# Анализ на задача segment

*Тагове: двоично търсене, теория на числата, монотонен стек, най-дълга подредица, изпълняваща условие, sparse table, разделяй и владей/divide & conquer*

За лекота на езика ще ползвам “gcd“ вместо “най-голям общ делител“.

## *Решение на първа и втора подзадача - 3/7 точки*

Тук итерираме върху всички възможни интервали $[l, r]$ и търсим gcd-то на числата в него. Разликата между двете решения е дали поддържаме gcd, докато обикаляме интервалите или всеки път го преизчисляваме.

## *Решение на трета подзадача - 6 точки*

Тук имаме допълнителното ограничение, че числата са само прости. Тогава отговорът ще е или дължината на цялата редица или някоя максимална по включване подредица, която се състои единствено от равни числа.

## *Решение на четвърта подзадача - 16 точки*

Подзадачата се поддава на различни решения. Ще обясня три, но няма да се очудя да има и други.

Преди всичко ще започнем като превърнем всяко $A_i := log_2(A_i)$. Така числата ще са в интервала $[1, 31]$.

## *Решение 1 - divide and conquer чрез sparse table*

Ако избера цялата редица за интервал $[l, r]$, ограничение върху gcd-то ще поставят най-малките елементи $A_{i_1}, A_{i_2}, ...A_{i_k}$. Евентуално, ако избера интервал, който не включва тях, той може да има по-голям отговор - това са интервалите $[1, i_1 - 1], [i_1 + 1, i_2 - 1]...[i_k + 1, N]$. Тези интервали отново могат да се разделят спрямо най-малките елементи в тях. Така получаваме решение тип ”разделяй и владей”. За бързо намиране на най-малките елементи в интервалите ще ползваме sparse table и ще получим сложност $O(Nlog_2N)$.

## *Решение 2 - монотонен стек*

Ако си представим редицата $A$ като хистограма, виждаме, че в тази подзадача отговорът съвпада с максималният правоъгълник в хистограмата. Тази задача можем да решим стандартно с монотонен стек, който намира първия по-малък елемент в ляво/дясно от всеки елемент. Това решение всъщност не се нуждае от трансформацията в началото, но ако я изпълним можем дори да заобиколим нуждата от монотонен стек и да направим някакво решение със сложност $O(Nlog_2N)$. Не съм го написал и то остава като упражнение за читателя.

## *Решение 3 - фиксиране на отговора*

Можем предварително да си фиксираме gcd-то(има 31 възможности за неговата стойност) и с линейно обхождане да намерим най-дългата подредица, такава че всеки елемент в нея да има стойност по-голяма или равна на фиксираният ни избор.

## *Решение на пета/шеста подзадача - 28/48 точки*

Основно наблюдение за тази задача е, че "префиксните gcd-та могат да заемат $log_2$ различни стойности". Да си представим, че постепенно добавяме числа към набора, на който взимаме gcd. Как може да се промени gcd-то? Евентуално, новото ни число може да постави ново ограничение върху броя прости делители $p$ - преди сме имали 5 пъти 7, сега сме добавили $7^4$ и съответно губим едната седмица. Така броят различни gcd-та е по-малък или равен на броя прости делители на началното число. Те пък са $log_2 A$, защото най-"лошия" случай се получава, когато имаме най-малки прости делители.

След като сме стигнали до това наблюдение можем да стигнем до следното решение: за всеки избор на $i$ ще търсим чрез двоично търсене най-далечните позиции, които имат някоя от $log_2 A$ стойностите на gcd-то. В зависимост как се справяме с намирането на gcd над числата в интервал(със сегментно или със sparse table) получаваме две решения със сложности $O(N(log_2 N)^2 log_2 A)$ или $O(N log_2 N log_2 A)$.

## *Решение на цялата задача - 100 точки*

Миналото ни решение страда от проблема, че за всяка позиция пресмятаме наново къде се падат показалките, които сочат към най-далечните позиции с различни gcd-та. Това щеше да е резонно, ако gcd върху интервал се държеше непредсказуемо, но това не е така.

