Задача A. Масив і ще кілька масивів

Автор задачі: **Антон Тригуб**
Задачу підготував: **Владислав Денисюк**
Розбір написав: **Владислав Денисюк**

Блок 1

Оскільки ми лише можемо додати однакове число d до усіх елементів масиву, то додамо таке число d, щоб максимальний елемент масиву став рівний n, а мінімальний — 1, якщо це неможливо, то відповідь «`No`». Інакше потрібно додати d до усіх елементів масиву і перевірити, чи дійсно після такої операції усі числа будуть різними, якщо так, то відповідь «`Yes`», інакше «`No`».

Блок 2

У цій групі кожен масив можна уявити як інтервал чисел. Фактично ми маємо k інтервалів, ми завжди можемо додати для кожного з них числа так, щоб усі числа на першому інтервалі були від 1 до l_1, на другому - від $l_1 + 1$ до $l_1 + l_2$, і так далі. Ми уявляємо собі масиви як інтервали чисел які ми можемо рухати, і наша ціль — скласти з них інтервал $[1; n]$.

Блок 3

У нас є дві групи, оскільки наша ціль — утворити n різних цілих чисел, то така множина чисел матиме лише один мінімум — число 1. Далі помітимо, що після додавання відповідних чисел, якщо вони формують правильну відповідь, 1 з'явиться або в першому або в другому масиві. Переберемо обидва варіанти. Нехай ми розглядаємо варіант, де мінімум після додавання буде у першому масиві. Нехай мінімум у першому масиві до додавання був m_1, тоді потрібно буде додати $1 - m_1$ до першого масиву, щоб його мінімум став рівний 1. Потім у нас залишиться другий масив і множина чисел, якої нам не вистачає — числа від 1 до n, яких немає в першому масиві. Нехай найменше таке число — x. Тоді потрібно буде додати $x - m_2$ (де m_2 — мінімум у другому масиві до додавання) до другого масиву і перевірити, чи така комбінація чисел підходить. Аналогічно розглянемо варіант, де мінімум у другому масиві. Якщо жоден варіант не підійшов, то відповідь — «`No`».

Блок 4

Рішення схоже на рішення блоку `3`. Будемо перебирати порядок мінімумів серед цих трьох масивів, тобто будемо перебирати, який масив матиме найменший мінімум, другий найменший, і найбільший. Аналогічно по черзі прирівнюватимемо мінімум першому числу, яке відсутнє у вже оброблених масивах і так будувати відповідь. Всього є 6 варіантів такого порядку `[1 2 3]`, `[1 3 2]`, `[2 1 3]`, `[2 3 1]`, `[3 1 2]`, `[3 2 1]`.

Блок 5

Рішення цього блоку схоже на рішення блоку `2`. Масиви діляться на дві групи — масиви, де після додавання усі числа будуть парні, і масиви, де після додавання усі числа будуть непарні. Групу кожного масиву можна перебрати — всього буде 2^k варіантів. Для кожної з цих груп нам потрібно знайти такі числа d_i, щоб множина чисел в одній групі була рівна `[1 3 5 7...]`, а друга - `[2 4 6 8...]`. Фактично можемо знову уявити масиви як інтервали, як ми це робили в блоці `2`, але тут уже потрібно зважати на те, що у нас просто немає в одному випадку парних точок, а в іншому - непарних.

Блок 6

Помітимо, що усі числа в кінцевій множині повинні бути різні, це означає, що максимальний мінімум рівний k, оскільки максимум рівний n, і в якому масиві він би не був — мінімумом такого масиву буде $n-(n-k) = k$. Тому мінімум кожного масиву після додавання буде від 1 до k. Переберемо значення кожного мінімуму і перевіримо кожен варіант. Складність такого рішення — $O(n \cdot k!)$.

Блок 7

Кожен мінімум буде рівний числу від 1 до n. Переберемо, чому рівний мінімум кожного масиву і перевіримо кожен варіант. Складність такого алгоритму буде $O(n^{k+1})$.

Блок 8

Будемо робити те, що ми робимо в блоці 4 для трьох масивів, але для будь-якої кількості до п'яти. Перебиратимемо перестановку чисел від 1 до k, якою ми задаватимемо відносний порядок мінімумів масивів за зростанням. Для кожної перестановки використаємо алгоритм описаний в блоці 4 — знаходимо мінімальне число, що не зустрічалось у вже пройдених масивах і перетворюємо на нього мінімум масиву, що розглядається. При перевірці всіх таких варіантів ми точно перевіримо усі комбінації чисел що могли б бути відповіддю, оскільки у правильній відповіді усі числа різні, а отже мають відносний порядок. Складність такого рішення — $O(n \cdot k!)$.

Задача B. Масив монет і запити зважування

Автор задачі: **Денисов Костянтин**
Задачу підготував: **Денисов Костянтин**
Розбір написав: **Денисов Костянтин**

Блок 1

З обмеження $k < n$ випливає, що хоча б одна з монет з номерами n та 1 є справжньою. Тому першим запитом, поклавши на чаші першу та n-ту монету, дізнаємося номер однієї справжньої монети. Далі пройдемося по всім монетам зліва направо і порівняємо знайдену справжню монету з поточною монетою, якщо поточна монета легша, то вона і буде першою фальшивою монетою. Отже, маємо рішення, яке використовує не більше n запитів.

Блок 2

Нехай $[l, r]$ — поточний відрізок монет, на якому шукаємо фальшиву монету. Початково $[l, r]$=$[1, n]$. Нехай $m = \lfloor \frac{r-l+1}{2} \rfloor$. Виділимо два підвідрізка $[l, r]$, які не перетинаються: $[l, l + m - 1]$, $[r - m + 1, r]$. Покладемо на першу чашу терезів монетки з номерами з першого відрізка і на другу чашу монети з другого відрізка. Якщо монети з номерами з першого відрізка легші, то на цьому відрізку знаходиться фальшива монета і перейдемо до задачі $[l, r] := [l, l + m - 1]$. Аналогічно коли монетки з номерами з другого відрізка легші. Якщо ж ваги врівноважені, то монетка, що знаходиться між цими відрізками (така буде рівно одна) і є фальшивою.

Отже, кожного разу ми зменшуємо довжину відрізка пошуку принаймні вдвічі, тому маємо рішення, яке використовує не більше $\lceil log_2(n) \rceil \leqslant \lceil log_2(10^4) \rceil \leqslant 14$ запитів.

Блок 3

Використаємо схожу ідею як у попередньому блоці, але будемо ділити відрізок не на дві частини, а на три частини. Нехай $m = \lceil \frac{r-l+1}{3} \rceil$. Покладемо на першу чашу терезів монетки з номерами з відрізка $[l, l + m - 1]$ і на другу чашу монети з відрізка $[r - m + 1, r]$ (такі ж відрізки як у другому блоці, але з іншим m). Коли терези врівноважені, то фальшива монета знаходиться поза даними відрізками, тому продовжуємо пошук на відрізку $[l+m, r-m]$. Випадки, коли терези неврівноважені є аналогічними до другого блоку.

