Анализ на задача **rich_num**

Подзадача 1: Дадена е за възможно най-brute force идеята - за всяко число от всяка заявка правим $O(\sqrt{N})$ проверки.
Подзадача 2: Миналото решение е естествено супер бавно - как да го подобрим? Тук имаме добавеното ограничение, че $V = 10$ винаги, тоест дали едно число е "богато" или не не се мени в различните заявки - това предполага да precompute-нем за всяко число статуса му, а допълнителното ограничение $L = 1$ за всяка заявка ни позволява с префиксни суми да отговаряме със сложност $O(1)$.
Подзадача 3: Тук имаме няколко промени спрямо миналата подзадача. Първо $V \leq 10$, а не равно на 10. Не е голям проблем, можем да си направим 10 префиксни масива и да спасим положението. $L = 1$ не е гарантирано, но това също може лесно да се заобиколи чрез префиксните ни суми - $prefix_r - prefix_{l-1}$ пак ни дава отговора в $O(1)$ сложност.

Вдигането на ограничението над R - от 10^4 на 10^5 - трябва да ни досети, че $O(N\sqrt{N})$ precomputing всъщност е малко бавно. Есенцията на идеята е, че всяко число ще се "обърне назад" и ще потърси делителите си. Това обаче ще ни създаде твърде много ненужни проверки(49 няма нужда да провери дали 6 му е делител). Затова ще обърнем логиката на обратно - вместо числото да търси делителите си, нека си представим, че всяко число казва на кратните си, че им е делител(5 ще "каже" на 10, 15, 20, 25…). Тук няма място за такива "ненужни проверки". Така всъщност оптимизираме precomputing-а от $O(N\sqrt{N})$ на $O(NlogN)$. Защо точно $O(NlogN)$? Следствие от harmonic series - може да го потърсите в интернет.

Важно, за да продължим с решението ни, е да отбележим странното ограничение $V \leq 10$. Точната стойност не е толкова интересна. Нека се замислим обаче какво се случва с числата, чиято сума от делители е по-голяма от 10. Независимо от какво точно е V за конкретната заявка, те винаги ще са "богати".
Подзадача 4: Тук вече нямаме допълнителни ограничения. За да продължим досегашното ни решение, последното наблюдение от миналата подзадача се оказва доста полезно - имаме някакъв случай, в който можем да добавим накуп числа към групата на "богатите". Какво точно означава това? Нека досега сме работили върху заявка с някакво V. Ако следващата ни заявка има $V - 1$, то можем да добавим към отговора броя на числата със сума на делителите равна на $V - 1$. Обаче да получим удобни две поредни заявки иска много късмет. Тук забелязваме, че задачата е offline, тоест можем да си сортираме заявките, така че да са ни удобни, а после само да върнем отговорите на правилните им места.

Как ще се случва това "добавяне"? Като си представим префиксните ни суми в момента и как те ще се променят - това ще е просто една 0 да стане 1, едно "небогато" число да стане "богато". Опитните състезатели бързо ще забележат, че тук трябва да махнем префиксните масиви и да ги заменим със сегментно дърво.

Финалната ни сложност е $O(NlogN)$.

Задачата има и алтернативно решение с merge sort tree, но то не е очаквано, понеже не е в syllabus-а за състезанието. Българските състезатели могат да погледнат НОИ-3 2022 D4.izobilni, за по-лесна версия на тази задача.

Автор: Иван Лупов