Задачі A, F, G є аналогами певних задач з ресурсу codeforces.com

Розбір задачі «Арчі та кольори»

Підтримуватимемо множину неперетинних відрізків стрічки пофарбованих в однаковий колір. При виконанні запиту першого типу кількість таких відрізків не може збільшитись більше ніж на 2. Також при зміні множини потрібно додавати на певних підвідрізках масиву a ціле число — необхідно скористатись технікою додавання на відрізку.

Складність: $O((n+m) \cdot log(n+m))$ часу та $O(n+m)$ пам'яті.

Розбір задачі «Арчі та підпослідовності»

Помітимо, що існує підпослідовність мінімальної ціни якій належить найлівіший мінімальний елемент масиву a. Оскільки ціна циклічно зсунутої підпослідовності рівна ціні цієї підпослідовності, то після циклічного зсуву масиву a задача зовсім не зміниться. Циклічно зсунемо масив a так, щоб найлівіший мінімальний елемент масиву a був першим елементом масиву після зсуву.

Скористаємось ідеєю бінарного пошуку по відповіді. Реалізуємо функцію $get(x)$ яка перевірятиме чи існує підпослідовність довжини k з ціною не більше x.

Нехай dp_i — довжина найдовшої підпослідовності з таких, що їх першим елементом є a_1, останнім — a_i, та для будь-якої пари їх сусідніх елементів сума цих елементів не перевищує x. $dp_i = \max\limits_{a_i+a_j \leqslant x}(dp_j + 1)$, де $j < i$. Ітеруватимемось по масиву a зліва направо та оновлюватимемо значення елементу з номером a_i дерева відрізків через значення dp_i. Тоді значення dp_i легко рахувати знаючи максимум в дереві відрізків на інтервалі $(-\infty; x - a_i]$.

$get(x)$ має повернути сигнал про те що потрібна підпослідовність існує, якщо існує таке i, що $a_1 + a_i \leqslant x$ та $dp_i \geqslant k$.

Складність: $O(n \cdot log(n) \cdot log(ans))$ часу та $O(n)$ пам'яті.

Розбір задачі «Арчі та дружні точки»

Помітимо, що пара точок дружня якщо прямокутник побудований на них як на протилежних вершинах не містить інших точок множини.

Відсортуємо точки. Порахуємо кількість пар дружніх точок з номерами $i < j$ у відсортованому масиві точок таких, що $y_i > y_j$. Аналогічним способом можна порахувати кількість пар дружніх точок з номерами $i < j$ у відсортованому масиві точок таких, що $y_i < y_j$. Випадок $y_i = y_j$ тривіальний.

Скористаємося трюком divide and conquer. Нехай $solve(l,\ r)$ — шукане значення для підмасиву точок з номерами $l, l+1, ..., r$. $solve(l,\ r) = solve(l,\ m) + solve(m+1,\ r) + P$, де P — кількість пар дружніх точок з номерами $i < j$ ($y_i > y_j$) такими, що $l \leqslant i \leqslant m < j \leqslant r$.

Для кожної точки з номером i з множини $l, l+1, ..., m$ знайдемо найвищу (за значенням y координати) точку з номером j таку, що $l \leqslant j \leqslant m$, $j \neq i$, $x_i \leqslant x_j$ та $y_i \geqslant y_j$. Випишемо масив пар L, де $L_i = (y_i,\ y$ координата описаної точки) для $l \leqslant i \leqslant m$.

Для кожної точки з номером i з множини $m+1, ..., r$ знайдемо найнижчу (за значенням y координати) точку з номером j таку, що $m < j \leqslant r$, $j \neq i$, $x_j \leqslant x_i$ та $y_j \geqslant y_i$. Випишемо масив пар R, де $R_i = (y_i,\ y$ координата описаної точки) для $m < i \leqslant r$.

