

UCPC 2026 - II stage

Ukrainian Collegiate Programming Contest

24 жовтня 2026 р.

Розбір

A Архівне дерево
B Побітові вибоїни
C Зручні партії
D Дні тижня
E Екзаменаційна сесія
F Чотири сходинки
G Згруповані вершини
H Зала пісень
I Незручні обміни
J Банка фарб
K Знаєте цю задачу?
L Загублений список
M Мозаїчна арка

A Архівне дерево

Підвісимо дерево за s і запишемо $f_v = k_v \cdot f_p$ для кожної папки $v \neq s$ з батьківською папкою p. Тоді f_v — добуток множників k на шляху від s до v, а дві підпапки однієї папки мають різні назви тоді й лише тоді, коли в них різні множники, бо f_p у них спільне. Якщо помножити всі назви в піддереві на одне й те саме число, піддерево лишиться коректним, тож піддерева можна оптимізувати незалежно. Нехай $h(u)$ — найменша можлива найбільша назва в піддереві u, якщо $f_u = 1$; тоді $h(\text{листок}) = 1$, а відповідь дорівнює $h(s)$. Якщо підпапка c папки u отримує множник k_c, то найкраще, чого може досягти її піддерево, — це $k_c \cdot h(c)$.

Ключове твердження. Нехай $g_1 \geq g_2 \geq \ldots \geq g_m$ — значення h для $m \geq 1$ підпапок папки u. Тоді

$$h(u) = \max_{1 \leq i \leq m} i \cdot g_i,$$

тобто підпапка з i-м за величиною значенням отримує множник i.

Доведення. По-перше, досить множників $1, 2, \ldots, m$: j-те найменше з m попарно різних додатних цілих чисел не менше за j, тож заміна його на j зберігає попарну різність множників і не збільшує жодного добутку. По-друге, більше значення має отримати менший множник. Якщо $g \geq g'$ і $a < b$, то

$$\max(ag,\ bg') \leq bg \leq \max(bg,\ ag'),$$

тож обмін множниками двох підпапок з $g \geq g'$, які отримали $b > a$, ніколи не збільшує максимуму (аргумент обміну). Після всіх таких обмінів множники $1, \ldots, m$ дістаються підпапкам у порядку спадання h.

Значення лишаються малими: $i \cdot g_i \leq g_1 + \ldots + g_i$, тож за індукцією $h(u)$ ніколи не перевищує розміру піддерева u. Відповідь не більша за n, хоча це добуток багатьох множників, і 32-бітних цілих досить.

Алгоритм. Запустимо BFS з s, щоб знайти батьківські папки, а потім оброблятимемо папки у зворотному порядку BFS: зберемо значення h підпапок, відсортуємо їх за спаданням і візьмемо $\max_i i \cdot g_i$. Кожна папка сортується один раз як підпапка, тож сортування загалом коштує $O(n \log n)$.

Складність — $O(n \log n)$ на тестовий випадок.

B Побітові вибоїни

Кожен маршрут гладкий тоді й лише тоді, коли на всіх дорогах однакова кількість вибоїн. Якщо всі кількості рівні, AND і OR кожного маршруту збігаються. Навпаки, візьмемо дві дороги e і f. Видалення e розбиває дерево на дві частини; нехай a — кінець e у частині без f, і симетрично визначимо b для f. Маршрут від a до b містить і e, і f. AND дорівнює OR на маршруті тоді й лише тоді, коли кожен біт однаковий на всіх його дорогах, тож кількості вибоїн на e і f рівні.

Біти незалежні. Операція змінює лише біт i, тому кожен біт розв'язуємо окремо: оберемо цільове значення $c \in \{0, 1\}$ біта i, і нехай S — множина доріг, у яких біт i відрізняється від c. Маршрути для біта i мають змінити рівно дороги з S, тобто їхня симетрична різниця має дорівнювати S.

