

UCPC 2026 - I stage

Ukrainian Collegiate Programming Contest

3 жовтня 2026 р.

Розбір

A Антивірус
B Колекційні пари
C Танок залишків
D Парні та непарні
E Чотири числа
F Переплутані покази
G Числова гра
H Візерунок
I Круглий стіл
J Терези
K Сума і добуток
L Три комп'ютери
M Гірлянда UCPC
N Щоденник погоди

A Антивірус

Моделювати кожен запуск окремо — це до $\Theta(nk)$ дій, тому запуски треба об'єднати. Для папки v позначимо через b_v найбільшу кількість рівнів, на яку ще може поширитися вглиб від v якийсь запуск:

$$b_v = \max\{\, d_j - \operatorname{dist}(c_j, v) \,\},$$

де $\operatorname{dist}(c_j, v)$ — кількість рівнів від c_j униз до v, а максимум береться за запусками j, для яких v лежить усередині c_j (на будь-якій глибині) або $v = c_j$ ($b_v = -1$, якщо таких немає). Папка v перевірена рівно тоді, коли $b_v \geq 0$.

Запуск потрапляє в $v \neq c_j$ лише через a_v, де в нього залишається на один рівень більше. Отже,

$$b_v = \max\Big(\max_{j:\ c_j = v} d_j,\ \ b_{a_v} - 1\Big),$$

а для кореневої папки лишається тільки перший доданок. Оскільки $a_v < v$, обчислюючи b_v для $v = 1, 2, \ldots, n$ саме в такому порядку, ми завжди вже знаємо b_{a_v}; ні DFS, ні явне дерево не потрібні. Відповідь — кількість v, для яких $b_v \geq 0$. Кілька запусків для однієї папки треба об'єднувати максимумом, а не перезаписувати.

Складність — $O(n + k)$.

B Колекційні пари

Для величезного m_i перебір неможливий, тож будуємо x з дуже малою сумою цифр. Візьмемо $x = c \cdot 10^B$, де сума цифр c дорівнює 8. Тоді сума цифр x дорівнює 8, а в $x+1$ (останній нуль просто замінюється одиницею) — 9. Якщо $B \geq 3$, то $8 \mid 1000 \mid 10^B$, тож $8 \mid x$. Число із сумою цифр 8 дає остачу 8 за модулем 9, тому $x+1 \equiv 0 \pmod 9$. Отже, x і $x+1$ утворюють красиву пару для будь-якого такого c і будь-якого $B \geq 3$.

Залишається вмістити пару в пачку: $m_i \leq x$ і $x+1 \leq 2m_i$. Нехай m_i має $L \geq 5$ цифр, p — число з перших двох його цифр, а $B = L-2 \geq 3$, тож $p \cdot 10^B \leq m_i < (p+1) \cdot 10^B$. Досить вибрати c з $p < c < 2p$: тоді $m_i < (p+1) \cdot 10^B \leq x$ і $x+1 < 2p \cdot 10^B \leq 2m_i$. П'ять значень c із сумою цифр 8 покривають усі $p \in [10, 99]$:

$$p \in [10, 16] \to 17, \quad [17, 25] \to 26, \quad [26, 34] \to 35, \quad [35, 61] \to 62, \quad [62, 99] \to 107,$$

бо в кожному проміжку $c > p$ і $c < 2 \cdot (\text{найменше } p)$. Отже, для $m_i \geq 10^4$ відповідь завжди існує: виводимо x — цифри c, за якими йде $L-2$ нулів.

Конструкції потрібно $B \geq 3$, тобто щонайменше п'ять цифр; за меншої кількості цифр $8 \nmid 10^B$, і вона не працює. Усі $m_i < 10^4$ обробляємо передобчисленням: позначаємо красиві числа до $2 \cdot 10^4$, для кожного m знаходимо найменше $x \geq m$, для якого x і $x+1$ утворюють красиву пару (прохід з кінця), і виводимо його, якщо $x+1 \leq 2m$. Якщо найменше таке x не підходить, то жодне більше теж не підходить. Значення x, для яких x і $x+1$ обидва красиві, починаються так: $1, 2, \ldots, 9, 20, 80, 110, \ldots$; з цієї таблиці видно, що відповідь -1 рівно для $m_i = 10$ (пара $20, 21$ виходить за межі $[10, 20]$) і для $21 \leq m_i \leq 40$ (наступна пара — $80, 81$, а $81 \leq 2m_i$ вимагає $m_i \geq 41$). Для кожного $41 \leq m_i < 10^4$ красива пара існує.