Нескладно помітити, що P рівне кількості пар номерів $(i,\ j)$ таких, що $L_i.second < R_j.first < L_i.first < R_j.second$. Щоб порахувати це значення скористаємось ідеєю скануючої прямої. Додамо події для кожного зі значень з цих масивів пар. Ітеруватимемось по них від менших значень y до більших.

4 типи подій:

- $L_i.second$ — віднімаємо від P суму на інтервалі дерева відрізків $(L_i.first; +\infty)$
- $R_j.first$ — додаємо значення 1 до елементу дерева відрізків з номером $R_j.second$
- $L_i.first$ — додаємо до P суму на інтервалі дерева відрізків $(L_i.first; +\infty)$
- $R_j.second$ — ігноруємо :)

Складність алгоритму можна оцінити наступним чином: $T(n) = 2 \cdot T(\frac{n}{2}) + O(n \cdot log(n))$.

Складність: $O(n \cdot log^2(n))$ часу та $O(n \cdot log(n))$ пам'яті.

Розбір задачі «Арчі та граф»

Після покраски Арчі у графі не може існувати "білого циклу". Якщо після покраски Арчі залишиться "білий ліс", то згідно з описаним алгоритмом його буде покрашено у чорний колір (завжди існуватиме біле ребро інцидентне вершині степені 1 цього "білого лісу").

Нескладно помітити, що відповідь рівна $m-$(кількість ребер остовного лісу графу).

Складність: $O(n + m)$ часу та $O(n + m)$ пам'яті.

Розбір задачі «Арчі та GCD»

Вважатимемо, що елемент a_1 не буде видалено. Тоді GCD масиву a після видалення з нього не більше ніж k його елементів є дільником числа a_1. Знайдемо для кожного дільника d числа a_1 кількість чисел з a що діляться на нього та перевіримо чи їх кількість не менша за $(n - k)$ — якщо це так то оновимо відповідь через d.

Факторизуємо число a_1. Після перебору його можливих простих дільників до $\sqrt[3]{a}$ у нього залишаться не факторизованими не більше ніж 2 прості дільники. Далі a' — число що утворилось після цієї часткової факторизації (яке також необхідно "дофакторизувати"). Якшо існує таке i, що $GCD(a', a_i) \in (1; a')$ то значення $GCD(a', a_i)$ — один з цих простих дільників. Інакше, в рамках подальшого розв'язку задачі можна вважати що a' — просте (неподільна частина факторизації), незалежно від того чи просте воно насправді.

Нехай описана факторизація має наступний вигляд: $a_1 = p_1^{b_1} \cdot p_2^{b_2} \cdot ... \cdot p_k^{b_k}$. Визначимо $dp_{i,j}$ — кількість елементів масиву a що діляться на дільник j числа a_1, за умови що степінь з яким входить p_l в розклад такого елементу збігається зі степінню з якою входить p_l в розклад j для всіх $l \geqslant i$. Початкові значення динаміки: $dp_{1,GCD(a_1,a_i)}++$ для всіх можливих i. Переходи динаміки: $dp_{i,j} \to dp_{i+1,j/(p_i^t)}$.

Перемішаємо масив a і виконаємо описаний алгоритм ще раз. Потім ще раз ... Оскільки $2 \cdot k \leqslant n$, то на кожному такому кроці вірогідність того що a_1 не буде видалено не менша за $\frac{1}{2}$.

Складність: $O(K \cdot (\sqrt[3]{a} + n \cdot log(a) + D \cdot \frac{log(a)}{log(log(a))}))$ часу та $O(n + D \cdot \frac{log(a)}{log(log(a))})$ пам'яті при вірогідності помилки не більше 2^{-K}.

Тут K — кількість ітерацій описаного алгоритму, а D — кількість дільників a ($D \approx O(\sqrt[3]{a})$).

Розбір задачі «Арчі та хрінька»

Розглянемо задачу у вигляді руху зліва направо вікна довжини len по масиву a. Необхідно досягати максимуму на ньому, а потім здвигати його на одну позицію вправо.