Отже, кожного разу ми зменшуємо довжину відрізка пошуку принаймні втричі, тому маємо рішення, яке використовує не більше $\lceil log_3(n) \rceil \leqslant \lceil log_3(10^4) \rceil \leqslant 9$ запитів.

Блок 4

Нехай $A = \{1, k + 1, 2 \cdot k + 1, ..., d \cdot k + 1\}$, де d — максимальне ціле число, що $d \cdot k + 1 \leqslant n$. Очевидно, що у цій множині є **рівно** одна фальшива монета. Знайдемо її алгоритмом з третього блоку. Нехай r — номер фальшивої монети, що ми отримали.

Тоді перша фальшива монета знаходиться на відрізку $[l, r]$, де $l = \max(1, r-k+1)$. Нехай $real$ — номер якоїсь справжньої монети (таку можна знайти, наприклад, додатковим зважуванням першої та останньої монети або з вже зроблених зважувань, використовуючи, що $|A| > 1$). Тоді можна запустити бінарний пошук по відрізку $[l, r]$, який буде порівнювати монету з номером, що відповідає середині поточного відрізка пошуку, з монетою з номером $real$. Якщо терези врівноважені, то перша фальшива монета буде у правій частині відрізка, інакше — у лівій.

Отже, перша частина рішення використовує не більше $\lceil log_3(\lceil \frac{n}{k} \rceil) \rceil$, а друга частина — $\lceil log_2(k) \rceil$. Отже, маємо рішення, яке використовує не більше $\lceil log_3(\lceil \frac{n}{k} \rceil) \rceil + \lceil log_2(k) \rceil \leqslant 11$ запитів.

Блок 5

Рішення п'ятого блоку повторюється з четвертим блоком крім частини з бінарним пошуком. Виберемо на поточному відрізку пошуку дві монети з номерами c та $d (c < d)$, щоб відрізки, на які ці номери поділяють відрізок були майже рівними за розмірами. Тоді покладемо на кожну чашу терезів дві монети. На першу — монети з номерами c та d. На другу — справжню монету з номером

real та фальшиву монету (одну фальшиву монету ми вже знаємо). Тоді оскільки фальшиві монети у цьому блоці мають однакову вагу, то:

- якщо ваги врівноважені, то монета з номером c — справжня та з номером d — фальшива і треба продовжувати пошук на відрізку $[c+1, d]$;
- якщо перша чаша важче, то монети з номерами c та d справжні і далі шукаємо на відрізку $[d+1, r]$;
- якщо друга чаша важче, то монети з номерами c та d фальшиві і далі шукаємо на відрізку $[l, c-1]$;

Отже, кожного разу відрізок пошуку зменшується принаймні втричі, тоді маємо рішення, що використовує не більше $\lceil log_3(\lceil \frac{n}{k} \rceil) \rceil + \lceil log_3(k) \rceil \leqslant 11$ запитів.

Блок 6

Нехай будемо підтримувати відрізок $[l, r]$, на якому знаходяться всі фальшиві монети. Тоді очевидно, що перша фальшива монета буде знаходитися на відрізку $[l, r-k+1]$. Нехай $m = \lceil \frac{r-k+1-l}{3} \rceil$. Покладемо на першу чашу вагів монетки з номерами з відрізка $[l, l+m-1]$ і на другу чашу монети з відрізка $[r-m+1, r]$. Помітимо, що не може бути такого, що на кожному з цих відрізків є фальшиві монети, оскільки довжина відрізка $[l+m-1, r-m+1] > k$. Тому маємо, що якщо ваги врівноважені, то фальшиві монети знаходяться на відрізку $[l+m, r-m]$. Якщо перша чаша важче, то — на відрізку $[r-m+1-k+1, r]$. Якщо друга чаша важче, то — на відрізку $[l, l+m-1+k-1]$. Будемо продовжувати зменшувати відрізок пошуку, поки його довжина більша за k.

Помітити, що кожного разу ми зменшуємо довжину відрізка, де знаходиться перша фальшива монета принаймні втричі, тому маємо рішення, що використовує не більше $\lceil log_3(n) \rceil \leqslant \lceil log_3(10^4) \rceil \leqslant 9$ запитів.

Задача C. Масив і додавання на відрізку

Автор задачі: **Антон Тригуб**
Задачу підготував: **Владислав Денисюк**
Розбір написав: **Владислав Денисюк**

Блок 1

Перше спостереження — це те, що ми можемо додавати нескінченно великі числа на відрізках і відповідно якщо ми додали на відрізку, то числа, які входять та не входять у відрізок, точно стали відрізнятись між собою. Розглянемо оптимальний вибір відрізків. Тоді назвемо кінці таких відрізків - перегородками. Фактичний зміст перегородки — це те, що інший кінець відрізка буде або зліва або справа від перегородки й відповідно змінить лише одне з чисел які вона розділяє. Тому нам потрібно мінімум $n-1$ перегородок між числами. Якщо жоден з відрізків не має кінця в точках 1 або n - то значення в точках 1 і n будуть рівними оскільки жоден з відрізків їх не чіпає. Тому потрібно принаймні n перегородок. Кожен відрізок має два кінці, тому може утворювати не більше двох перегородок. Відповідно нам потрібно $\lceil \frac{n}{2} \rceil$ перегородок. Покажемо, що такої кількості достатньо. Додаватимемо на відрізках $[1; \lceil \frac{n}{2} \rceil]$, $[2; \lceil \frac{n}{2} \rceil + 1]$, ... Якщо n парне - то в кінці ми додамо на відрізку $[\frac{n}{2}, n-1]$. Якщо ні, то на відрізках $[\lceil \frac{n}{2} \rceil; n-2]$ та $[n-1, n-1]$.

Блок 2

У цьому блоці ми фактично розв'язуємо задачу на двох масивах у яких дві перегородки спільні. Тому відповідь тут - $\lceil \frac{n-2}{2} \rceil$. Але тут потрібно врахувати два крайніх випадки: коли усі числа рівні, тоді потрібно використати розв'язок блоку 1 і випадок, коли лише одне число відмінне від інших - тоді воно або в кінці або на початку і ми розв'язуємо блок 1, але в масиві $n-1$ чисел - тому відповідь $\lceil \frac{n-1}{2} \rceil$.

Блок 3

У цьому блоці ми можемо використати рекурсивний перебір. Додавання на відрізку буде відрізняти числа на відрізку від чисел за межами відрізка. Всього ми можемо так зробити не більше $\lceil \frac{n}{2} \rceil$ разів, оскільки відповідь коли всі числа однакові точно не краще за відповідь коли деякі з них - різні. Таким чином маємо рекурсію де ми перебираємо n^2 варіантів відрізка $\lceil \frac{n}{2} \rceil$. Складність $O(n^n)$ що при максимальному n - $8^8 = 16777216$, це повинно проходити по часу.

