UOI 2026 Розбір

Тур 1

8 травня 2026 р.

Зміст

Унікальні літери — Умова

Задача

Дано рядки s та t. **Унікалізація** рядка x: йдемо зліва направо, підтримуючи рядок a.

- Якщо $c \notin a$: дописати c в кінець a.
- Якщо $c \in a$: видалити з a суфікс до першого входження c *включно* (саме c не додається).

Знайти такі перестановки s' і t', що результат унікалізації s' дорівнює t', або вивести `NO`.

Обмеження: $1 \le |s|, |t| \le 2 \cdot 10^5$

Унікальні літери — Приклад

$s' =$ `bcdbef`, $t' =$ `ef`. Покрокова унікалізація:

крок	символ	дія	a
1	`b`	дописати	`b`
2	`c`	дописати	`bc`
3	`d`	дописати	`bcd`
4	`b`	**видалити суфікс**	(порожній)
5	`e`	дописати	`e`
6	`f`	дописати	`ef`

Кінцевий $a =$ `ef` $= t'$. **Відповідь:** YES.

Унікальні літери — Дві миттєві перевірки на NO

Спостереження

Через те, що a ніколи не містить двох однакових літер, t' також має лише унікальні символи.

Перевірки

1. Якщо в t якась літера зустрічається більше ніж раз $\Rightarrow$ NO.
2. Якщо літера є в t, але її немає в $s \Rightarrow$ NO (звідки б вона з'явилась у фінальному a?).

Далі вважатимемо, що ці перевірки пройшли успішно.

Унікальні літери — Як літери “зникають”

Доля кожного входження літери c в s'

Зафіксуємо літеру c. Кожне її входження в s' або:

- **Виживає в t'** — лише одна копія, і лише якщо $c \in t$.
- **Зникає через дублікату** — зустріла свою копію в a, дві копії c зникають разом.
- **Стерта чужим дублікатом** — стояла в a, інша літера зачистила суфікс. Стирається сам елемент c.

Підрахунок

$\text{cnt}_s[c] = \text{cnt}_t[c] + 2k_c + m_c$, де k_c — скільки разів c була видалена своїм дублікатом, m_c — скільки разів її стерла чужа літера.

Висновок

$\text{cnt}_s[c] - \text{cnt}_t[c] \equiv m_c \pmod 2$. Парність “залишку” для c задає парність числа її стирань чужим числом.

Унікальні літери — Простий випадок: усі парності збігаються

Коли працює сорт

Якщо $\mathrm{cnt}_s[c] - \mathrm{cnt}_t[c]$ парне для всіх c, то $m_c = 0$ для всіх c працює: жодних чужих стирань не потрібно.

Конструкція: виводимо $s' = \mathrm{sort}(s)$ і $t' = \mathrm{sort}(t)$. Однакові літери йдуть підряд $\Rightarrow$ кожна пара самостирається.

Приклад: $s = \texttt{aaabbcc}$, $t = \texttt{a}$

Прохід: $\texttt{a} \to \texttt{a}$; $\texttt{a} \to$ (порож.); $\texttt{a} \to \texttt{a}$; $\texttt{b} \to \texttt{ab}$; $\texttt{b} \to \texttt{a}$; $\texttt{c} \to \texttt{ac}$; $\texttt{c} \to \texttt{a}$.

Кінцевий $a = \texttt{a} = t'$. ✓

Невелика тонкість

Тут літери t опиняються в кінці алфавіту в обох рядках. Якщо це не так, то парні групи літер не з t самі себе знищують, а непарні групи літер з t залишають по одній копії — сорт усе одно дає валідну конструкцію.

Унікальні літери — А якщо парності НЕ збігаються?

Що це означає

Тоді є літери з непарним $\text{cnt}_s[c] - \text{cnt}_t[c]$. Для них m_c непарне, тобто їх *мусить* стирати якась чужа літера.

Як побудувати чуже стирання

Беремо літеру X: ставимо одну копію в a, потім ставимо суфікс "непарних" літер, потім другу копію X — це дублікат, що зачищає все, включно зі "вкладеними" непарними літерами.

Хто може бути X

Літера має мати **запас** ≥ 2: $\text{cnt}_s[X] - \text{cnt}_t[X] \geq 2$. Список таких літер позначимо have.

