LGCD SOLUTION

Đề: Cho dãy số nguyên dương A ($A_i \leq 10^{18}$) có độ dài là N ($N \leq 10^5$) và một số k ($1 \leq k \leq 10^{18}$). Hãy tìm dãy con liên tiếp dài nhất có ước chung lớn nhất bằng k.

Solution

Nếu dãy con liên tiếp ta cần tìm có ước chung lớn nhất là k, thì có phải rằng mọi số trong dãy con liên tiếp này đều chia hết cho k không? Đúng vậy, vì thế chúng ta chỉ cần dùng 1 vòng lặp đơn giản để tìm đoạn dài nhất chia hết cho k mà thôi.

Thế bài toán chỉ có tới đây thôi à? Bạn nghĩ làm gì có chuyện ngon ăn như thế? Bởi vì thuật toán trên **sai bét.** Vấn đề nằm ở chỗ là: Nếu một dãy có ước chung lớn nhất là k thì đương nhiên mọi phần tử trong dãy này đều chia hết cho k, nhưng nếu mọi phần tử trong dãy này chia hết cho k thì **chưa chắc** dãy này có ước chung lớn nhất là k. Bạn vẫn chưa hiểu tại sao ư? Hãy cùng xem một ví dụ nhỏ như sau:

$$A = \{18, 36, 54\}$$

$$k = 9$$

Ta có thể thấy rõ ràng rằng ước chung lớn nhất của 3 số trên bằng 18 mặc dù số nào cũng chia hết cho 9?!?!?! Và thế là từ đó điểm của bạn bay màu, kéo theo cả những huy chương vàng và giải quốc gia của bạn xuống nấm mồ… giỡn thôi. Vì chúng ta sẽ đi vào cách giải bài này theo thuật toán đúng đắn ngay ở dưới đây.

Subtask 1

Cái này thì dễ rồi, bạn có thể làm một thuật trâu $O(N^3)$ đơn giản: 1 vòng lặp để xét hết tất cả các độ dài, 2 vòng lặp nữa để tính ước chung lớn nhất của mọi đoạn con và so sánh chúng. Bạn có thể code đơn giản như sau:

```
// Vòng for đầu tiên để thử từng độ dài một
for(int d = n; d > 0; --d){
	// 2 vòng for này dùng để xét hết gcd của các đoạn có độ dài = d
	for(int i = 0; i + d - 1 < n; ++i){
		long long res = arr[i];
		for(int j = i; j <= i + d - 1; ++j) res = __gcd(res, arr[j]);
		if(res == k) return d;
	}
}
return 0; // Nếu không độ dài nào thỏa trả về 0
```

Subtask 2

Trước khi đi vào lời giải, có 2 điều cần phải được chứng minh (bạn sẽ biết vì sao nó cần sau đó):

1. Gọi ước chung lớn nhất của dãy a bất kỳ là x. Nếu ta tăng số lượng phần tử của dãy a thì giá trị của x sẽ không bao giờ tăng.

2. Với mọi số $n \geq 3$, nếu ước chung lớn nhất của n số là x thì tồn tại $n-1$ số liên tiếp trong n số đó có ước chung lớn nhất là x.

Với điều ***1***, ta có thể dễ dàng chứng minh như sau:

Giả sử dãy a có 2 phần tử, ta gọi 2 phần tử đó là i và j. Ta thêm một phần tử k vào dãy a và gọi gcd của 3 số này là x. Khi đó:

$$x = gcd(i,j,k) = gcd(gcd(i,j),k)$$

Vì $x = gcd(gcd(i,j),k)$ nên:

$$x \leq min(gcd(i,j),k)$$

$$\rightarrow x \leq gcd(i,j)$$

$$\rightarrow gcd(i,j,k) \leq gcd(i,j)$$ **(điều phải chứng minh)**

Ta có thể mở rộng chứng minh trên ra thành 3, 4 hay n số.

Điều ***2*** có thể được chứng minh như sau:

Gọi n số trên là dãy a và $gcd(a_1, a_2, \cdots, a_n) = x$. Khi đó:

$$gcd\left(\frac{a_1}{x}, \frac{a_2}{x}, \cdots, \frac{a_n}{x}\right) = 1$$

Gọi dãy $\left(\frac{a_1}{x}, \frac{a_2}{x}, \cdots, \frac{a_n}{x}\right)$ là dãy b. Nếu $gcd(b) = 1$ thì ta có thể suy ra rằng tồn tại 1 cặp số trong dãy b sao cho 2 số này nguyên tố cùng nhau (vì nếu tất cả các số trong dãy này không nguyên tố cùng nhau thì $gcd(b) > 1$).

Gọi 2 số nguyên tố cùng nhau này là b_i và b_j. Khi đó:

$$gcd(b_i, b_{i+1}, \cdots, b_j) = 1$$

$$\rightarrow gcd(b_1, b_2, \cdots, b_i, \cdots, b_j, \cdots, b_{n-1}) = 1$$

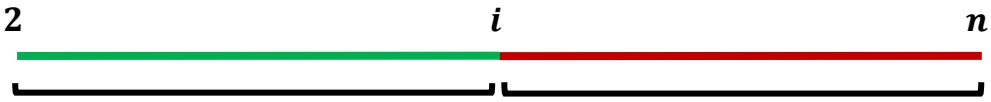
$$\rightarrow gcd(b_1 * x, b_2 * x, \cdots, b_i * x, \cdots, b_j * x, \cdots, b_{n-1}) = x$$

$$\rightarrow gcd(a_1, a_2, \cdots, a_i, \cdots, a_j, \cdots, a_{n-1}) = x$$

→ Tồn tại $n-1$ số liên tiếp trong dãy sao cho ước chung lớn nhất của các số này là x **(điều phải chứng minh)**.