Вартість S дорівнює $\mathrm{odd}(S)/2$, де $\mathrm{odd}(S)$ — кількість міст із непарною кількістю доріг з S (вона парна за лемою про рукостискання). Нижня оцінка: маршрут змінює парність степеня лише двох своїх кінців, тож кожне непарне місто S є кінцем якогось маршруту, і потрібно щонайменше $\mathrm{odd}(S)/2$ маршрутів. Верхня оцінка: довільно розіб'ємо непарні міста на пари й візьмемо маршрути між парами. Їхня симетрична різниця P має ті самі непарні міста, що й S. У дереві це визначає множину: дорога належить P (або S) тоді й лише тоді, коли частина з одного її боку містить непарну кількість непарних міст, бо сума степенів у цій частині дорівнює подвоєній кількості внутрішніх доріг плюс один за саму дорогу, якщо вона належить множині. Отже, $P = S$.

Підрахунок. Нехай X_v — XOR кількостей вибоїн на дорогах при місті v, а d_v — його степінь. Для цілі $c = 0$ парність v в S — це біт i числа X_v; для $c = 1$ — цей біт XOR $(d_v \bmod 2)$. Підрахувавши ці парності за всіма містами, отримаємо $\mathrm{odd}_c(i) = \mathrm{odd}(S)$ для обох цілей, і відповідь дорівнює

$$\sum_{i=0}^{29} \frac{\min(\mathrm{odd}_0(i),\ \mathrm{odd}_1(i))}{2}.$$

Старші біти дорівнюють 0 на всіх дорогах і нічого не коштують. Складність — $O(30 \cdot n)$ на тестовий випадок.

C Зручні партії

Ключове спостереження. Зафіксуймо лівий кінець i і розширюймо групу праворуч. Значення $\gcd(a_i, \dots, a_j)$ утворюють ланцюжок дільників, і щоразу, коли значення змінюється, воно стає власним дільником попереднього, тобто щонайменше вдвічі меншає. Оскільки $a_i \le 10^9 < 2^{30}$, різних значень не більше ніж 31. Отже, префіксні gcd будь-якого відрізка розбиваються на $O(\log A)$ блоків послідовних префіксів з однаковим gcd, і те саме справджується для суфіксів.

Дерево відрізків. У кожній вершині зберігаємо: cnt — кількість зручних груп усередині неї; tot — gcd усієї вершини; pre — список різних префіксних gcd у порядку зростання префікса, кожен разом з кількістю префіксів, що його мають; suf — те саме для суфіксів. Листок зі значенням x має $\text{cnt} = [x > 1]$ і $\text{pre} = \text{suf} = ((x, 1))$.

Щоб об'єднати лівого сина L з правим сином R: $\text{tot} = \gcd(L.\text{tot}, R.\text{tot})$; pre — це L.pre, після якого йдуть блоки $(\gcd(L.\text{tot}, g), c)$ для (g, c) з R.pre, причому однакові сусіди зливаються; suf — симетрично. Нарешті, $\text{cnt} = L.\text{cnt} + R.\text{cnt} + \text{cross}$, де cross — кількість зручних груп, що перетинають середину, тобто пар (суфікс L, префікс R) з $\gcd(s, g) > 1$ з урахуванням кратностей.

Підрахунок пар через середину. Якщо $\gcd(s, g) > 1$ для суфіксного gcd s і префіксного gcd g, то це справджується і для кожного коротшого суфікса та кожного коротшого префікса, бо їхні gcd кратні s і g. Тож коли суфікс росте, множина префіксів, що дають з ним gcd більший за 1, є префіксом R.pre, який може лише зменшуватися. Проходимо L.suf у порядку зростання довжини й рухаємо вказівник по R.pre назад, поки поточна пара має gcd 1, підтримуючи сумарну кратність решти блоків префіксів. Це $O(\log A)$ обчислень gcd на одне об'єднання.