Складність — $O(10^4)$ на таблицю плюс $O(\text{сумарна довжина})$ на всіх колекціонерів.

C Танок залишків

Нехай r — спільний залишок. Остача менша за дільник, тому $r = a \bmod b < b$, $r = b \bmod c < c$ і $r = c \bmod a < a$; отже, r менше за кожне з чисел a, b, c.

Припустимо, що $a < b$. Тоді $a \bmod b = a$, тобто $r = a$, що суперечить $r < a$. Отже, $a \geq b$. Так само $b \geq c$ і $c \geq a$, тому $a \geq b \geq c \geq a$, тобто $a = b = c$. Навпаки, для кожної трійки з $a = b = c$ танець гармонійний: усі три залишки дорівнюють 0.

Отже, танець гармонійний рівно для трійок (x, x, x) з $1 \leq x \leq \min(n, m, k)$, і відповідь дорівнює $\min(n, m, k)$. Складність — $O(1)$ на тестовий випадок.

D Парні та непарні

Позначимо $\Delta(A) = S_{\text{odd}} - P_{\text{odd}} - S_{\text{even}} + P_{\text{even}}$. Цифра d на позиції i додає $+d$ або $-d$ через S-доданки ($+$, якщо d непарна) і $+d$ або $-d$ через P-доданки ($+$, якщо i парна). Отже, її внесок дорівнює $2d$, якщо d непарна, а i парна, $-2d$, якщо d парна, а i непарна, і 0 в інших випадках: A збалансоване тоді й лише тоді, коли сума непарних цифр на парних позиціях дорівнює сумі парних цифр на непарних позиціях.

Нехай M — це N, після якого дописано блок із k цифр. Блок займає позиції $0, \dots, k-1$, а кожна цифра N зсувається на k позицій, тому внесок N залежить лише від $k \bmod 2$. Обчислимо за $O(|N|)$ значення D_0 і D_1 величини $\Delta/2$ для цифр N, зсунутих на парну та на непарну кількість позицій. Тоді блок має задовольняти $x - y = T$, де $T = -D_{k \bmod 2}$, x — сума непарних цифр на його $e = \lceil k/2 \rceil$ парних позиціях, а y — сума парних цифр на його $o = \lfloor k/2 \rfloor$ непарних позиціях.

Перевірка за $O(1)$. Можливі значення y — це рівно парні числа з $[0, 8o]$. Сума j непарних цифр має парність j і набуває всіх значень цієї парності з $[j, 9j]$. Об'єднуючи за $j \le e$, досяжні x парності p — це всі числа парності p з $[p, 9E_p]$, де E_p — найбільше ціле $\le e$ парності p (при $p = 1$ і $e = 0$ досяжних немає). Оскільки $x \equiv T \pmod 2$, беремо $p = T \bmod 2$; блок існує тоді й лише тоді, коли $\max(T, p) \le \min(T + 8o,\ 9E_p)$. Парність тут суттєва: наприклад, дві непарні цифри ніколи не дають у сумі 11.

Найменше M. Оскільки N не має провідного нуля, коротше M менше, а блоки однакової довжини порівнюються лексикографічно. Знайдемо мінімальне k, перебираючи $k = 0, 1, 2, \dots$, а потім заповнимо блок від позиції $k-1$ до 0, щоразу ставлячи найменшу цифру, після якої решта позицій ще допускає розв'язок. На парній позиції достатньо пробувати $0, 1, 3, 5, 7, 9$, на непарній — $0, 2, 4, 6, 8$: будь-яка інша цифра нічого не вносить, так само як 0.

Існування і довжина. Явний блок існує завжди. Якщо $T \ge 0$, поставимо j непарних цифр на найнижчі парні позиції, де j — найменше ціле $\ge \lceil T/9 \rceil$ з $j \equiv T \pmod 2$; для цього потрібно $k \ge 2j - 1$. Якщо $T < 0$, поставимо 1 на позицію 0, коли T непарне, і запишемо $y = |T|$ (або $|T| + 1$) як суму $\lceil y/8 \rceil$ парних цифр на найнижчих непарних позиціях; для цього потрібно $k \ge 2\lceil y/8 \rceil$. Оскільки $|D_0|, |D_1| \le 9\lceil |N|/2 \rceil$, мінімальне k не перевищує $\frac{9}{8}|N| + O(1)$, тож M може бути більш ніж удвічі довшим за N, але перебір і побудова лишаються лінійними.