Блок 4

Оскільки числа на позиціях 1 і n рівні одна з перегородок буде на початку або в кінці, також можна уявити що елементи 1 і n - сусідні, тобто числа розташовані по колу, і тоді нам потрібно розставити перегородки по колу, це нічого не змінює для даного блоку, оскільки ми завжди ставимо перегородку між 1 і n, але у наступних блоках така інтуїція буде корисною. Щоб розставити всі інші перегородки для кожного значення від 1 до n згенеруємо інтервали позицій на яких би ми могли розставити перегородки для чисел з даним значенням. Тепер у нас є кілька інтервалів і нам потрібно вибрати мінімальну кількість точок, щоб кожному інтервалу належала хоча б одна точка. Посортуємо кінці відрізків і додаватимемо відрізок коли зустрічатимемо лівий кінець. Коли ми зустріли правий кінець відрізка, якому ще не належить жодна перегородка ми не можемо поставити найлівішу з ще не поставлених перегородок правіше від даної позиції. Якщо ж ми поставимо таку перегородку лівіше - то це буде точно не краще, оскільки поставивши перегородку в даній позиції ми задовольнимо всі незадовільнені відрізки, що мають кінці лівіше або в даній точці. Тому поставимо таку перегородку і відмітимо всі незадовільнені відрізки у яких ми вже зустріли ліві кінці - як задовільнені. Повторимо поки не пройдемо всі відрізки. Отримаємо P перегородок і ще мусимо поставити додаткову перегородку на початку або в кінці, в сумі матимемо $P+1$ перегородок. Аналогічно конструкції з блоку 1 ми можемо знайти відрізки що задовольнять умову з кінцями в цих перегородках. Відповідь $\lceil \frac{P+1}{2} \rceil$.

Блок 5

Для будь-якої групи чисел помітимо що конструкція на перегородках яку ми будуємо в блоці 1 фактично теж повинна бути витримана відносно перегородок, оскільки при її побудові підхід є аксіоматичним і всі інші конструкції містять в собі дану. Помітимо, що якщо зациклити масив і ставити перегородки на такому масиві то нічого не змінюється, як бонус - ми автоматично отримуємо виконання умови, що перегородка має бути перед першим елементом або після останнього. Якщо ж масив циклічний - то "перед першим"і "після останнього це одне й те ж. Ми можемо перебрати точку де ми точно ставимо перегородку, взяти її за початок масиву і запустити сканлайн подібний до рішення в блоці 4.

Блок 6

Цей блок був також було створено для неоптимальних реалізацій. Зокрема в ньому немає потреби зжимати числа в масиві.

Блок 7

Для кожної точки порахуємо наступну точку справа в якій ми муситимемо поставити нову перегородку, це можна зробити знайшовши найближчий справа (по колу звісно) кінець відрізка, який ми не перетинаємо наразі, наприклад це можна порахувати сканлайном. Потім побудуємо граф на таких переходах. Кожна позиція має однозначного наступника, тому побудуємо бінарні підйоми на такому графі. Для кожної позиції порахуємо скільки перегородок потрібно поставити, якщо одну з перегородок ми ставимо в даній позиції і переходами ми покриваємо весь масив по колу. Складність $O(n * log(n))$.

Задача D. Масив і часткові суми

Автор задачі: **Valerio Stancanelli**
Задачу підготував: **Alessandro Bortolin**
Розбір написав: **Роман Білий**

Блок 1

Якщо усі числа 0 або 1, то відповідь рівна 0. Якщо ж усі числа -1 або 0, то відповідь рівна 1 — можна виконати одну операцію першого типу.

Тепер нехай ми хочемо зробити одну операцію 2 типу. Покажемо, що ми можемо її виконати на всьому масиві. Нехай існує проміжок $[l..r]$, що виконавши операцію другого типу на цьому проміжку, всі елементи стали невід'ємними. Для цього необхідно, щоб усі елементи $a_1, a_2, \ldots, a_{l-1}$ та $a_{r+1}, \ldots, a_n$ були невід'ємними. А отже після виконання операції другого типу на всьому масиві всі елементи стануть не меншими, ніж якщо виконати операцію на $[l..r]$. Отже ми можемо спробувати виконати її на всьому масиві і подивитись, чи стали всі елементи невід'ємними.

Аналогічно, спробуємо зробити операцію 3 типу на всьому масиві.

Блок 2

Розглянемо будь-який елемент зі значенням 1 (якщо такого не існує, зробимо одну операцію першого типу). Нехай його позиція p. Будемо робити операції другого типу на проміжках $[p..r_i]$ так, щоб кожного разу усі елементи, які замінюються, були невід'ємними. Наприклад, якщо масив $[1, -1, 0, -1, 1, -1, -1]$ та $p = 1$ (нумерація від 1), то зробимо операції на відрізках [1..3], [1..6], [1..7]: $[1, -1, 0, -1, 1, -1, -1] \to [1, 0, 0, -1, 1, -1, -1] \to [1, 1, 1, 0, 1, 0, -1] \to [1, 2, 3, 3, 4, 4, 3]$.

Таким чином ми зможемо зробити усі елементи правіше p невід'ємними. Аналогічно операціями 3 типу зробимо усі елементи лівіше p невід'ємними.

Найгірший випадок для такого розв'язку — усі елементи крім одного рівні -1. Можна експериментально визначити, що в такому випадку потрібно не більше 7 операцій 2 типу і 7 операцій 3 типу. Тобто такий розв'язок виконує не більше 14 операцій.

Блок 3

Покажемо, що завжди можливо використати не більше 3 операцій. Розглянемо масив $s_0 = 0$, $s_i = a_1 + a_2 + \ldots + a_i$.

Тоді якщо ми зробимо операцію другого типу на відрізку $[l..r]$, то масив стане рівним $[a_1, a_2, \ldots, a_{l-1}, s_l - s_{l-1}, s_{l+1} - s_{l-1}, \ldots, s_r - s_{l-1}, a_{r+1}, \ldots, a_n]$. А якщо зробимо операцію третього типу, то масив стане рівним $[a_1, a_2, \ldots, a_{l-1}, s_r - s_{l-1}, s_r - s_l, \ldots, s_r - s_{r-2}, s_r - s_{r-1}, a_{r+1}, \ldots, a_n]$.

Розглянемо тепер s_x — мінімальне значення масиву s та s_y — максимальне значення масиву s. Якщо $x > y$, виконаємо операцію першого типу, тобто домножимо усі значення a на -1, відповідно усі значення s також домножаться на -1, а мінімальне та максимальне значення поміняються місцями.

Тепер припустимо ми виконаємо операцію другого типу на відрізку $[x+1..n]$. Усі значення на цьому проміжку стануть рівними $a'_i = s_i - s_x \geqslant 0$. А також $a'_i = s_i - s_x = s_{i-1} + a_i - s_x = a_i + (s_{i-1} - s_x) \geqslant a_i$.