Якщо have $= \emptyset$, *а непарні літери є* $\Rightarrow$ `NO`.

Унікальні літери — Один цикл стирання

Конструкція префіксу

Беремо літеру $i \in$ have з *парним* $\text{cnt}_s[i] - \text{cnt}_t[i]$ та будуємо префікс

$$\underbrace{i}_{\text{1-ше } i} \quad \underbrace{[\text{непарні літери, крім } i]}_{\text{вкладений суфікс}} \quad \underbrace{i}_{\text{дублікат}}$$

Покроково

1. $a \leftarrow i$.
2. Для кожного $j \neq i$ з непарним $\text{cnt}_s[j] - \text{cnt}_t[j]$: додати одну копію j, $\text{cnt}_s[j] \mathrel{-}= 1$. Тепер $a = ij_1j_2\ldots$
3. Друге i — дублікат: стирає все від першої i. a порожній. $\text{cnt}_s[i] \mathrel{-}= 2$.
4. Решту літер дописуємо у відсортованому порядку.

Чому це працює

Кожна непарна літера зникла рівно один раз чужим дублікатом ($m_j = 1$) — парність виправлено. Для i ми витратили 2 копії — її парність не змінилась (а вона й була парною). Решта парні $\Rightarrow$ сорт спрацює.

Унікальні літери — А якщо have містить одну літеру з непарним залишком?

Глухий кут

Тоді сама i потребує виправлення парності, але один цикл $i\,[\ldots]\,i$ витрачає 2 копії i парою — це *не* змінює парність i.

Парність завжди непарна $\Rightarrow$ `NO`.

Що рятує ситуацію

Якщо ж у have є друга літера pos, ми можемо побудувати **два цикли**: перший виправляє непарності всіх літер, крім i; другий виправляє саму i.

Унікальні літери — Два цикли стирання

Конструкція префіксу

Нехай $i, \text{pos} \in \text{have}$.

$$\underbrace{i\ [\text{непарні крім } i]\ i}_{\text{цикл 1}} \quad \underbrace{\text{pos}\ [\text{можливо } i]\ \text{pos}}_{\text{цикл 2}} \quad \underbrace{[\text{решта відсортовано}]}_{= t'}$$

Що відбувається

1. Цикл 1 виправляє парність кожної непарної літери крім i ($m = 1$ для них).
2. Якщо після циклу 1 літера i досі непарна — встромляємо її між двома pos, її m_i збільшується на 1, парність виправлено.
3. Цикл 2 витрачає 2 копії pos парою — парність pos не змінюється.

Результат

Після обох циклів a порожній і всі залишки парні $\Rightarrow$ дописуємо решту відсортовано $\Rightarrow$ кінцевий $a = t'$.

Унікальні літери — Підсумкова таблиця рішень

Усі випадки

Стан	Дія
$\text{cnt}_t[c] > 1$	`NO`
$\text{cnt}_t[c] > \text{cnt}_s[c]$	`NO`
Усі $\text{cnt}_s + \text{cnt}_t$ парні	`YES`, sort(s) та sort(t)
Є непарні, have $= \emptyset$	`NO`
$\lvert\text{have}\rvert = 1$, єдина i парна	`YES`, один цикл стирання
$\lvert\text{have}\rvert = 1$, єдина i непарна	`NO`
$\lvert\text{have}\rvert \geq 2$	`YES`, два цикли стирання

Складність

$O(|s| + |t| + 26)$ — лінійні перевірки та однопрохідна побудова.

Зміст

Перетин префіксних сум — Умова

Задача

Дано масив a та число x. Треба переставити елементи так, щоб жодна непорожня префіксна сума не дорівнювала x:

$$p_1 + p_2 + \cdots + p_k \neq x$$

для всіх $1 \leq k \leq n$.

Позначення

$$S = a_1 + a_2 + \cdots + a_n.$$

Якщо $S = x$, відповідь одразу `NO`, бо весь масив теж є префіксом.

Перетин префіксних сум — Нормалізація

Можемо вважати, що $x \geq 0$

Якщо $x < 0$, помножимо x і всі числа масиву на -1.

Кожна префіксна сума s перетвориться на $-s$, тому задача не зміниться.