Операції. Оновлення змінює листок і заново об'єднує його $O(\log n)$ предків. Запит $[l, r]$ згортає $O(\log n)$ канонічних вершин зліва направо тим самим об'єднанням; акумулятору потрібні лише cnt, tot і suf. Відповідь може сягати $n(n+1)/2 \approx 4.5 \cdot 10^{10}$, тож потрібні 64-бітні цілі.

Складність — $O((n+q)\log n \log A)$ обчислень gcd, де $A = 10^9$. Пам'ять — $O(n)$, бо вершина довжини k має списки довжини не більше ніж $\min(k, 31)$.

D Дні тижня

Пронумеруємо дні тижня від 0 (`monday`) до 6 (`sunday`), і нехай d_1, d_2, d_3 — дні тижня 1/8 фіналу, 1/4 фіналу та SEERC. Якщо один етап відбувся в день тижня a, а наступний — у день тижня b, то кількість днів між ними додатна й конгруентна $b - a$ за модулем 7. Отже, вона не менша за $g(a, b)$, де $g(a, b) = (b - a) \bmod 7$ береться з $\{0, \dots, 6\}$, але 0 замінюється на 7: два етапи в один і той самий день тижня відбуваються в різні календарні дні, тож між ними щонайменше тиждень.

Кількість днів від 1/8 фіналу до SEERC включно дорівнює сумі двох проміжків плюс один, тож вона не менша за $g(d_1, d_2) + g(d_2, d_3) + 1$. Це значення досягається, якщо кожен етап провести в найраніший відповідний день після попереднього. Отже, відповідь — $g(d_1, d_2) + g(d_2, d_3) + 1$, і вона завжди лежить між 3 і 15.

У C++ і Java $(b - a) \,\%\, 7$ від'ємне, коли $b < a$; використовуйте $((b - a) \,\%\, 7 + 7) \,\%\, 7$.

Складність — $O(1)$ на один тестовий випадок.

E Екзаменаційна сесія

Пройдемо дні за зростанням і для кожного дня вирішимо, який іспит скласти в цей день. **Жадібний алгоритм (спершу найближчий дедлайн):** серед іспитів, які ще не складено і вікно яких уже відкрилося ($l_i \le d$, де d — поточний день), беремо той, у якого r_i найменше. Якщо це $r_i < d$, відповідь `NO`; якщо кожен іспит отримав день, відповідь `YES`.

Чому це правильно. Нехай деякий допустимий вибір днів збігається з жадібним на всіх днях до d, і нехай e — іспит, обраний у день d. У цьому виборі $x_e \ge d$: кожен попередній день зайнятий іншим іспитом або був пропущений, коли всі l_i решти іспитів були більшими. Отже, $r_e \ge x_e \ge d$, і в день d жадібний алгоритм не відповідає `NO`. Якщо день d у цьому виборі вільний, перенесемо e на день d. Якщо день d зайнятий іншим іспитом f, поміняємо їх: e переходить на день d, а f — на день x_e, і це можна, бо $l_f \le d < x_e \le r_e \le r_f$ (жадібний алгоритм обрав e, тож $r_e \le r_f$). В обох випадках вибір лишається допустимим і тепер збігається з жадібним і в день d. За індукцією, якщо допустимий вибір існує, жадібний алгоритм ніколи не відповідає `NO`; а якщо він відповідає `YES`, то обрані ним дні і є допустимим вибором.

Реалізація. Відсортуємо іспити за l_i і зберігатимемо відкриті в мін-купі за r_i. Дні сягають 10^9, тому не перебираємо їх по одному: коли купа порожня, одразу переходимо до наступного l_i. Кожен крок циклу виймає з купи один іспит, тож кроків рівно n.

Той самий критерій можна сформулювати через теорему Голла: відповідь `YES` тоді й лише тоді, коли кожен відрізок днів $[a, b]$ містить не більше ніж $b - a + 1$ вікон $[l_i, r_i]$ цілком. Жадібний алгоритм перевіряє це неявно.