Аналогічно, якщо ми виконаємо операцію третього типу на відрізку $[1..y]$. Усі значення на цьому проміжку стануть рівними $a'_i = s_y - s_i \geqslant 0$.

Тому зробимо спочатку операцію другого типу на проміжку $[x+1..n]$, а потім операцію третього типу на відрізку $[1..y]$. Якщо б ми робили ці операції окремо, то усі значення стали б невід'ємними. Але після першої з цих двох операцій всі елементи стали не меншими, тому ми можемо виконати їх в даному порядку і усі значення стануть невід'ємними.

Блок 4

Поєднаємо розв'язок блоку 3 з розв'язком блоку 1.

Блок 5

Ми вміємо розв'язувати за одну операцію. А також точно існує розв'язок за 3 операції, який робить операцію 3 типу. Тому нам потрібно визначити, чи можливо зробити усі елементи невід'ємними за 2 операції 2 типу.

Ми вже знаємо, що другу операцію вигідно виконувати на всьому масиві. Нехай ми виконуємо першу операцію на проміжку $[l..r]$. Порахуємо для кожного r, яке повинне бути мінімальне значення a'_r після першої операції так, щоб усі значення на проміжку $[r+1..n]$ стали невід'ємними після другої операції.

Тепер переберемо l, після цього будемо перебирати r від l до n та підтримувати значення після першої та після другої операції. Відповідно ми можемо визначити, чи можливо розв'язати задачу за 2 операції другого типу.

Складність такого розв'язку $O(n^2)$.

Блок 6

Нехай ми виконуємо першу операцію на відрізку $[l..r]$. Тоді усі значення a_1, $a_1 + a_2$, ..., $a_1 + a_2 + \ldots + a_{l-1}$ повинні бути невід'ємними, а отже, якщо ми зробимо першу операцію з $l = 1$, то усі значення стануть не меншими. А отже ми можемо перебирати лише r.

Складність такого розв'язку $O(n)$.

Блок 7

Тепер потрібно визначити, чи можна розв'язати за 2 операції використовуючи операції не лише другого типу. Якщо ми виконуємо операцію першого типу, то операція другого чи третього типу повинна бути на всьому масиві. Переберемо ці варіанти.

Розв'язок для двох операцій третього типу аналогічний розв'язку з двома операціями другого типу.

Тепер розглянемо варіант, коли ми виконуємо спершу операцію третього типу, потім операцію другого типу. В іншому порядку розв'язок аналогічний.

Нехай ми робимо операцію 3 типу на проміжку $[l..r]$. Переберемо r, після цього переберемо l, рахуючи значення схожі до попередніх блоків. Складність такого розв'язку $O(n^2)$.

Блок 8

Будемо розв'язувати методом розділяй і володарюй. Нехай зараз ми розв'язуємо задачу, коли $L \leqslant l \leqslant r \leqslant R$. Нехай $\frac{M=L+R}{2}$.

Якщо $r \leqslant M$, то ми можемо перейти в меншу підзадачу L, M. Але також усі значення від $M+1$ повинні стати невід'ємними після другої операції. Для цього порахуємо, яке мінімальне значення повинне бути на позиції M після першої операції, щоб далі все було добре.

Аналогічно, якщо $l > M$, то ми можемо перейти в меншу підзадачу $M+1, R$. Але також усі значення до M повинні стати невід'ємними після другої операції. Для цього порахуємо, чи це дійсно станеться, а також яке значення буде на позиції M після другої операції.

Тепер, якщо $l \leqslant M < r$.

Переберемо r. Порахуємо значення $y1_r$ — яке буде значення на позиції $M+1$ після першої операції та $y2_r$ — мінімальне значення, яке повинне бути на позиції M після другої операції так, щоб справа всі значення стали невід'ємними. Щоб порахувати значення $y2_r$ необхідно виписати, як змінюються значення при даних операціях. Після чого потрібно виконати запити двох типів: додати пряму з заданими параметрами та порахувати максимальне значення в точці. Ці запити можна робити підтримуючи опуклу оболонку прямих.

Переберемо l. Порахуємо значення $x1_l$ — мінімальне значення на позиції $M+1$ після першої операції, щоб зліва усі значення стали невід'ємними. Припустимо y — значення на позиції $M+1$ після першої операції, тоді порахуємо такі 2 значення $x2_l$ та $x3_l$, що значення на позиції M після другої операції рівне $x2_l \cdot y + x3_l$. Для знаходження $x1_l$ також скористаємось методом опуклих оболонок.

Тепер потрібно знайти, чи існує така пара l, r, що $y1_r \geqslant x1_l$ та $x2_l \cdot y1_r + x3_l \geqslant y2_r$.

Посортуємо всі l, r разом по значеннях $x1_l$ та $y1_r$, починаючи від найбільших. Якщо ми розглядаємо наступне значення r, то додамо пряму з параметрами $y1_r, -y2_r$. Якщо ми розглядаємо наступне значення l, то потрібно перевірити, чи існує пряма, для якої значення в точці $x2_l$ більше за $-x3_l$. Для цього також будемо підтримувати опуклу оболонку прямих.

Складність такого розв'язку $O(n \log^2 n)$.

Задача A. Масив і медіани підмасивів

Автор задачі: **Антон Тригуб**
Задачу підготував: **Роман Білий**
Розбір написав: **Андрій Филипюк**

В задачі необхідно було визначити, чи можливо розділити масив парної довжини на певну кількість підмасивів непарної довжини так, щоб їхні медіани були рівні.

Теза 1. Ми розбиваємо наш масив виключно на парну кількість підмасивів.

Доведення. Нехай ми розбили на непарну кількість підмасивів. Так як кожен підмасив має непарну довжину, маємо що сумарна довжина масиву це сума непарної кількості непарних чисел, з чого випливає, що довжина масиву — непарне число, що протирічить умові.

Блок 1

Додаткова умова: $n = 2$

В будь-якому разі, ми розбиваємо наш масив рівно на 2 підмасиви, що складаються з одного елементу. Медіана масиву з одного елементу — власне цей елемент. Відповідно, відповідь `Yes`, якщо ці два масиви рівні (тобто, перший елемент оригінального масиву рівний другому елементові того ж масиву), та `No` в іншому випадку.

Теза 2. Візьмемо довільний підмасив довжини m, нехай його медіана є рівна x, тоді, і тільки тоді, у нас виконуються одночасно дві нерівності:

- `[` кількість елементів строго *менших* за x `]` $\leqslant \frac{m-1}{2}$
- `[` кількість елементів строго *більших* за x `]` $\leqslant \frac{m-1}{2}$

Доведення. Це витікає напряму з визначення медіани.

Наслідок тези 2. Якщо у підмасиві непарної довжини m існує елемент x, який зустрічається строго більше ніж $\frac{m-1}{2}$ раз, то ме діана даного підмасиву є рівна x.

Блок 2

Додаткова умова: $1 \leqslant a_i \leqslant 2$, для $1 \leqslant i \leqslant n$.