Далі

Після цього розглядаємо тільки випадок:

$$x \geq 0.$$

Перетин префіксних сум — Випадок $x = 0$

Ідея

Випадок $S = x$ уже оброблено, тому $S \neq 0$.

Побудова

Якщо $S > 0$, розташуємо числа у спадному порядку.
Тоді всі префіксні суми будуть додатними.

Якщо $S < 0$, розташуємо числа у зростаючому порядку.
Тоді всі префіксні суми будуть від'ємними.

Висновок

Префіксної суми, рівної 0, не буде.

Перетин префіксних сум — Випадок $S < x$

Побудова

Сортуємо масив за зростанням: спочатку йдуть недодатні числа, потім додатні.

Чому це працює?

Спочатку префіксні суми не більші за 0.

Потім вони починають зростати, але фінальна сума дорівнює S, а

$$S < x.$$

Отже, жоден префікс не може дорівнювати x.

Перетин префіксних сум — Додатні числа

Залишився основний випадок

Далі маємо:

$$x > 0, \qquad S > x.$$

Розглянемо додатні числа

Нехай

$$c_{\min} = \min(a_i > 0), \qquad c_{\max} = \max(a_i).$$

Якщо $c_{\min} \neq c_{\max}$, то додатні числа не всі однакові.

Ідея

Коли додатні числа не всі однакові, можна змінити порядок так, щоб “перестрибнути” через x.

Перетин префіксних сум — Додатні не всі однакові

Побудова

Беремо всі додатні числа першими у відсортованому порядку.

Їхні префіксні суми строго зростають, тому серед них може бути не більше одного префікса із сумою x.

Якщо такий префікс є

Нехай проблемний елемент стоїть на позиції i.

Міняємо його з іншим додатним числом так, щоб сума цього префікса змінилася.

Оскільки додатні числа не всі однакові, такий обмін можливий.

Після цього

У кінець додаємо всі недодатні числа.
Поточна сума вже більша за x, а фінальна сума теж більша за x, тому до x ми не опустимося.

Перетин префіксних сум — Усі додатні однакові

Позначення

Нехай усі додатні числа рівні c.

Ключове спостереження

Кожне збільшення префіксної суми відбувається тільки кроком $+c$.

Якщо

$$x \not\equiv 0 \pmod{c},$$

то сума, кратна c, не може дорівнювати x.

Висновок

Якщо x не ділиться на c, відповідь `YES`.

Перетин префіксних сум — Зсув сум

Ще один спосіб отримати YES

Нехай усі додатні числа рівні c, і x ділиться на c.

Якщо існує недодатне число b, яке не ділиться на c, ставимо його першим.

Чому це працює?

Після першого елемента всі наступні префіксні суми матимуть залишок

$$b \bmod c.$$

Цей залишок не дорівнює 0, а x ділиться на c.

Тому такі префіксні суми не можуть дорівнювати x.

Перетин префіксних сум — Єдиний поганий випадок

Відповідь NO

Відповідь неможлива, якщо $S = x$.

Також відповідь неможлива, якщо одночасно:

$$S > x,$$

усі додатні числа рівні c,

$$x \equiv 0 \pmod{c},$$

$$a_i \equiv 0 \pmod{c} \quad \text{для всіх } a_i \leq 0.$$

Чому?

Усі префіксні суми кратні c, а вгору можна рухатися тільки кроком $+c$. Тому неможливо перейти з суми меншої за x до суми більшої за x, не потрапивши рівно в x.

Перетин префіксних сум — Алгоритм

Алгоритм

1. Якщо $x < 0$, множимо x і всі a_i на -1.
2. Сортуємо масив.
3. Якщо $S = x$, виводимо `NO`.
4. Якщо $x = 0$, будуємо порядок за знаком S.
5. Якщо $S < x$, виводимо масив у зростаючому порядку.
6. Інакше перевіряємо додатні числа.
7. Якщо вони не всі однакові, виправляємо можливий поганий префікс обміном.
8. Якщо всі додатні рівні c, перевіряємо кратність x та недодатних чисел.

Складність

$$O(n \log n)$$

через сортування. Пам'ять:

$$O(n).$$

Зміст

Відгадай число — Умова

Задача (інтерактивна)

Загадано ціле число n ($1 \le n < 10^8$), яке не ділиться на 10.