Складність — $O(n \log n)$ на тестовий випадок.

F Чотири сходинки

Для рядка цифр u позначимо через c_0, c_{01}, c_{012} і c_{0123} кількості його підпослідовностей, що дорівнюють `0`, `01`, `012` і `0123`; кількість сходжень — це c_{0123}. Дописування однієї цифри змінює рівно один лічильник, і лише додаванням іншого:

$$\texttt{0}\colon\ c_0 \mathrel{+}= 1, \qquad \texttt{1}\colon\ c_{01} \mathrel{+}= c_0, \qquad \texttt{2}\colon\ c_{012} \mathrel{+}= c_{01}, \qquad \texttt{3}\colon\ c_{0123} \mathrel{+}= c_{012}.$$

Справді, нове входження `01` мусить закінчуватися новою `1`, а його `0` — будь-який із c_0 попередніх нулів; для інших цифр аналогічно.

Ключове спостереження. Нам потрібне лише $c_{0123} \bmod 4$, а оскільки оновлення використовують тільки додавання, усі чотири лічильники можна від самого початку зберігати за модулем 4: взяття за модулем 4 комутує з додаванням, тож зведені лічильники після кожного префікса — це саме справжні лічильники за модулем 4. Справжні кількості величезні (сходжень може бути більше за 2^{64}), але вони ніколи не потрібні.

Отже, увесь рядок обробляє автомат із $4^4 = 256$ станами $(c_0, c_{01}, c_{012}, c_{0123}) \bmod 4$, що стартує в $(0,0,0,0)$. Нехай $\mathrm{dp}[q]$ — кількість способів заповнити `?` поточного префікса так, щоб він закінчився у стані q, за модулем $998\,244\,353$. Фіксована цифра переносить кожне $\mathrm{dp}[q]$ у стан, заданий її оновленням; `?` додає $\mathrm{dp}[q]$ до кожного з чотирьох наступників q. Відповідь — сума $\mathrm{dp}[q]$ за 64 станами з $c_{0123} \equiv 0 \pmod 4$. Зберігати треба лише поточний шар dp; якщо заздалегідь обчислити наступника кожного стану для кожної цифри, внутрішній цикл стає зовсім простим.

Усі 256 станів потрібні: наприклад, зберігати c_0 лише за модулем 2 неправильно, бо згодом c_0 множиться на кількість `123` після нього, а вона може бути непарною.

Складність — $O(256 \cdot 4 \cdot |s|)$ на тестовий випадок, пам'ять — $O(256)$.

G Згруповані вершини

Для вершини c позначимо через $B(c, r)$ множину вершин на відстані не більше r від c, а для ребра uv покладемо $B(uv, r) = B(u, r) \cup B(v, r)$. Обидві множини зв'язні (кожну вершину з'єднано з u або v шляхом усередині множини), а їхні діаметри не перевищують $2r$ і $2r + 1$ відповідно.

Ключове спостереження. Відповідь для x — це розмір найбільшої кулі відповідної парності:

$$\max_c |B(c, x/2)| \text{ для парного } x, \qquad \max_{uv} |B(uv, (x-1)/2)| \text{ для непарного } x$$

(і 1 для непарного x, якщо $n = 1$).

За наведеними оцінками ці кулі є допустимими множинами. Навпаки, нехай S — зв'язна множина з діаметром $D \le x$. Оскільки S зв'язна, відстані всередині S збігаються з відстанями в дереві. Візьмемо в S шлях $a \dots b$ довжини D і його середину: вершину c, якщо D парне, або ребро uv, якщо D непарне. Кожна вершина $w \in S$ лежить на відстані не більше $\lfloor D/2 \rfloor$ від середини. Справді, нехай p — найближча до w вершина шляху $a \dots b$, скажімо, на половині ближче до a; тоді маршрут від w до b проходить через p, а потім через середину, тож якби w була далі ніж $\lfloor D/2 \rfloor$ від середини, ми отримали б $d(w, b) > D$. Тепер вмістимо S у кулю потрібної парності:

- x парне, D парне: $S \subseteq B(c, D/2) \subseteq B(c, x/2)$;
- x парне, D непарне: $S \subseteq B(uv, (D-1)/2) \subseteq B(u, (D+1)/2) \subseteq B(u, x/2)$;
- x непарне, D непарне: $S \subseteq B(uv, (D-1)/2) \subseteq B(uv, (x-1)/2)$;
- x непарне, D парне: $S \subseteq B(c, D/2) \subseteq B(cv, (x-1)/2)$ для будь-якого сусіда v вершини c (він існує, якщо $n > 1$).

Алгоритм. Запустимо BFS з кожної вершини c і порахуємо вершини на кожній відстані; префіксні суми цих кількостей дають $|B(c, r)|$ одразу для всіх r (для r, більших за найбільшу відстань, куля — це все дерево). Для кожного r зберігаємо максимум. Те саме робимо з BFS із двох джерел — обох кінців кожного ребра. Це $2n - 1$ запусків BFS по $O(n)$ кожен, тож складність — $O(n^2)$ за часом і $O(n)$ за пам'яттю на один тестовий випадок.

H Зала пісень

За лінійністю математичного сподівання відповідь дорівнює сумі за всіма n початковими піснями ймовірностей того, що пісню ще знають після k вечорів. Усі пісні симетричні, тож відповідь дорівнює $n \cdot s$, де s — ймовірність того, що одна фіксована пісня збережеться.

Стежитимемо за цією піснею, і нехай c — кількість музикантів, які її знають; спочатку $c = 1$. Вечір змінює c, лише якщо пісню знає рівно один із a і b. Якщо її знає b, а a не знає, то c збільшується на одиницю; якщо її знає a, а b не знає, то c зменшується на одиницю. Кожна з цих двох подій має ймовірність

$$r_c = \frac{c\,(n-c)}{n\,(n-1)},$$

а з імовірністю $1 - 2r_c$ нічого не змінюється. Наступне значення c залежить лише від поточного, тож c — ланцюг Маркова на станах $0, 1, \dots, n$, і пісня зникає назавжди саме тоді, коли c досягає 0 (тоді $r_0 = 0$).

Нехай T — матриця переходів цього ланцюга розміру $(n+1) \times (n+1)$: $T_{c,c-1} = T_{c,c+1} = r_c$ і $T_{c,c} = 1 - 2r_c$. Розподіл c після k вечорів — це рядок 1 матриці T^k, отже,

$$\text{відповідь} = n \cdot \bigl(1 - (T^k)_{1,0}\bigr).$$

Усі обчислення можна виконувати одразу за модулем $M = 998\,244\,353$: імовірності r_c стають $c(n-c) \cdot (n(n-1))^{-1} \bmod M$, а піднесення матриці до степеня використовує лише додавання й множення, тож результат — саме потрібний лишок $p \cdot q^{-1} \bmod M$. Обчислимо T^k двійковим піднесенням до степеня: не більше ніж $2\log_2 k \approx 120$ множень матриць розміру не більше 101×101. Зауважимо, що для k потрібен 64-бітний тип. Зводити k за модулем $M - 1$ не можна: матриця T вироджена, тож мала теорема Ферма до неї не застосовна.

Складність — $O(n^3 \log k)$.

Зауваження. Якщо дивитися назад у часі, родовідні лінії n музикантів зливаються, як у коалесценті Кінгмена, і це дає замкнену формулу, яку можна обчислити за $O(n \log k)$; тут вона не потрібна.

I Незручні обміни

Два елементи можна поміняти місцями тоді й лише тоді, коли їхні значення мають однакову остачу від ділення на 3, де остача береться з $\{0, 1, 2\}$. Розіб'ємо позиції на три класи за значенням $a_i \bmod 3$. Значення ніколи не змінюються, а операція зачіпає дві позиції одного класу, тож кожен клас зберігає на своїх позиціях свою мультимножину значень.