Нехай c_1 та c_2 — кількість одиничок та двійок в масиві відповідно.

Розглянемо два випадки:

$c_1 \neq c_2$*:* Без втрати загальності, скажемо, що $c_1 > c_2$. Зауважимо, що в такому разі $c_1 - c_2 \geqslant 2$. (Інакше, тобто, якщо б $c_1 - c_2 = 1$, це означало б, що $c_1 = c_2 + 1$, тобто $n = c_1 + c_2 = 2 \cdot c_2 + 1$, що не може бути правдою, бо n — парне.) Візьмемо мінімальний префікс непарної довжини, такий, що кількість одиничок на ньому рівно на 1 більше ніж кількість двійок на ньому (очевидно, що такий завжди існує). Відновідно, з того, що $c_1 - c_2 \geqslant 2$, очевидно, що серед елементів поза цим префіксом одиничок теж буде більше, ніж двійок. Тобто, ми фактично розбили масив на два підмасиви, на кожному з яких одиничок більше ніж двійок, тобто згідно Тези 2, ми маємо, що на обох масивах медіана є рівна 1. Випадок, коли у всьому масиві двійок більше ніж одиничок доказується аналогічно. Іншими словами, відповідь завжди `Yes`.

$c_1 = c_2$*:* Ми розбиваємо завжди на парну кількість підмасивів непарної довжини (згідно Тези 1). Теж ми знаємо, що у нас внаслідок того, що лише два можливих елементи, медіана кожного підвідрізку, це той елемент, який там зустрічається строго частіше ніж інший (наслідок Тези 2). Якщо ми візьмемо будь-яке розбиття (без втрати загальності, нехай це буде 1), то це означає, що у кожному підмасиві кількість 1 > кількості 2, тобто загалом $c_1 > c_2$ (бо сума нерівностей це теж нерівність), що є протиріччям. Тобто, такого розбиття не існує, і відповідь завжди `No`.

Теза 3. Якщо розбиття існує, то воно складається з 2-х підмасивів.

Доведення.

Частина 1. Візьмемо три довільні масиви з одинаковою медіаною x. Нехай розміри масивів, це m_1, m_2, m_3 відповідно. Розглянемо медіану об'єднання цих трьох масивів, та доведемо, що вона теж рівна x. Нехай c_1, c_2, c_3 це кількість елементів, що менші за x у відповідних масивах. Тоді, з Тези 2 маємо:

- $c_1 \leqslant \frac{m_1-1}{2}$
- $c_2 \leqslant \frac{m_2-1}{2}$
- $c_3 \leqslant \frac{m_3-1}{2}$

Якщо просумуємо нерівності, то матимемо $c_1 + c_2 + c_3 \leqslant \frac{m_1+m_2+m_3-3}{2} \leqslant \frac{m_1+m_2+m_3-1}{2}$. Що значить, що кількість елементів менше x у об'єднанні є не більше половини розміру об'єднання. Аналогічні операції можна провести відносно елементів більших x. З цього відповідно випливає, що x є медіаною об'єднання.

Частина 2. Відповідно, якщо ми маємо більше 2-х підмасивів, то ми постійно можемо зменшувати їх кількість на 2 шляхом об'єднання сусідніх підмасивів. Так, як кількість підмасивів має бути парна, то ми зупинимось на 2-х підмасивах.

Блок 3

Додаткова умова: $a_i \leqslant a_{i+1}$, для $1 \leqslant i < n$.

З Тези 3 ми знаємо те, що відповідь має складатися з двох підмасивів. Розглянемо довільний елемент x. Очевидно те, що x може бути медіаною обох підмасивів, якщо вони обидва містять x. Так як у нас масив неспадний, межа між підмасивами мусить проходити так, що в першому підмасиві у нас x це буде максимальний елемент, а в другому — мінімальний. Для того, щоб максимальний/мінімальний елемент якогось набору з m елементів був медіаною (де m — непарне), необхідно щоб він зустрічався хоча б $\frac{m+1}{2}$ разів (витікає з визначення). Нехай m_1, m_2 розміри підмасивів якогось розподілу масива, та $c_{x,1}, c_{x,2}$ — кількість елементів x відповідно у першому та другому підмасиві, тоді необхідно, щоб:

- $c_{x,1} \geqslant \frac{m_1+1}{2}$
- $c_{x,2} \geqslant \frac{m_2+1}{2}$

Що, в рамках даного блоку, буде виконуватись, якщо $c_{x,1} + c_{x,2} \geqslant \frac{m_1+m_2+2}{2} > \frac{m_1+m_2}{2}$. Тобто, достатньо буде перевірити, чи існує елемент, який зустрічається більше ніж $\frac{n}{2}$ раз.

Блок 4

Додаткова умова: $1 \leqslant a_i \leqslant 3$ для $1 \leqslant i \leqslant n$ та кожне значення зустрічається в a не більше ніж $\frac{n}{2}$ разів.

Якщо у нас існує лише два унікальних значення, то блок вирішується ідентично блоку 2.

В іншому випадку, так як 1 та 3 є мінімальним та відповідно максимальним елементом, для того, щоб вони були медіанами підмасивів, вони мають зустрічатись хоча б $\frac{m}{2}$ разів (де m — розмір підмасива) там, а якщо так буде, то 1 чи 3 зустрічатимуться більше ніж $\frac{n}{2}$ разів (див. блок 3). Внаслідок чого, медіаною може бути лише число 2.

Тепер залишилось перевірити, чи таке розбиття на підмасиви існує. Спершу опишемо умову того, що число 2 є медіаною якогось підмасиву розміру m (згідно тези 2):

- [кількість 1] $\leqslant \frac{m-1}{2}$
- [кількість 3] $\leqslant \frac{m-1}{2}$

Тепер, коли ми проходимось по масиву зліва направо зберігаючи кількість 1 та 3 на певному префіксі, ми можемо бистро оприділити чи цей префікс є коректним підмасивом з медіаною 2. Залишилось перевірити чи суфікс елементів, що залишаються, теж є коректним підмасивом з медіаною 2. Для цього можемо перед початком алгоритму порахувати сумарну кількість 1 та 3, тоді на певному етапі кількість 1 та 3 на суфіксі елементів, що залишаються, можна просто порахувати як [кількість загальна] - [кількість на префіксі]. Таким чином ми зможемо теж бистро перевіряти чи суфікс елементів теж є підмасивом з медіаною 2. Під час реалізації, варто не забути про те, що нам підходять лише префікси непарного розміру.

Часова складність рішення: $O(n)$.

Блок 5

Додаткова умова: $n \leqslant 100$.

Просто перебираємо можливу медіану серед елементів масиву, та перевіряємо її за допомогою техніки динамічного програмування: нехай ми зафіксували якесь x — спільна медіана, тоді dp_i це значення `Так`/`Ні`, яке означає те, чи ми можемо розбити префікс масива довжиною i на будь-яку кількість підмасивів непарного розміру, так що у них всіх медіана є рівна x. Тоді, очевидно, що dp_i = `Так`, якщо i — непарне, а також:

- Медіана префіксу розміру i є рівна x, або
- Існує таке j, що dp_j = `Так` і підмасив від j до i елементу має медіану x.