Можна задавати запити: вибрати x ($1 \le x \le 10^9$) — отримаєш першу цифру числа $n \cdot x$.

Потрібно знайти n, використовуючи **не більше 28 запитів**.

Перша цифра числа — його найстарша цифра у десятковому записі. Наприклад: $\mathrm{fd}(7) = 7$, $\mathrm{fd}(42) = 4$, $\mathrm{fd}(123456) = 1$.

Відгадай число — Ключова ідея

Крок 1: дізнатися першу цифру n (1 запит)

Запитаємо $x = 1$. Тоді: перша цифра n = y.

Приклад

$n = 37 \Rightarrow y = 3$
$n = 11 \Rightarrow y = 1$

Відгадай число — Ключова ідея

Крок 2: знайти точку переходу x^* (27 запитів)

Бінарний пошук на відрізку $[10^8,\ 2 \cdot 10^8]$ — шукаємо **перший** x^*, де перша цифра $n \cdot x^*$ змінюється (стає $\neq y$).

Зауваження про кількість цифр n

$n < 10^8$, тобто n має **не більше 8 цифр**. Однак кількість цифр нас **не цікавить** — ми вважаємо n «числом з 8 цифр» (з можливими ведучими нулями). Множення на $x \in [10^8, 2 \cdot 10^8]$ гарантує, що перехід першої цифри завжди відбудеться у цьому відрізку — незалежно від того, скільки цифр насправді має n.

Відгадай число — Чому відрізок $[10^8, 2 \cdot 10^8]$?

Гарантія існування переходу

При $x = 10^8$: перша цифра $= y$ ✓

При $x = 2 \cdot 10^8$: $n \cdot 2 \cdot 10^8 = 2 \cdot (n \cdot 10^8)$, тобто ми подвоїли число. Подвоєння **завжди** змінює першу цифру: якби $\text{fd}(2m) = \text{fd}(m) = d$, то $2m \in [d \cdot 10^k, (d{+}1) \cdot 10^k)$ і $m \in [d \cdot 10^k, (d{+}1) \cdot 10^k)$, але тоді $2m \geq 2d \cdot 10^k \geq (d{+}1) \cdot 10^k$ — суперечність.

Бінарний пошук

- $lo = 10^8$ (перша цифра $= y$), $hi = 2 \cdot 10^8$ (перша цифра $\neq y$)
- Поки $lo + 1 < hi$: беремо $mid = (lo + hi)/2$, запитуємо
  - якщо відповідь $= y$: $lo = mid$
  - інакше: $hi = mid$
- $x^* = hi$ — перший x, де цифра змінилась

Ширина відрізку $= 10^8$, тому $\lceil \log_2 10^8 \rceil = 27$ кроків достатньо.

Відгадай число — Відновлення n

Що означає x^*?

При $x^* - 1$: $n \cdot (x^* - 1)$ починається з цифри y.
При x^*: $n \cdot x^*$ починається з цифри $y + 1$.

Отже між $n \cdot (x^* - 1)$ та $n \cdot x^*$ міститься рівно одне порогове значення вигляду $(y + 1) \cdot 10^p$:

$$n \cdot (x^* - 1) < (y + 1) \cdot 10^p \leq n \cdot x^*$$

Формула для n

Довжина відрізку для n рівна $\frac{(y+1)\cdot 10^p}{x^*(x^*-1)} \approx \frac{n}{x^*} < \frac{10^8}{10^8} = 1$

Тому у відрізку **рівно одне ціле число**:

$$\boxed{n = \left\lceil \frac{(y+1)\cdot 10^p}{x^*} \right\rceil}$$

Відгадай число — Підрахунок запитів

Кількість запитів

Крок 1: знайти y	1 запит
Крок 2: бін. пошук на $[10^8,\ 2 \cdot 10^8]$	27 запитів
Разом	**28** ✓

Зміст

mex plus — Умова

Задачка

Дано набір із n пар цілих невід'ємних чисел.
Для кожної пари (x, y) можна зробити один із двох виборів:

- додати x до масиву A, а y до масиву B;
- додати y до масиву A, а x до масиву B.

Також є q запитів. Кожен запит або додає одну пару, або видаляє одну пару. Після кожного запиту та перед усіма ними потрібно знайти максимальне можливе значення $\text{mex}(A) + \text{mex}(B)$.