Складність — $O(|N| + |M|) = O(|N|)$ на запит.

E Чотири числа

Сума не залежить від порядку індексів, тож ми вибираємо мультимножину з чотирьох елементів масиву. Кожна така мультимножина розбивається на дві пари $\{i, j\}$ та $\{k, t\}$, і навпаки, будь-які дві пари $x \le y$ та $z \le w$, навіть та сама пара, взята двічі, дають допустиму четвірку.

Побудуємо всі $s = n(n+1)/2$ попарних сум $a_x + a_y$ з $x \le y$ і відсортуємо їх: $p_1 \le p_2 \le ... \le p_s$. Відповідь — це максимум $p_l + p_r \le m$ за $l \le r$; зауважимо, що $l = r$ дозволено (це потрібно, наприклад, при $n = 1$). Знайдемо його двома вказівниками: почнемо з $l = 1$, $r = s$ і, поки $l \le r$, якщо $p_l + p_r \le m$, оновимо відповідь і збільшимо l (для цього l більшого партнера не лишилося), інакше зменшимо r (ця сума перевищує m навіть з найменшим із решти p_l). Якщо жодна сума не підходить, що буває рівно тоді, коли $4 \min a_i > m$, виведемо -1. Відповідь може бути від'ємною, нулем або -1, тому слід тримати окремий прапорець, що хоча б одна сума підходить, а не значення-сторож. Усі суми за модулем не перевищують $4 \cdot 10^{18}$ і вміщуються у знакові 64-бітні цілі.

Прохід двома вказівниками займає $O(n^2)$ часу; єдиний нелінійний крок — сортування приблизно $4.5 \cdot 10^6$ чисел, одне послідовне сортування з малою константою. Формально складність становить $O(n^2 \log n)$ за часом та $O(n^2)$ за пам'яттю; якщо замінити сортування на порозрядне, вийде $O(n^2)$. Обмеження часу жорстке: розв'язки за $O(n^2 \log n)$ з великою константою, якот двійковий пошук у відсортованих сумах для кожної з n^2 пар (довільний доступ до пам'яті) чи збалансоване дерево пошуку попарних сум, можуть не вкластися, а розв'язки за $O(n^3)$ не вкладаються.

F Переплутані покази

Упорядкуємо невідомий масив: $a_1 \le a_2 \le \ldots \le a_n$. Відновлюватимемо його по одному елементу, щоразу беручи найменший показ, який ще не пояснено.

Ключове спостереження. Кожен показ групи T не менший за $\max_{i \in T} a_i$: сума додатних чисел не менша за кожен доданок, а АБО містить усі біти кожного операнда, тож не менше за кожен операнд. Нехай $a_1, \ldots, a_k$ уже відомі, і вилучимо з журналу $2 \cdot (2^k - 1)$ показів усіх непорожніх груп $S \subseteq \{1, \ldots, k\}$. Решта показів належить групам T, що містять якийсь індекс $j > k$, тому кожен із них не менший за $a_j \ge a_{k+1}$. А сама група $\{k+1\}$ дає рівно a_{k+1}. Отже, мінімум решти мультимножини дорівнює a_{k+1}. За індукцією це відновлює весь масив, а заразом доводить, що відповідь єдина з точністю до порядку.

Вилучення показів. Коли знайдено новий елемент $x = a_{k+1}$, нові групи — це $S \cup \{k+1\}$ для всіх $S \subseteq \{1, \ldots, k\}$, включно з $S = \varnothing$. Їхні покази — $\mathrm{sum}(S) + x$ та $\mathrm{or}(S) \mid x$. Зберігатимемо два масиви зі значеннями $\mathrm{sum}(S)$ та $\mathrm{or}(S)$ для всіх 2^k підмножин знайдених елементів (починаючи з порожньої множини зі значеннями 0 і 0); кожен новий елемент подвоює їх, дописуючи $\mathrm{sum}(S) + x$ та $\mathrm{or}(S) \mid x$.

Тепер відсортуємо b і пройдемо його за зростанням, підтримуючи мін-купу показів, які вже пояснено, але ще не зіставлено. Якщо поточне b_p дорівнює вершині купи, воно пояснене: видаляємо вершину й ідемо далі. Інакше b_p — найменший непояснений показ, тобто наступний елемент x; додаємо до купи $2 \cdot 2^k$ нових показів. Пара для $S = \varnothing$ двічі дорівнює x і зіставляється із самим b_p та його копією-близнюком. Усі додані значення не менші за $x = b_p$, тож жодне з них не лишилося позаду, і прохід обчислює саме різницю мультимножин. Зупиняємося після n-го елемента. Зауважимо, що однакові покази трапляються часто (наприклад, $\mathrm{sum}(S) = \mathrm{or}(S)$ для елементів без спільних бітів), тому пояснені значення треба рахувати з кратністю, а не зберігати як множину.