Алгоритм швидкої перевірки використовується такий самий, як у блоці 4, тільки ми трактуємо:

- Числа менші за x, як 1,
- Числа рівні x, як 2,
- Числа більші за x, як 3.

Часова складність рішення: $O(n^3)$.

Теза 4. Спільна медіана повинна бути така сама, як медіана усього масиву.

Доведення. Аналогічне доведенню Тези 3, Частини 1.

Блок 6

Додаткова умова: $n \leqslant 2000$.

Рішення 1

Помітимо, що коли ми зафіксували якесь число x і хочемо перевірити чи воно може бути спільною медіаною, у нас буквально перед очима Підзадача 4, лише трактуємо:

- Числа менші за x, як 1,
- Числа рівні x, як 2,
- Числа більші за x, як 3.

Часова складність рішення: $O(n^2)$.

Рішення 2

Використаємо Тезу 4, зафіксуємо медіану та використаємо підхід динамічного програмування, аналогічний Підзадачі 5.

Часова складність рішення: $O(n^2)$.

Повне рішення.

Використаємо Тезу 4, зафіксуємо медіану (нехай це буде x), тоді ми маємо задачу, аналогічну до Підзадачі 4, лише трактуємо:

- Числа менші за x, як 1,
- Числа рівні x, як 2,
- Числа більші за x, як 3.

Задача B. Масив символів і майже паліндроми

Автор задачі: **Антон Тригуб**
Задачу підготували: **Владислав Денисюк і Владислав Заводник**
Розбір написав: **Владислав Денисюк**

Блок 1

Перше спостереження — це те, що рядок є майже паліндромом тоді і тільки тоді, коли кількість символів, що зустрічаються непарну кількість разів у цьому рядку не більша 1. Якщо є тільки один такий символ - то ми можемо поставити його в центр рядку і зробити ліву і праву половину зеркальними, якщо таких символів 0 - то просто зробити зеркальні ліву і праву половини. Можна показати, що за допомогою цих двох варіантів можна сформувати який-завгодно паліндром, а отже якщо таких символів більше 1 — то це не майже паліндром. Тоді у данному блоці є чотири випадки: 1) кількість усіх символів парна, 2) кількість `a` — непарна, 3) кількість `b` — непарна, 4) кількість усіх символів - непарна. У випадку 4 - відповідь - весь рядок, у випадках 1,2,3 - потрібно видалити якийсь префікс і якийсь суфікс, щоб кількість `a` і `b` стала непарною, для цього переберемо усі пари префіксів і суфіксів не довших за 4, якщо префікс або суфікс довший за 4, то при видаленні перших чотирьох його символів - парність кожного символу в ньому - не зміниться.

Блок 2

Блок створено для дуже неоптимальних реалізацій підтримки парності входжень конкреного символу в блоках `3` і `4`.

Блок 3

Будемо розв'язувати задачу подібно до блоку `1`, але переберемо усі пари початку і кінця, і для кожної перевіримо кількість символів з непарною кількістю входжень.

Блок 4

Оптимізуємо рішення блоку `2`: перебиратимемо початок, і потім будемо одночасно рухати кінець вправо і додавати новий символ перераховуючи кількість непарних. Ми можемо робити це підтримуючи кількість кожного символа.

Блок 5

Порахуємо на префіксі і на суфіксі найближчу позицію де на префіксі або суфіксі відповідно парність кількості `a` дорівнює x, а `b` — y. Всього є 4 пари (x, y), бо кожне з них приймає лише два значення. Будемо перебирати комбінації парностей на префіксі і суффіксі і так порахуємо відповідь. Складність: $O(n + q)$.

Блок 6

Якщо рядок не є майже паліндромом — виведемо всю його довжину. Інакше — ми можемо видалити або лівий або правий або лівий і правий кінці і рядок перестане бути майже паліндромом. Це так тому, що якщо обидва з них входять парну кількість раз - то після видалення - обидва будуть входити непарну кількість раз і це не буде майже паліндромом. А якщо хоча б одне з них входить непарну кількість раз — то потрібно видалити те, що входить парну кількість раз і ми знову матимемо два символи які входять непарну кількість раз.

Блок 7

Розв'язок схожий до блоку `6`. Якщо символи на кінцях відрізку рівні, то при видалені символу з одного з кінців вони не будуть рівні і ми отримаємо блок `6`. Тому якщо у блоці `6` ми перебирали усі суфікси і префікси не довше одного, то тут — не довше за 2.

Блок 9

Оскільки довжина рядку непарна - один з символів точно входить в нього непарну кількість разів. Нам потрібно видалити хоча б один інший символ і парну кількість данного. Для цього знайдемо найближчий до одного з кінців відрізка символ, що зустрічається парну кількість раз. Це кінець відрізка якщо хоча б один з них не є тим символом, що зустрічається непарну кількість раз, або якщо обидва кінця відрізка — це непарний символ, ми можемо до обробки всіх запитів порахувати на префіксі і на суфіксі останню позицію де стояв інший символ, тоді ми зможемо знайти найближчий до кінців відрізка парний символ за дві перевірки. Якщо для того щоб видалити його потрібно буде видалити непарний символ непарну кількість раз, просто видалимо його ще раз з іншого кінця відрізка. Складність: $O(n + q)$.

Блок 10

По черзі видалимо початок і кінець звівши задачу до блоку 9 і візьмемо максимальну відповідь.

Задача C. Масив і знову додавання

Автор задачі: **Valerio Stancanelli**
Задачу підготував: **Tommaso Dossi**
Розбір написав: **Valerio Stancanelli**

Блок 1

$n \leqslant 100$. Потрібно заповнити позиції $[2, 100]$ одиницями, потім n разів додати $a_2 = 1$ до a_1.

Кількість операцій: $98 + n \leqslant 198 \leqslant 300$.

Блок 2

$n = k^2, k \leqslant 100$. Потрібно заповнити позиції $[3, 100]$ одиницями, потім k разів додати $a_3 = 1$ до a_2, потім k разів додати $a_2 = k$ до a_1.

Кількість операцій: $97 + k + k \leqslant 297 \leqslant 300$.

Блок 3

$n = 2^k - 1$. Спочатку заповнимо позиції $[1, 100]$ одиницями. Мета: $a_1 = 2^k - 1$.

$>$ Спостереження: $2^k - 1 = 2(2^{k-1} - 1) + 1$.

$>$ Нова мета: $a_i = 2^{k-i+1} - 1$.

Отже, коли ви отримаєте правильне значення a_i, ви можете додати a_i двічі до a_{i-1}, щоб отримати правильне значення a_{i-1}.