Lưu ý rằng điều trên không áp dụng được với $n = 2$, vì nếu ước chung lớn nhất của 2 số là x thì chưa chắc một trong 2 số đó là x. Ví dụ, $gcd(18, 12) = 6$ trong khi $18 \neq 6$ và $12 \neq 6$.

Thế 2 điều trên giúp ích gì cho bài toán của chúng ta? Hãy gọi độ dài của đoạn liên tiếp dài nhất có $gcd = k$ là i. Nếu bạn lần lượt xét các độ dài và ghi nhận nó lên trục thì nó sẽ trông như sau:



Với đoạn có độ dài từ 2 đến i sẽ tồn tại gcd = k vì theo điều 2 thì nếu tồn tại đoạn i phần tử có gcd = k thì sẽ tồn tại đoạn i - 1 phần tử có gcd = k	Với độ dài từ i + 1 đến n sẽ không tồn tại gcd = k vì theo điều 1 thì độ dài tăng thì gcd chỉ giảm hoặc giữ nguyên

Với minh họa này, không khó để nhận thấy rằng ta có thể thực hiện *tìm kiếm nhị phân* trên độ dài của đoạn. Qua đó, ta giảm được thời gian chạy của thuật từ $O(N^3)$ xuống còn $O(N^2 * \log_2 N)$, vừa đủ để chạy $N = 2000$. Code mẫu:

```
// Hàm found ở đây dùng để xét gcd của các đoạn có độ dài d cho trước, bạn có thể tham khảo lại đoạn code ở Subtask 1 để viết hàm found.

if(found(1) == false && found(2) == false) return 0;

// Nếu các đoạn có d = 1 và d = 2 không tìm thấy thì đương nhiên sẽ không thể tìm thấy các đoạn có d > 2.

// Vì theo điều 2 thì nếu tồn tại đoạn d > 2 thì phải có một đoạn d = 2. Điều 2 không áp dụng cho đoạn có d = 1 nên ta phải xét luôn cả d = 1.


int ans = 0;

if(found(1) == true) ans = 1;

// Như đã nói thì điều 2 không áp dụng cho đoạn d = 1 nên phải xét riêng.

// Đoạn code dưới đây dùng để tìm kiếm nhị phân
```

```
int l = 2, r = n, mid; // Bắt đầu xét từ những đoạn có độ dài 2
while(l <= r){
	mid = (l + r) / 2;
	if(found(mid) == true){
		l = mid + 1;
		ans = mid;
	}
	else r = mid - 1;
}

return ans;
```

Subtask 3

Bạn vẫn còn nhớ cái thuật toán "ngộ nhận" ở phần giới thiệu chứ? Vâng, nó không phải được thêm vào cho vui đâu, vì chỉ cần "tinh chỉnh" một tí là ta đã có một thuật toán hoàn hảo cho bài này.

Đầu tiên thay vì đi tìm đoạn dài nhất $\vdots k$ thì ta sẽ xét toàn bộ các đoạn $\vdots k$. Vì các đoạn ta đang xét $\vdots k$ nên ta nhận thấy rằng gcd của những đoạn đó chỉ có thể là k hoặc bội của k.

Theo điều ***1*** mình đã đề cập ở ***Subtask 2*** thì nếu độ dài của một dãy nào đó tăng thì gcd của dãy đó chỉ có thể giữ nguyên hoặc giảm. Mà nếu gcd của đoạn đang xét là bội của k thì theo điều ***1***, gcd của các đoạn con trong đó chỉ có thể là bội của k hoặc hơn. Từ đó suy ra rằng những đoạn mà có gcd là bội của k thì mọi đoạn con của nó không thể nào có $gcd = k$, và ta có thể quăng những đoạn này vào sọt rác.

Với những đoạn đang xét mà có gcd bằng đúng k thì đây chính là chìa khóa cho bài toán. Những đoạn này có thể được coi là dài nhất, vì 2 phần tử trước đầu và sau đuôi đều là những số không $\vdots k$ (nếu chúng $\vdots k$ thì chúng đã được thêm vào đoạn của chúng ta). Điều cuối cùng ta cần phải làm chỉ là tìm đoạn dài nhất trong số các đoạn có $gcd = k$ và cho kết quả, một vòng lặp đơn giản có thể thực hiện điều này trong $O(N)$.

Ngoài ra, ta cũng có thể thử cách tiếp cận khác là giảm thời gian chạy của hàm *found()* trong ***Subtask 2***. Điều này có thể thực hiện được sử dụng các cấu trúc dữ liệu lấy truy vấn nhanh như *sparse table* - $O(1)$, *segment tree* (*cây phân đoạn*) - $O(\log_2 N)$ hoặc là *SQRT decomposition* (*chia căn*) - $O(\sqrt{N})$. Bằng cách đó, với mỗi lần gọi hàm *found()* thì thời gian chạy giảm xuống chỉ còn $O(N)$ với *sparse table* hoặc $O(\mathrm{N} * \log_2 N)$ nếu bạn thích dùng *segment tree* hơn.

Theo thực nghiệm của mình thì chia căn vẫn chưa đủ tốt để làm bài này, *segment tree* thì "mấp mé bên bờ vực thẳm" còn *sparse table* thì vừa đủ nhanh khi chạy với độ lớn 10^5. Nhưng nếu đề bài nâng lên thành 10^6 thì *sparse table* cũng sẽ khó mà ăn hết.

Code $O(N)$

Code sparse table

Code segment tree

Tổng độ phức tạp của bài toán: $O(N)$ hoặc $O(N * \log_2 N)$ hoặc $O(N * (\log_2 N)^2)$.