Складність — $O(2^n \cdot n)$ на тестовий випадок (сортування та купа).

G Числова гра

На першому кроці ходу Марта може стирати й дописувати нулі де завгодно, тож число з c одиницями можна перетворити на будь-яке число, що має рівно c одиниць (його старша цифра лишається одиницею). Отже, важливе лише c: один хід перетворює число на дошці на $x+1$ для довільного x з $\text{popcount}(x) = c$.

Як змінюється c за один хід? Якщо x закінчується рівно k одиницями ($0 \le k \le c$), то «+1» перетворює ці k одиниць на нулі, а нуль перед ними — на одиницю, тож $x+1$ має $c-k+1$ одиниць. Можливе будь-яке k, тому після ходу кількість може бути будь-яким значенням з $[1, c+1]$: вона може як завгодно зменшитися, але зростає щонайбільше на одиницю.

Останній хід мусить збільшити на 1 число $x = m-1$, тож безпосередньо перед ним на дошці має бути число з рівно $p = \text{popcount}(m-1)$ одиницями; і навпаки, будь-яке таке число можна перетворити на $m-1$. Нехай $c = \text{popcount}(n)$. Відповідь:

- 0, якщо $n = m$ (цю умову перевіряємо першою: вона охоплює $n = m = 1$);
- -1, якщо $m = 1$, бо на дошці мало б бути $x = 0$;
- 1, якщо $c = p$;
- 2, якщо $c > p$: один хід, щоб зменшити кількість до $p \ge 1$, і ще один, щоб отримати m;
- $p - c + 1$, якщо $c < p$: щоб кількість зросла до p, потрібно щонайменше $p - c$ ходів, і рівно $p - c$ достатньо.

Віднімання 1 перетворює кінцеві нулі m на одиниці, а наймолодшу одиницю — на нуль, тож $p = \text{popcount}(m) - 1 + (\text{кількість кінцевих нулів } m)$. Числа мають до 1000 двійкових цифр, тому все обчислюємо безпосередньо на рядках. Складність — $O(|n| + |m|)$ на один тестовий випадок.

H Візерунок

Нехай $u(n)$ — відповідь. Запишемо штамп як слово $w = w_1 \dots w_n$ з кольорів його клітинок. Бордером слова w назвемо слово довжини l, $1 \le l \le n-1$, яке є водночас префіксом і суфіксом w; штамп вдалий тоді й лише тоді, коли в нього немає бордерів. Порахуємо невдалі штампи за їхнім найкоротшим бордером.

Лема. Найкоротший бордер b штампа w з n клітинок сам є вдалим штампом, і $2|b| \le n$.

Доведення. Бордер c слова b є префіксом і суфіксом w, тобто коротшим бордером w; отже, b вдалий. Тепер припустимо, що $l = |b| > n/2$, і нехай $s = n - l < l$. Два входження b у w починаються в позиціях 1 та $s+1$ і мають спільний шматок $w_{s+1} \dots w_l$. Він є суфіксом першого входження і префіксом другого, тож це бордер b довжини $l - s \ge 1$ — суперечність. □

Навпаки, візьмемо будь-який вдалий b з $2|b| \le n$ і будь-яке розфарбування x з $n - 2|b|$ клітинок. Тоді найкоротший бордер штампа $w = bxb$ — саме b: коротший бордер w був би префіксом і суфіксом b, тобто бордером b. Отже, невдалі штампи з n клітинок взаємно однозначно відповідають таким парам (b, x), і

$$u(n) = k^n - \sum_{j=1}^{\lfloor n/2 \rfloor} k^{\,n-2j}\, u(j).$$

Якщо рахувати цю суму напряму, вийде $O(n^2)$. Віднімемо k, помножене на ту саму рівність для $n-1$: k^n скорочується, як і всі доданки з $j \le \lfloor (n-1)/2 \rfloor$. Для непарного n не лишається нічого; для $n = 2m$ лишається тільки доданок $j = m$, тобто $u(m)$:

$$u(1) = k, \qquad u(2m+1) = k\,u(2m), \qquad u(2m) = k\,u(2m-1) - u(m).$$

Для кожного запиту обчислюємо $u(1), \dots, u(n)$ за модулем $10^9 + 7$ з 64-бітними добутками, виправляючи знак після віднімання; один масив служить для всіх запитів.