Приклад послідовності операцій для $n = 31$:

```
 1   1   1   1   1   1   1   ...
 1   1   1   3   1   1   1   ...
 1   1   7   3   1   1   1   ...
 1  15   7   3   1   1   1   ...
31  15   7   3   1   1   1   ...
```

Кількість операцій: $99 + 2k \leqslant 99 + 2 \cdot 59 = 217 \leqslant 300$.

Блок 4

n — число Фібоначчі. Використовуючи основну властивість чисел Фібоначчі $f_{i+2} = f_{i+1} + f_i$, можемо перетворити послідовність $\ldots, 0, f_{i+1}, f_i, \ldots$ на $\ldots, f_{i+3}, f_{i+2}, f_i$ за 3 операції:

```
...          0  f[i + 1]  f[i]  ...
...  f[i + 1]  f[i + 1]  f[i]  ...
...  f[i + 1]  f[i + 2]  f[i]  ...
...  f[i + 3]  f[i + 2]  f[i]  ...
```

Використовуючи дані 3 операції необхідну кількість разів, можемо отримати будь-яке число Фібоначчі.

Приклад послідовності операцій для $n = 55$:

```
 0   0   0   0   1   1   ...
 0   0   0   1   1   1   ...
 0   0   0   1   2   1   ...
 0   0   0   3   2   1   ...
 0   0   3   3   2   1   ...
 0   0   3   5   2   1   ...
 0   0   8   5   2   1   ...
 0   8   8   5   2   1   ...
 0   8  13   5   2   1   ...
 0  21  13   5   2   1   ...
21  21  13   5   2   1   ...
21  34  13   5   2   1   ...
55  21  13   5   2   1   ...
```

Кількість операцій: менше $99 + \frac{3}{2} \cdot \log_{\varphi}(10^{18}) \leqslant 300$.

Блок 5, 1927 операцій

Представимо n у двійковому записі. Знайдемо таку послідовність операцій, під час якої a_2 буде приймати значення $1, 2, \ldots, 2^{59}$ в певні моменти часи.

Заповнимо позиції $[2, 100]$ одиницями. Тепер для кожного j від 1 до 59, виконаємо операції $(j+1), j, \ldots, 2$, щоб виконувалися рівності $a_{j+1} = 2^1, a_j = 2^2, \ldots, a_2 = 2^j$.

Якщо двійковий запис числа n містить i-й біт, то виконаємо операцію `a[1] += a[2]`, коли $a_2 = 2^i$.

Приклад послідовності операцій для $n = 42 = 2 + 8 + 32$:

```
 0   1   1   1   1   1   1   ...
 2   2   1   1   1   1   1   ...
 2   4   2   1   1   1   1   ...
10   8   4   2   1   1   1   ...
10  16   8   4   2   1   1   ...
42  32  16   8   4   2   1   ...
```

Кількість операцій: $98 + \frac{59 \cdot 60}{2} + 59 = 1927$.

Блок 5, 1086 операцій

У попередньому алгоритмі отримання великих степеней двійки потребує великої кількості операцій. Виявляється, що є кращий спосіб створення степенів 2. У другій половині процесу ми можемо створювати степені 4 замість того, щоб створювати степені 2.

Приклад послідовності операцій для $n = 426 = 2 + 8 + 32 + 128 + 256$:

```
  0   1   1   1   1   1   1   ...
  2   2   1   1   1   1   1   ...
  2   4   2   1   1   1   1   ...
 10   8   4   2   1   1   1   ...
 10  16   8   4   2   1   1   ...

 42  32  16   8   4   1   1   ...
 42  64  32  16   4   1   1   ...
170 128  64  16   4   1   1   ...
426 256  64  16   4   1   1   ...
```

Зауважте, що нам потрібна одна додаткова операція для кожного рядка у другій половині процесу (оскільки нам потрібно додати 4^k двічі до $2 \cdot 4^k$, щоб отримати 4^{k+1}).

Кількість операцій: $98 + \frac{30 \cdot 31}{2} + \frac{29 \cdot 30}{2} + 29 + 59 = 1086$.

Блок 5, 929 операцій

Краще використовувати систему числення з основою 3.

Цього разу нам потрібні 2 операції для того, щоб отримати 3^{k+1}, починаючи з 3^k. Також необхідно виконати 4 додаткові операції для кожного рядка у другій половині процесу (оскільки нам потрібно додати 9^k шість разів до $3 \cdot 9^k$, щоб отримати 9^{k+1}). Проте система числення з основою 3 є більш ефективною, ніж система числення з основою 2, оскільки $log_3(10^{18}) \approx 38$ є «набагато» меншим за $log_2(10^{18}) \approx 60$.

Кількість операцій: $98 + 2 \cdot \frac{19 \cdot 20}{2} + 2 \cdot \frac{18 \cdot 19}{2} + 4 \cdot 18 + 37 = 929$.

Блок 5, 276 операцій

Попередні рішення використовують $O(\log^2 n)$ операцій. Чи можемо ми використати $O(\log n)$ операцій?

Ми знаємо, як отримати значення $2x$ починаючи з x:

```
 0   x   ?   ?   ?   ?   ?   ...
 x   x   ?   ?   ?   ?   ?   ...
2x   x   ?   ?   ?   ?   ?   ...
```

Ми також хочемо мати можливість отримати $(2x+1)$ починаючи з x. Одним з можливих способів є додавання 1 до всіх позицій спочатку:

```
      1   1   1   1   1   1   1   ...
      1   x   ?   ?   ?   ?   ?   ...
  x + 1   x   ?   ?   ?   ?   ?   ...
 2x + 1   x   ?   ?   ?   ?   ?   ...
```

Але тепер ми не можемо отримати $2x$ з x. Виявляється, що замість цього ми можемо отримати $2x + 2$.

```
      1   1   1   1   1   1   1   ...
      2   1   1   1   1   1   1   ...
      2   x   ?   ?   ?   ?   ?   ...
  x + 2   x   ?   ?   ?   ?   ?   ...
 2x + 2   x   ?   ?   ?   ?   ?   ...
```

Цього достатньо, щоб обчислити цільові значення a_i:

1. $a_1 = n$;
2. $a_i = \frac{a_{i-1}-1}{2}$ якщо a_{i-1} непарне;
3. $a_i = \frac{a_{i-1}-2}{2}$ якщо a_{i-1} парне.

Приклад послідовності операцій у зворотньому порядку для $n = 150$:

```
150  74  36  17   8   3   1   ...
  2  74  36  17   8   3   1   ...
  2   2  36  17   8   3   1   ...
  2   2   2  17   8   3   1   ...
  2   2   2   1   8   3   1   ...
  2   2   2   1   2   3   1   ...
  2   2   2   1   2   1   1   ...
  1   1   1   1   1   1   1   ...
```

Таким чином, нам просто потрібно 99 операцій, щоб заповнити масив одиницями, 59 операцій, щоб поставити 2 там, де це необхідно, а потім $2 \cdot 59$ операцій, щоб отримати цільові значення a_i.