Навпаки, будь-які два елементи одного класу можна поміняти місцями безпосередньо, а кожна перестановка є добутком транспозицій. Отже, значення класу можна розставити на позиціях цього класу в будь-якому порядку незалежно від інших класів. Досяжні масиви — це рівно ті масиви, у яких кожен клас містить свої значення в якомусь порядку.

Нехай b — масив, у якому кожен клас отримує свої значення, відсортовані за зростанням і розставлені на його позиціях зліва направо. Це лексикографічно найменший досяжний масив: візьмемо будь-який інший досяжний масив c і першу позицію i, де $b_i \neq c_i$, і нехай r — клас позиції i. Позиції класу r ліворуч від i містять у b і в c однакові значення, тож c_i — одне з решти значень класу r, а b_i — найменше з них. Отже, $b_i < c_i$.

Тому для кожного класу збираємо його позиції та значення, сортуємо значення й записуємо їх назад по порядку. Єдина пастка — остача від'ємного числа: у C++ і Java $-1 \,\%\, 3 = -1$, хоча $2 - (-1) = 3$ ділиться на 3, тож слід використовувати $((a_i \,\%\, 3) + 3) \,\%\, 3$.

Складність — $O(n \log n)$ на один тестовий випадок.

J Банка фарб

Мазок лише копіює кольори праворуч. Тому в кожному досяжному розфарбуванні b кожна дошка i має джерело $p(i) \le i$, для якого $b_i = c_{p(i)}$ (спочатку $p(i) = i$). Мазок (l, r) робить $p(l), \dots, p(r)$ рівними $p(l)$, і p лишається неспадною: $p(j) \le p(l)$ для $j < l$ і $p(j) \ge p(r) \ge p(l)$ для $j > r$.

Твердження. b досяжне тоді й лише тоді, коли існує неспадна p, для якої $p(i) \le i$ і $b_i = c_{p(i)}$ для всіх i.

Необхідність показано вище. Навпаки, кожне використане джерело k обслуговує відрізок $[s_k, e_k]$, де $k \le s_k$. Зробимо мазки (k, e_k) для використаних джерел справа наліво. У цей момент дошка k ще не змінена, бо попередні мазки починаються правіше від k. Мазок фарбує також $[k, s_k - 1]$, але ці дошки належать меншим джерелам і будуть перефарбовані пізніше, а всі наступні мазки закінчуються до s_k.

Кожне розфарбування рахуємо один раз. Розіб'ємо b на максимальні блоки однакового кольору; блок j починається з дошки s_j і має колір v_j. Якщо замінити всі джерела всередині блоку на найменше з них, p лишиться коректною, тож блок j може мати одне джерело k_j, де $1 = k_1 < k_2 < \dots$ (сусідні блоки мають різні кольори) і $k_j \le s_j$. Виберемо джерела жадібно: k_j — перша дошка після k_{j-1} кольору v_j. За індукцією жадібне k_j ніколи не стоїть правіше за j-те джерело будь-якої коректної p, тож b досяжне тоді й лише тоді, коли його жадібний ланцюжок існує і $k_j \le s_j$ для всіх j.

Нехай prv_k — попередня дошка кольору c_k (0, якщо такої немає). Жадібний ланцюжок може перейти від k до k' тоді й лише тоді, коли $\mathrm{prv}_{k'} < k < k'$; це також виключає $c_{k'} = c_k$. Отже, досяжні розфарбування взаємно однозначно відповідають парам (жадібний ланцюжок, початки блоків $1 = s_1 < s_2 < \dots$) з $k_j \le s_j$.