Обмеження: $1 \le n \le 2 \cdot 10^5$, $0 \le q \le 2 \cdot 10^5$, $0 \le a_i \le b_i \le 10^9$, $0 \le x \le y \le 10^9$

mex plus — Представимо задачку як граф

Конструкція

Вершинками графу будуть числа. Якщо ми маємо пару (x, y), то в графі буде ребро між вершинками x та y. Ребра можуть повторюватися та можуть бути петлі. Тепер будемо орієнтовувати ребра, і якщо ребро орієнтовано від x до y, то це означає, що x буде в першому масиві, а y в другому.

Підхід до розв'язування

Помітимо, що x знаходиться в першій множині тоді і тільки тоді, коли з нього виходить принаймні одне ребро, а в другій, якщо в нього входить принаймні одне ребро. Тепер відповіддю буде сума найменшої вершинки, з якої нічого не виходить і найменшої в яку нічого не входить.

mex plus — Одна компонента

Компонента з циклом

Цикл ми завжди можемо орієнтувати, так щоб всі вершинки в ньому мали одне вхідне та одне вихідне ребро. Всі інші ребра будемо орієнтувати так щоб всі окрім листочків мали вхідні і вихідні ребра. Листочки в свою чергу будуть мати лише одне вихідне ребро. Тоді всі будуть в першій множині, і всі окрім листочків будуть в другій, тому *mex* другої множини не може бути більший за найменший листочок.

Дерево

У випадку з деревом ми можемо орієнтувати всі ребра в сторону найбільшої вершинки, тоді всі окрім найважчої вершинки будуть впершій множині. В другій будуть всі не листочки та найважча вершинка.

mex plus — П’ятий блок: $q = 0$

Відповідь

mex другої множини — це найменший листочок, оскільки ми в кожній з компонент не візьмемо його в другу множину. А *mex* першої множини — це найменший з найбільших вершинок в кожному з дерев.

Алгоритм

Щоб знайти найменший листочок, ми можемо пройтися по всіх вершинках після побудови графу і перевірити чи вони є листочками. Та запустити DFS і підрахувати суму степеней в компоненті, розмір та максимум в ній.

Часова складність: $\mathcal{O}(n \log n)$

mex plus — Шостий блок: Використаємо DSU

Шостий блок: усі запити мають вигляд + x y

Оскільки ми можемо лише додавати ребра, то можемо підтримувати наші компоненти в DSU, пам'ятати, чи вони мають в собі цикл та підтримувати максимум в тій компоненті. Щоб швидко шукати *mex* для першої і другої будемо використовувати **set**'и.

Часова складність: $\mathcal{O}(n \log n)$

mex plus — Сьомий блок: Використаємо DSU з відкатами

Сьомий блок: усі запити мають вигляд - `x y`

Оскільки початково відомі нам пари ми можемо додавати в довільному порядку, то додамо спершу всіх, кого ми не будемо видаляти, а потім в оберненому порядку будемо додавати ті що ми маємо видалити. Тоді ми видалення можемо трактувати, як відмінити останнє додавання.

Часова складність: $\mathcal{O}(n \log n)$

mex plus — Повне рішення: Використаємо DCP

Повний розв'язок

Зауважимо, що ми тепер можемо розв'язувати нашу задачку подібно до рішення Dynamic Connectivity в офлайні, оскільки ми вміємо додавати нове ребро і видаляти останнє додане. Для цього ми будемо додавати ребро на відрізок, від моменту коли ми його додали до моменту коли ми його видалили. Тепер в дерево відрізків додамо це ребро на той відрізок та зробимо обхід цього дерева відрізків.

Часова складність: $\mathcal{O}(n \log^2 n)$

Зміст

Підмножини дерева — Умова

Умова

Є дерево, підвішене за вершину 1. Кожна вершина має значення $a_i \in \{0, 1\}$. Для кожної пари (k, x), де $1 \leq k \leq n$ та $x \in \{0, 1\}$, треба з'ясувати, чи існує множина S з такими властивостями:

- 1. $|S| = k$;
- 2. $x = \oplus a_i, i \in S$, тобто ксор всіх елементів множини дорівнює x;
- 3. Якщо $v \in S$, то і всі вершини з піддерева також в S.

