngược lại. Mục tiêu của ta là làm cho tất cả các chỉ số trở thành "tốt" sau đúng hai lần lật.

Nhận xét rằng các chỉ số "xấu" sẽ tạo thành một số nhóm rời nhau, mỗi nhóm là một đoạn liên tiếp. Ví dụ:

$$A = \texttt{1001101}, \quad B = \texttt{0010111}$$

Khi đó, các chỉ số xấu là $\{1, 3, 4, 6\}$, tạo thành các đoạn $[1], [3, 4], [6]$.

Quan sát quan trọng

Một lần lật đoạn có thể giảm số lượng các đoạn xấu đi **nhiều nhất là một**.

Chứng minh. Giả sử một phép lật cắt qua $K \ge 2$ đoạn xấu. Khi đó, nó cũng phải bao gồm tất cả các đoạn tốt nằm giữa chúng. Có $K - 1$ đoạn tốt như vậy, và chúng sẽ trở thành "xấu" sau khi lật. Như vậy, số đoạn xấu giảm đi nhiều nhất là $K - (K - 1) = 1$.

Do đó, nếu số đoạn xấu > 2, thì ta **không thể** biến A thành B trong đúng hai lần lật, và đáp án là 0. Vì vậy, ta chỉ cần xét các trường hợp có nhiều nhất hai đoạn xấu. Ký hiệu X là số đoạn xấu.

Trường hợp 1: $X = 0$

Khi đó, ban đầu $A = B$. Sau lần lật đầu tiên, ta sẽ tạo ra đúng một đoạn xấu — chính là đoạn được lật. Lần lật thứ hai phải là lật lại chính đoạn đó để khôi phục.

Vì vậy, sau khi chọn đoạn đầu tiên, đoạn thứ hai là duy nhất. Số lựa chọn cho lần lật đầu tiên chính là số đoạn con của mảng độ dài N:

$$\frac{N \cdot (N + 1)}{2}.$$

Trường hợp 2: $X = 1$

Giả sử đoạn xấu duy nhất là $[L, R]$. Lần lật thứ hai cũng là duy nhất — nó phải biến đoạn xấu còn lại thành tốt. Ta cần đếm số cách chọn lần lật đầu tiên để vẫn còn đúng một đoạn xấu sau khi thực hiện.

Các khả năng:

- Cắt bớt đoạn xấu bằng cách bỏ đi một phần đầu hoặc phần cuối: có $(R - L)$ cách chọn tiền tố $[L, i]$ và $(R - L)$ cách chọn hậu tố $[i, R]$.
- Mở rộng đoạn $[L, R]$ sang trái: chọn $[i, L - 1]$, có $(L - 1)$ cách.

- Hoặc chọn $[i, R]$ để xóa hoàn toàn đoạn xấu ban đầu — thêm $(L-1)$ cách nữa.
- Mở rộng sang phải: chọn $[R+1, i]$, có $(N-R)$ cách.
- Hoặc chọn $[L, i]$ để xóa đoạn xấu ban đầu — thêm $(N-R)$ cách.

Tổng cộng:

$$2 \cdot (R-L) + 2 \cdot (L-1) + 2 \cdot (N-R) = 2 \cdot (N-1),$$

và kết quả này không phụ thuộc vào L, R.

Trường hợp 3: $X = 2$

Giả sử hai đoạn xấu là $S_1 = [L_1, R_1]$ và $S_2 = [L_2, R_2]$ với $R_1 < L_2$. Một khi lần lật đầu tiên đã chọn, lần lật thứ hai cũng cố định. Ta chỉ cần đếm số lựa chọn của lần lật đầu tiên sao cho còn đúng một đoạn xấu.

Các khả năng hợp lệ:

- Xóa hoàn toàn S_1, rồi xóa hoàn toàn S_2, hoặc ngược lại.
- Chọn $[L_1, L_2 - 1]$ rồi $[R_1 + 1, R_2]$, hoặc ngược lại.
- Chọn $[L_1, R_2]$ rồi $[R_1 + 1, L_2 - 1]$, hoặc ngược lại.

Tổng cộng có đúng 6 cách.