Складність — $O(n)$ на запит, $O(\sum n)$ загалом.

I Круглий стіл

Розгорнімо коло: випишемо студентів двічі як позиції $1, 2, \dots, 2n$, де позиції x і $x+n$ — це один і той самий студент. Команда (l_i, r_i) стає відрізком $[l_i, e_i]$, де $e_i = r_i$, якщо $l_i \le r_i$, і $e_i = r_i + n$ інакше (його довжина від 1 до n; для всього кола $e_i = l_i + n - 1$), разом із копією $[l_i + n, e_i + n]$. Вікно $[p, p+n-1]$, де $1 \le p \le n$, містить кожного студента рівно один раз, тож попарно неперетинні відрізки всередині одного вікна — це попарно неперетинні команди.

Усередині фіксованого вікна маємо класичну задачу на прямій, яку розв'язує жадібний алгоритм: з поточної позиції x беремо відрізок, що починається в x або пізніше, з найменшим кінцем, і продовжуємо одразу після нього (перший відрізок будь-якого розв'язку можна замінити жадібним, який закінчується не пізніше). Отже, нехай

$$\mathrm{nxt}(x) = 1 + \min\{e : [s, e] \text{ — відрізок або копія, } s \ge x\};$$

це обчислюється для всіх $x \le 2n$ суфіксними мінімумами за $O(n+m)$; якщо такого відрізка немає або значення більше за $2n$, покладемо $\mathrm{nxt}(x) = 2n+1$, а $2n+1$ зробимо нерухомою точкою. t-та команда, яку бере жадібний алгоритм від p, закінчується в $\mathrm{nxt}^t(p) - 1$, тож жадібний алгоритм у вікні p формує $f(p)$ команд, де

$$f(p) = \max\{t : \mathrm{nxt}^t(p) \le p + n\}.$$

Відповідь дорівнює $\max_{1 \le p \le n} f(p)$. Справді, кожне $f(p)$ досяжне. Навпаки, візьмімо оптимальний набір і нехай p — перший студент однієї з його команд: жодна команда набору не містить водночас студента p і студента перед ним (інакше вона мала б спільного студента p із цією командою), тож у вікні p усі вони є відрізками, і жадібний алгоритм на прямій знаходить не менше. Одного розрізу недостатньо: у другому прикладі жадібний алгоритм від студента 1 формує лише 2 команди, а від студента 3 — 3.

Щоб швидко обчислити всі $f(p)$, використаємо двійкові підйоми: $\mathrm{up}_0 = \mathrm{nxt}$ і $\mathrm{up}_j(x) = \mathrm{up}_{j-1}(\mathrm{up}_{j-1}(x))$, тобто $\mathrm{up}_j = \mathrm{nxt}^{2^j}$, для $2^j \le n$ (завжди $f(p) \le n$). Оскільки $\mathrm{nxt}(x) > x$ для $x \le 2n$, послідовність $\mathrm{nxt}^t(p)$ неспадна за t, тож $f(p)$ шукаємо як двійковим пошуком: починаємо з $c = p$ і перебираємо j від найбільшого до нуля; щойно $\mathrm{up}_j(c) \le p + n$, виконуємо $c \leftarrow \mathrm{up}_j(c)$ і додаємо 2^j до лічильника. Це $O(\log n)$ на один початок.

Складність — $O(m + n \log n)$ за часом і $O(n \log n)$ за пам'яттю.

J Терези

Результат порівняння двох чисел визначає найстарший біт, у якому вони відрізняються. Знаходитимемо біти a та b згори донизу. Нехай A та B — уже відомі біти a та b вище за біт j, і робимо лише зважування (x, y), де $x = A \oplus s$, $y = B \oplus t$ для деяких $s, t < 2^{j+1}$. Відомі біти скорочуються, тож таке зважування порівнює $a' \oplus s$ з $b' \oplus t$, де $a' = a \bmod 2^{j+1}$ та $b' = b \bmod 2^{j+1}$ — невідомі частини. Підтримуємо c — результат порівняння a' з b'; спочатку $j = n-1$, а зважування $(0, 0)$ дає c.

Нехай α та β — біт j чисел a та b.