Нехай $f_i[k]$ — кількість таких префіксів довжини i, останній блок яких має джерело k. Тоді $f_1[1] = 1$ і для $k' \le i + 1$

$$f_{i+1}[k'] = f_i[k'] + \sum_{k=\mathrm{prv}_{k'}+1}^{k'-1} f_i[k].$$

Перший доданок продовжує останній блок; сума починає новий блок з дошки $i + 1$, що дозволено саме тому, що $k' \le i + 1$. Відповідь — $\sum_k f_n[k]$. (Для попарно різних кольорів це число Каталана C_n.)

З префіксними сумами f_i кожен крок займає $O(n)$; prv знаходимо сортуванням пар (c_i, i). Складність — $O(n^2)$ за часом і $O(n)$ за пам'яттю на тестовий випадок.

K Знаєте цю задачу?

Назвемо неспадну підпослідовність U, а незростаючу — D, і оброблятимемо елементи зліва направо. Чи вдасться розподілити решту масиву, залежить лише від останнього елемента U та останнього елемента D. Новий елемент x може приєднатися до U, якщо він не менший за останній елемент U, і до D, якщо він не більший за останній елемент D. Тож менший останній елемент U і більший останній елемент D ніколи не гірші. Порожня U поводиться як $-\infty$, а порожня D — як $+\infty$.

Досить двох чисел. Після обробки h_i один із двох останніх елементів — це саме h_i. Тому зберігаємо лише

- u: найбільший можливий останній елемент D серед усіх допустимих розбиттів $h_1, \dots, h_i$ з $h_i \in U$ ($+\infty$, якщо D порожня, і «неможливо», якщо такого розбиття немає);
- d: найменший можливий останній елемент U серед усіх допустимих розбиттів з $h_i \in D$ ($-\infty$, якщо U порожня, і «неможливо», якщо такого немає).

Спочатку $u = +\infty$, $d = -\infty$. Нехай $p = h_i$, $x = h_{i+1}$.

- Із $h_i \in U$ (останній елемент U — p, останній елемент D — u): якщо $p \le x$, то x може приєднатися до U, це кандидат u для нового u; якщо $u \ge x$, то x може приєднатися до D, це кандидат p для нового d.
- Із $h_i \in D$ (останній елемент D — p, останній елемент U — d): якщо $p \ge x$, це кандидат d для нового d; якщо $d \le x$, це кандидат p для нового u.

Новий u — максимум його кандидатів, новий d — мінімум його кандидатів («неможливо», якщо кандидатів немає). Відповідь `YES` тоді й лише тоді, коли наприкінці u або d можливе. Це $O(n)$ часу, і масив навіть не треба зберігати.

Усі порівняння нестрогі, бо однакові значення дозволені в обох підпослідовностях: розв'язок для строго монотонних підпослідовностей не проходить. Тримайте «неможливо» окремо від «порожня» і від справжніх значень: 10^9 — допустиме значення, тож воно не може правити за $+\infty$.

Зауваження: як знайти розбиття. Для кожного i запам'ятовуємо, який перехід дав u і d, і йдемо назад від $i = n$. Рівносильний жадібний алгоритм: якщо h_i підходить лише одній підпослідовності, віддаємо його їй; якщо обом, дивимося на наступне значення $w \neq h_i$ і віддаємо h_i до D, якщо $w < h_i$, і до U інакше. Дивитися лише на h_{i+1} недостатньо, бо перед w може стояти довга серія однакових значень.

L Загублений список

Лижник з обмеженнями (x, y) вважає трасу типу (i, j) придатною саме тоді, коли $x \geq i$ і $y \geq j$, тож $a_{i,j}$ — це сума $c_{x,y}$ за всіма $x \geq i$, $y \geq j$: a — це двовимірна суфіксна сума c. Вона складається з двох одновимірних суфіксних сум, уздовж рівнів крутизни і уздовж рівнів довжини, і кожну з них однозначно обертаємо: якщо $s_j = v_j + v_{j+1} + ... + v_m$, то $v_j = s_j - s_{j+1}$ (де $s_{m+1} = 0$). Отже, цілі числа c з потрібними сумами завжди існують і єдині:

$$c_{x,y} = a_{x,y} - a_{x+1,y} - a_{x,y+1} + a_{x+1,y+1},$$

де поза таблицею (для $i = n + 1$ або $j = m + 1$) a дорівнює 0. Результати опитування несуперечливі тоді й лише тоді, коли всі ці $c_{x,y}$ невід'ємні; інакше відповідь -1. Перевіряйте кожну пару (x, y), зокрема $x = n$ та $y = m$, і не беріть значення з попереднього тестового випадку як нульову межу.