Kết luận

Khi đã biết X, đáp án xác định ngay trong $\mathcal{O}(1)$. Ta chỉ cần đếm X — số đoạn xấu — trong $\mathcal{O}(N)$, ví dụ duyệt qua chuỗi và đếm số chỉ số i sao cho $A_i \neq B_i$ nhưng $A_{i-1} = B_{i-1}$.

Độ phức tạp thời gian:

$\boxed{\mathcal{O}(N) \text{ cho mỗi test.}}$

Bài 3. Dãy con

Phân tích bài toán

Các phần tử trong C chỉ có thể đạt được các giá trị $\geq$ một phần tử nào đó của A, vì ta chỉ có thể nối thêm bản sao của A và tăng giá trị một số phần tử. Do đó, nếu tồn tại phần tử trong B nhỏ hơn tất cả phần tử của A, thì phần tử đó sẽ không bao giờ xuất hiện trong C. Nói cách khác, nếu

$$\min(B) < \min(A),$$

thì kết quả chắc chắn là -1. Trong các trường hợp khác, luôn tồn tại lời giải — ta chỉ cần tìm số thao tác tối thiểu.

Ý tưởng xây dựng dãy B trong C

Giả sử ta đang cố gắng tạo ra dãy B theo thứ tự trong C. Gọi ptr là con trỏ chỉ vị trí hiện tại trong C. Với mỗi phần tử B_i (duyệt lần lượt $i = 1, 2, \ldots, M$), ta cần làm cho B_i xuất hiện tại một vị trí $>$ ptr trong C với số thao tác ít nhất có thể.

Có hai khả năng:

1. Tìm một vị trí $j >$ ptr sao cho $C_j \leq B_i$, sau đó tăng C_j lên tới B_i.
2. Hoặc, nối thêm một bản sao mới của A vào C (mất một thao tác), rồi quay lại bước đầu tiên.

Quyết định chiến lược: Nối thêm hay không

Câu hỏi đặt ra: khi nào thì nên nối thêm bản sao mới của A? Điều này được xác định bởi một lựa chọn tham lam (greedy):

Mệnh đề. Khi chọn phần tử của A để biến thành B_i, luôn tối ưu khi chọn phần tử A_j gần nhất với B_i mà không vượt quá B_i. Tức là:

$$A_j = \max\{A_t \mid A_t \leq B_i\}.$$

Chứng minh. Xét một phần tử $A_k < A_j$ và giả sử ta dùng A_k thay vì A_j.

- Nếu không cần nối thêm bản sao mới của A, chi phí là $(B_i - A_k) \geq (B_i - A_j) + 1$, nên chọn A_j luôn tốt hơn.
- Nếu cần nối thêm bản sao mới, ta có hai tình huống:
  - Nếu cả A_k và A_j đều chỉ có thể dùng trong bản sao mới, rõ ràng A_j vẫn tốt hơn.
  - Nếu A_k có thể dùng sớm hơn, thì việc chọn A_j trong bản sao mới không tốn kém hơn — thậm chí còn "thuận lợi" hơn cho các phần tử sau.

Vì vậy, không bao giờ tệ hơn khi chọn A_j thay cho A_k; do đó ta luôn có thể chọn phần tử lớn nhất $\leq B_i$ trong A.

Cách hiện thực

Với mỗi B_i, ta thực hiện:

1. Tìm x là phần tử trong A lớn nhất sao cho $x \leq B_i$ (dùng **binary search** trên mảng A).
2. Nếu tồn tại một vị trí của x nằm sau ptr, di chuyển con trỏ tới vị trí đó.
3. Nếu không tồn tại, nối thêm một bản sao của A (tăng thêm một thao tác) rồi di chuyển tới lần xuất hiện đầu tiên của x.

Để tìm nhanh lần xuất hiện tiếp theo của x, ta lưu danh sách các chỉ số mà $A_j = x$ và dùng tìm kiếm nhị phân trên danh sách đó. Lưu ý rằng ptr thực chất là vị trí trong C, nhưng do C được tạo bằng cách nối nhiều bản sao của A, ta chỉ cần quản lý ptr theo *chỉ số trong một bản sao của* A (tức là theo modulo N).

Độ phức tạp

$\mathcal{O}(N \log N + M \log N)$ cho mỗi test.