Обикновено се оказва, че да "добавяме" неща(елементи към редица или заявка) е по-лесно за измисляне, така че ще разглеждаме задачата по следния начин: имаме информация за интервалите, които започват от позиция $l$(нека десните им граници са $r_1, r_2, ..r_k$, а gcd-тата им да са $g_1, g_2, ...g_k$) и ще добавим $A_{l-1}$ към тях. Как ще се променят старите интервалите? Gcd-тата им ще станат $g'_i = gcd(g_i, A_{l-1})$. След тази операция, някои интервали евентуално ще имат равни gcd-та и така ще се "слеят" в един. Относно нови интервали - ще се добави най-много един интервал $[l-1, l-1]$. Така всяка позиция $l-1$ получи своите $logA$ интервали чрез интервалите на тази пред нея $l$. Това се оказва, че сваля сложността на $O(NlogA)$ макар че все още не е кристално ясно защо.

За да финализираме доказателството трябва да отбележим, че истинската сложност на gcd функцията/алгоритъма на Евклид е $O(log(min(a,b)/gcd(a,b)))$, защото:

- Броят операции, нужни за да изчислим $gcd(a,b)$ съвпадат с тези нужни за изчислението на $gcd(a \times d, b \times d)$. Тогава можем да изкараме gcd-то на $a$ и $b$ пред $gcd$ функцията и да смятаме $gcd(a', b')$.
- $min(a,b)$ всъщност е малко "abuse of notation". Обикновено алгоритъма на Евклид има сложност $O(log(max(a,b)))$, но след една стъпка по-голямото от двете числа всъщност ще е по-малкото от началните числа. Реално игнорираме една операция на алгоритъма, но тя не е огромен проблем.

Благодарение на свойствата на функцията $log$ горната оценка на сложността става $O(log(min(a,b)/gcd(a,b))) = O(log(min(a,b)) - log(gcd(a,b)))$. Това изразяване на сложността ще ни е много удобно в случая, защото ни интересува сбора от извършването на $logA$ алгоритъма на Евклид, всеки от които работи с резултата на миналия и някое ново число. Сумарната им сложност ще е $\sum log(G_i) - log(G_{i+1}) = log(G_0) - log(G_k) \leq log(G_k)$. Това ще се извърши за всяка позиция, така че сложността е точно $O(NlogA)$.

## *Divide & conquer решение*

Задачи, които се свеждат до “пребройте интервали/намерете екстремен интервал“ се поддават на идеи тип ”разделяй и владей”. Тази не беше изключение - имплементацията на това решение се намира във файл **viktor_dnc.cpp**. Сложността е $O(N(logA + logN))$ с аргументи подобни на тези от главното решение.

## *Рандомизирано решение*

Решението беше следното: ще направим brute force проверка за всички интервали с размер $\leq C$ за някакво разумно малко $C$. Ако отговорът е голям интервал обаче, така ще го изпуснем, но пък за този случай ще се възползваме от именно големината му. $K$ на брой пъти ще изберем случайна позиция и ще търсим интервалите с различни стойности на gcd вляво и вдясно. Шансът за успех ще е:

$$\frac{C}{N}\left(\left(\frac{N-C}{N}\right)^{K+1}-1\right)$$

Въпрос на вкус е да изберем удобни $C$ и $K$. За изборът на $K$ можем пък и напълно да го игнорираме и да избираме случайна позиция с някаква time bomb-а.

Трета и четвърта подзадача предлагат възможност на състезатели, които не успяват да решат цялата задача да изкарат някакви допълнителни точки. Силно препоръчвам да се оглеждате за такива възможности. Понякога в процеса на решаване на такава “странична“ подзадача човек може да измисли цялата задача. Такъв е случаят с подзадача 6 Есенен 2023 A1.jumps.

*Релевантни материали:*

- блог в codeforces.com за сложността на $gcd$
- подобна задача в codeforces.com
- монотонен стек в usaco.guide
- sparse table в cp-algo
- Лекция на тема ”Разделяй и владей” на Дениз Потурлиев от НШИ-Ловеч 2023

*Автор: Иван Лупов*