Кожне $c_{x,y}$ лежить між $-2 \cdot 10^9$ та $2 \cdot 10^9$, тож 32-бітного `int` достатньо.

Складність $O(nm)$ на тестовий випадок.

M Мозаїчна арка

Перефарбування. Нехай $x_{i,j}$ — це колір клітинки (i, j), якщо $i+j$ парне, і протилежний колір інакше. У двох сусідніх за стороною клітинок парність $i+j$ різна, тому їхні кольори різні тоді й лише тоді, коли їхні значення x рівні. Сусідні клітинки П-фігури є сусідніми за стороною, отже, П-фігура гарна тоді й лише тоді, коли всі її клітинки мають однакове x.

Нехай $d_{i,j}$ — кількість клітинок у найдовшому вертикальному відрізку з однаковим x, що починається в (i, j) і йде вниз (обчислюємо від нижнього рядка вгору). Зафіксуємо r, c_1, c_2. Клітинки $(r, c_1), \dots, (r, c_2)$ мають лежати всередині однієї серії рядка r — максимального відрізка з однаковим x; тоді П-фігура з цими r, c_1, c_2 і заданим k гарна тоді й лише тоді, коли $k \le d_{r,c_1}$ і $k \le d_{r,c_2}$. Усі $a_{i,j}$ додатні, тож для фіксованих r, c_1, c_2 найкраще взяти $k = \min(d_{r,c_1}, d_{r,c_2})$, якщо воно не менше за 2.

Групування пар за k. У широкій таблиці пар (c_1, c_2) забагато, тому згрупуємо їх за k. Нехай $P_j = a_{r,1} + \dots + a_{r,j}$ і $L_k(c) = a_{r+1,c} + \dots + a_{r+k-1,c}$ (друге обчислюється за $O(1)$ через префіксні суми по стовпцях). П-фігура (r, c_1, c_2, k) має вагу

$$\big(L_k(c_1) - P_{c_1-1}\big) + \big(P_{c_2} + L_k(c_2)\big).$$

Зафіксуємо серію і переберемо різні значення $k \ge 2$ серед $d_{r,c}$ у цій серії. Для кожного k назвемо активними стовпці серії з $d_{r,c} \ge k$ і пройдемо їх зліва направо: для активного c_2 додаємо його праву дужку до максимуму лівої дужки за активними $c_1 \le c_2 - 2$ — це префіксний максимум, який підтримуємо другим вказівником. Кожне отримане значення є вагою гарної П-фігури (бо $k \le \min(d_{r,c_1}, d_{r,c_2})$), а кожна пара стовпців отримує своє найкраще значення в проході з $k = \min(d_{r,c_1}, d_{r,c_2})$. Отже, відповідь — максимум за всіма проходами, або -1, якщо жоден прохід не знайшов пари.

Складність. Серія ширини w має не більше $\min(w, n)$ різних значень d, а кожен прохід займає $O(w)$ часу (якщо тримати активні стовпці у зв'язному списку й після проходу видаляти ті, де $d_{r,c} = k$, він стає ще швидшим). Сумарна ширина серій рядка не перевищує m, тож рядок коштує $O(m \cdot \min(n, m))$, а складність — $O(nm \cdot \min(n, m)) \subseteq O((nm)^{3/2})$ на тестовий випадок, плюс сортування значень d у кожній серії. Відповідь може перевищувати 10^{14}, тому потрібні 64-бітні цілі числа.