Обмеження: $1 \leq n \leq 4 \cdot 10^5$

Підмножини дерева — простий рюкзак

ДП і його підтримка

Нехай для кожної вершини v будемо тримати $dp[v][t]$ — відповідь для піддерева v. Нехай ми маємо вершину v тільки з 2 синами x, y, $c = dp[v], a = dp[x], b = dp[y]$. Тоді можливі значення $c[t]$ будуть $a[i] \oplus b[t-i]$, якщо $a[i], b[t-i] \in \{0, 1\}$, інакше 2. Якщо для якогось t ми маємо ≥ 2 кандидатів, то замінюємо на 2. І також в кінці додаємо $dp[v][\text{sz}(v)] = dp[v][\text{sz}(v)-1] \oplus a[v]$. Нехай $sz(x) = m \geq sz(y) = k$, це працює за $O(mk)$. Коли в нас більше ніж 2 дітей, то ми можемо робити першу частину процесу для цієї пари й об'єднувати їх в одного великого сина.

Сумарно це працює за $O(n^2)$.

Підмножини дерева — об'єднання за лінію

Швидша підтримка ДП

Нехай пройдемося по $t = 0..k + m$: одне значення для t ми можемо легко порахувати, потрібно перевірити, чи воно унікальне. В нас буде проміжок чисел $[l_1, r_1]$ в першому сині (x) та $[l_2, r_2]$ в другому сині (y), а також виконується $l_1 + r_2 = l_2 + r_1$. Ми робимо перевірку для $i \in [l_1, r_1], j \in [l_2, r_2], i + j = t$, а це можна переписати як $j = t - i$, і потрібно перевірити, чи існує таке q, що $a[l_1..r_1] = b[r_2..l_2] \oplus q$. І це можна робити хешами.

Це працює за $O(m + k)$.

Підмножини дерева — повне рішення

Перший крок

Нехай порахуємо перші k значень і останні k з попередньої ідеї, це буде $O(k)$.

Другий крок

Нам потрібно порахувати $c[k+1..m]$. Один кандидат для кожного з них — це a. Для перевірки на іншого в нас є 2 випадки:

- Легкий: в b є 2, нехай її позиція — $j \geq 1$, тоді для кожного $c[i]$ можемо взяти $b[j], a[i-j]$, $i-j \in [k+1-j, m-j]$, тому все добре.
- Складний: в b немає 2.

Підмножини дерева — повне рішення

Перший крок

Нехай порахуємо перші k значень і останні k з попередньої ідеї, це буде $O(k)$.

Складний випадок

Якщо в $dp[v]$ існує такий індекс $i \in [k+1, m]$, що $c[i] = a[i], c[i+1] = a[i+1]$. Тоді $a[i] = a[i] \oplus b[0] = c[i] = a[i-1] \oplus b[1] \Leftrightarrow b[1] = a[i] \oplus a[i-1]$, аналогічно $b[1] = a[i-1] \oplus a[i-2]$. Це означає $a[i] = a[i-2]$. І можемо по індукції показати, що $b[2i] = 0, b[2i+1] = c \in \{0, 1\}$. Нехай тоді розіб'ємо a на блоки форми $s, s \oplus c, s, s \oplus c, s, s \oplus c, \ldots$. Тоді кожен блок будемо репрезентувати як $(s_0, len_0), (s_1, len_1), \ldots$, зауважимо, що деякі s можуть бути 2.

Підмножини дерева — повне рішення

Продовження

Замітимо, що для таких b в нас тільки префікс розміру $k-1$ кожного блоку буде змінюватися, бо для $f \geq k$ в блоці в нас $a[f] = a[f-1] \oplus c = a[f-2] = a[f-3] \oplus c = \cdots = a[f-k](\oplus c)$, де останньої дужки може не бути (залежно від парності), і всі індекси всередині цього блоку, і після цього цей блок самоузгоджений. І тоді ми можемо робити це напряму, оскільки кожен раз ці суперечності збільшують кількість 2, а їх сумарно по всьому дереву може бути не більше n.

Випадок, коли не існує послідовних i, або b має іншу форму

Тоді серед $c[k+1..m]$ буде щонайменше половина 2.