Нехай зафіксовано положення вікна. Розглянемо окремо всі додатні та від'ємні числа у вікні. Якщо додатних не більше ніж k, або від'ємних не більше ніж k, то можна перетворити всі доданки суми на доданки однакового знаку за допомогою масиву p. Тоді відповідь цього вікна — сума абсолютних значень чисел вікна.

Нескладно помітити, що за допомогою масиву p неоптимально одночасно робити якісь додатні доданки суми від'ємними та від'ємні додатними. Переберемо групу доданків (додатні, від'ємні) знак яких будемо змінювати за допомогою масиву p. Очевидно, що вигідно змінювати знак k максимальних за модулем доданків.

При здвигах вікна підтримуватимемо два дерева відрізків: одне для додатних чисел, інше для від'ємних. У вершинах зберігатимемо пари (кількість чисел у відповідному піддереві, сума цих чисел). Виконувати потрібний запит можна за допомогою техніки каскадного спуску по такому дереву.

Складність: $O(n \cdot log(n))$ часу та $O(n)$ пам'яті.

Розбір задачі «Арчі та максимальні мінімуми»

Помітимо, що для певного запиту можна скористатись ідеєю бінарного пошуку по відповіді. Нехай зафіксовано висоту h і необхідно дізнатись чи існує підвідрізок підвідрізку $[l; r]$ довжини w такий, що всі його елементи не менше h та нехай всім елементам масиву a в дереві відрізків які

не менше h відповідає число 1, а тим що менше h — 0. Тоді все що необхідно дізнатись — чи існує підвідрізок довжини хоча б w з одиниць на відповідному підвідрізку дерева? Такий запит легко виконувати підтримуючи у кожній вершині дерева відрізків три значення: довжину найдовшого підвідрізка з одиниць, довжину найдовшого префікса з одиниць та довжину найдовшого суфікса з одиниць відповідного підмасиву.

Змінювати елементи дерева відрізків потрібно у порядку зменшення відповідних елементів масиву a.

Щоб знайти відповіді на всі запити потрібно або скористатись паралельним бінарним пошуком, або ж будувати описане дерево персистентно.

Складність: $O((n+m) \cdot log^2(n))$ часу та $O(n+m)$ пам'яті.

Розбір задачі «Арчі та суми підмасивів»

Скористаємось ідеєю бінарного пошуку по відповіді. Реалізуємо функцію $get(x)$ яка знаходитиме кількість підвідрізків масиву з сумою чисел не більше x.

Порахувавши масив префіксних сум $pref$ ($pref_0 = 0, pref_i = a_1 + a_2 + ... + a_i$) рахувати $get(x)$ стає достатньо просто: $get(x)$ рівний кількості пар $1 \leqslant i \leqslant j \leqslant n$ таких, що $pref_j - pref_{i-1} \leqslant x$. Ітеруючись по i зліва направо по масиву $pref$ та додаючи значення 1 до елементу з номером $pref_{i-1}$ дерева відрізків до результату необхідно буде додати суму на інтервалі $[pref_i - x; +\infty)$ дерева відрізків.

Для зручності всі значення $pref_i$ та $pref_i - x$ можна змасштабувати.

Складність: $O(n \cdot log(n) \cdot log(ans))$ часу та $O(n)$ пам'яті.

Розбір задачі «Арчі та додавання на відрізку»

Будуватимемо відповідь як послідовність віднімань числа 1 на підвідрізках масиву a, допоки всі числа масиву не будуть рівними 0.

Поки у масиві всі числа додатні то найбільш вигідним кроком є відняти 1 на всьому масиві, а коли у масиві з'являється хоча б одне значення 0 то задача стає незалежною для підвідрізків на які розбивають масив a нульові елементи.

Складність: $O((ans+n) \cdot log(n))$ часу та $O(ans+n)$ пам'яті.