Кількість операцій: $99 + 59 + 2 \cdot 59 = 276 \leqslant 300$.

Задача D. Масив і XOR

Автор задачі: **Денисов Костянтин**
Задачу підготував: **Денисов Костянтин**
Розбір написав: **Денисов Костянтин**

Блок 1

Напишемо повний перебір. Порахуємо $f_i(x)$ для кожного x від 0 до 2^m-1 проходимо по елементам масиву з відповідного відрізка запиту.

Блок 2

Будемо поступово будувати відповідь на запит від більших бітів до менших. Дізнаємося чи буде $m-1$ біт у відповіді. Поділимо наш вхідний массив a на 2 інші масиви b_0 та b_1, де у b_0 знаходяться всі числа вхідного масиву, у яких $m-1$ біт дорівнює 0, аналогічно b_1 містить всі числа вхідного масиву, у яких $m-1$ біт дорівнює 1. Тоді очевидно, що якщо хоча б один з масивів b_0, b_1 є порожнім, то $m-1$ біт буде дорівнювати 1 у відповіді. Розглянемо випадок, коли обидва масива непорожні. Розглянемо x, у яких $m-1$ біт дорівнює 0, тоді $f(x) = \min_{y\in a} y \oplus x = \min_{y\in b_0} y \oplus x$, оскільки для чисел з масиву b_1 число $y \oplus x$ буде мати $m-1$ біт, отже, на ньому мінімум не буде досягатися. Аналогічно, якщо розглянути x, у яких $m-1$ біт дорівнює 1, то отримаємо $f(x) = \min_{y\in a} y \oplus x = \min_{y\in b_1} y \oplus x$. Отже, фактично ми розділили задачу на дві незалежні задачі (в залежності від $m-1$ біта в x), в яких відрізняються масиви на яких шукаємо відповідь.

З цих міркувань маємо наступний алгоритм (псевдокод), що поступово знаходить відповідь.

```
solve (m, a) {
    if (m == 0) return 0
    build b[0], b[1] from a
    if (b[0] is empty)
        return solve(m - 1, b[1]) ^ (1 << (m - 1));
    if (b[1] is empty)
        return solve(m - 1, b[0]) ^ (1 << (m - 1));
    return max(solve(m - 1, b[0]), solve(m - 1, b[1]));
}
```

Блок 4

Помітимо, що рекурсивне розділення масиву на b_0 та b_1 аналогічне побудові двійково бору на числах масиву. У вершинах бору можемо зберігати кількість чисел, що проходить через поточну вершину бора та підтримувати відповідь для цього піддерева бора. Отже, знаючи значення у синах вершини можемо просто порахувати значення у батьківській вершині бора. Таким чином ми можемо додадавати та видаляти числа з масиву при цьому оновлюючи всі значення у вершинах за $O(m)$. Таким чином можемо використати алгоритм Мо. Складність рішення $O(n \cdot \sqrt{q} \cdot m)$.

Блок 3

Побудуємо бор на всьому вхідному масиві. Навчимося відповідати на запит на відрізку $[l, r]$. Для цього будемо ітеруватися вглиб бора поки всі числа з поточного відрізку проходять через поточну вершину бора. Спочатку ми знаходимся у корені бора. Далі розглянемо два варіанти:

1. Якщо всі числа з поточного відрізка запиту проходять через один з синів поточної вершини, то перейдемо до цього сина і продовжимо цю ж процедуру.

2. В іншому випадку числа з поточного відрізка знаходяться у двох синах вершини. Можна помітити, що якщо подивитися на взагалі всі числа масиву, що проходять через якогось сина поточної вершини, то числа з відрізка запиту будуть утоворювати якийсь його префікс або суфікс. Тоді можна зберегти відповіді для кожного префікса (суфікса) у кожній вершині бору поки його будуємо.

Блок 5

Цей блок було створено для неоптимальних рішень, які використовують дерево відрізків або дерева фенвіка у заключній частині рішення з блоку 6.

Блок 6

Можемо помітити, що, неформально некажучи, всі попередні алгоритми обирають якийсь шлях у борі і «максимізують» значення на цьому шляху. А саме, якщо ми пішли на шляху до якогось сина вершини у борі, якій відповідає k-ий біт, то значення на цьому шляху буде мати k-ий біт рівний 1 тоді і тільки тоді, коли немає другого сина у поточній вершині.

Тоді оберемо якийсь шлях у нашому борі, розглянемо як інші шляхи впливають на його значення. Оберемо якийсь інший шлях. Нехай вони перетинаються по k найбільших бітах. Тоді значення на першому шляху буде мати $k+1$ найбільший біт рівний 0. Отже, нас цікавить тільки довжина перетину двох шляхів. Є $m+1$ різних довжин перетину. Шлях відповідає певному числа з масива a. Нехай воно має індекс j. Тоді для кожної можливої довжини len перетину шляхів знайдемо найбільший індекс l_len, що $l_len < j$ та шлях числа a_{l_len} має довжину перетину зі шляхом a_j рівну len. Аналогічно знайдемо такі найменші значення r_len, що $r_len > j$. Інші шляхи нас не цікавлять, бо вони вже ніяк не змінять значення на шляху для числа a_j.

Тоді для кожного шляху, що відповідає числу a_j маємо не більше m шляхів зліва, що йому змінять значення на цьому шляху і аналогічно не більше m шляхів справа. Тоді маємо, що є не більше m^2 різних відрізків, де поточний шлях приймає різні значення. Тоді кожного шляху будемо мати не більше m^2 таких «оновлень»: $[c_i, d_i, b_i]$, що означає, що значення цього шляху на відрізку $[c_i, d_i]$ дорівнює b_i. Іншими словами, якщо в нає є запит, який міститься у відрізку $[c_i, d_i]$, то в нього буде значення принаймі b_i.

Тобто зараз ми маємо таку задачу: маємо не більще $n \cdot m^2$ $[c_i, d_i]$ відрізків кожен зі значенням b_i і для кожного запиту нам потрібно дізнатися максимальне значення серед відрізків, які повністю його містять.

Цю задачу будемо вирішувати в офлайні, тобто спершу зчитаємо всі запити і будемо відповідати на них у довільному порядку, а не в тому, що їх задають.

Заведемо новий масив e довжиною n, початково $e_i = 0$. Відсортуємо наші запити $[l_i, r_i]$ за зростанням l_i. Далі розглядаємо наші запити у порядку збільшення l_i. Розглянемо запит $[l_i, r_i]$. Тоді для всіх відрізків j, що $c_j \leqslant l_i$ оновимо $e_{d_j} := max(e_{d_j}, b_j)$. Тоді відповіддю на запит $[l, r]$ буде максимальне значення на відрізку $[r_i, n]$. Оскільки запитів оновлення значно більше ніж запитів знаходження максимуму, то будемо підтримувати це за допомогою sqrt-декомпозиції.

Отже, маємо рішення за $O(n \cdot m^2 + q \cdot \sqrt{n})$.