- Якщо c є `=`, то $a' = b'$, тому $a = \beta$, і одне зважування $(x, y) = (A \oplus 2^j, B)$ знаходить біт: після інвертування біт j стає єдиною відмінністю, тож відповідь `>` тоді й лише тоді, коли $a = \beta = 0$; c лишається `=`.
- Інакше робимо зважування $(A \oplus 2^j, B)$ та $(A, B \oplus 2^j)$; нехай r_1 та r_2 — відповіді. Якщо $a = \beta$, інвертований біт є найстаршою відмінністю, тож r_1 та r_2 строгі й протилежні (r_1 є `>` тоді й лише тоді, коли $a = \beta = 0$). Якщо $a \neq \beta$, інвертування робить біти j рівними, і обидві відповіді дорівнюють порівнянню нижніх j бітів. Отже, $r_1 = r_2$ тоді й лише тоді, коли $a \neq \beta$; тоді c показує, у якого числа одиниця, а r_1 стає новим c.

Це класичний розв'язок з $2n+1$ зважуванням. Одне спостереження зменшує це число до $2n$: на біті 0 при c, відмінному від `=`, невідомі частини — це окремі біти, і вони різні, тож c показує, який з них дорівнює 1, і зважування не потрібне. Якщо c ніколи не стає `=`, біти $n-1, \dots, 1$ коштують по два зважування, а біт 0 безкоштовний: $1 + 2(n-1) = 2n-1$ зважувань. Якщо c стає `=` на біті $j \geq 1$, кожен з j нижніх бітів коштує одне зважування, разом $1 + 2(n-j) + j = 2n - j + 1 \leq 2n$; якщо c є `=` від самого початку, $n+1 \leq 2n$. (Вистачає навіть $2n-1$ зважувань: якщо r_1 є `=`, то він не може походити від рівних бітів, тож $a \neq \beta$, нижні частини рівні, і r_2 можна пропустити.)

Журі може бути адаптивним, тому кожен біт треба виводити для найгіршого випадку: схему, яка на деяких парах використовує $2n+1$ зважування і «економить» одне, вгадуючи біт, буде спіймано.

Складність — $O(n)$.

K Сума і добуток

Одного числа достатньо лише при $s = 1$.

Два числа. Якщо $a_1 + a_2 = s$ і $a_1 a_2 = 1$, то a_1, a_2 — корені рівняння $t^2 - st + 1 = 0$, які дійсні тоді й лише тоді, коли $s^2 \geq 4$. Тож при $|s| \geq 2$ відповідь $n = 2$ (одного числа не вистачить, бо $s \neq 1$), а самі числа — $a_{1,2} = (s \pm \sqrt{s^2 - 4})/2$.

Три числа для всіх інших s, тобто при $|s| < 2$ і $s \neq 1$. Одного числа не вистачить, бо $s \neq 1$, а двох — бо для будь-яких двох дійсних чисел $a_1 a_2 \leq (a_1 + a_2)^2/4 = s^2/4 < 1$. Трьох чисел достатньо для будь-якого s:

- візьмемо $a_1 = -1$; тоді $a_2 + a_3 = s + 1$ і $a_2 a_3 = -1$, тобто a_2, a_3 — корені рівняння $t^2 - (s+1)t - 1 = 0$, дискримінант якого $(s+1)^2 + 4$ додатний;
- або при $s < 0$ візьмемо s, $1/\sqrt{-s}$, $-1/\sqrt{-s}$; при $s \geq 0$ — $s + 2x$, $-x$, $-x$, де $(s + 2x)x^2 = 1$: ліва частина зростає від 0 до $s + 2 > 1$ на $[0, 1]$, тож x знаходимо бінарним пошуком.

Точність. Оскільки $|s|$ може сягати 10^9, менший за модулем корінь при $n = 2$ має порядок 10^{-9}, і формула $(|s| - \sqrt{s^2 - 4})/2$ втрачає всі цифри через віднімання близьких чисел: обчислюйте більший за модулем корінь $y = \operatorname{sgn}(s)(|s| + \sqrt{s^2 - 4})/2$, а другий — як $1/y$, розбір випадків робіть за точним десятковим s (наприклад, за 64-бітним цілим $1000 \cdot s$), а не за округленим значенням з рухомою комою, і виводьте близько 17 значущих цифр.

Складність — $O(1)$ на одне значення s (з бінарним пошуком — близько 100 ітерацій).

L Три комп'ютери

Уявімо кожного учасника команди як шлях уздовж осі часу: він або чекає, або розв'язує задачу, і це переносить його з хвилини l_i у хвилину $r_i + 1$. Це потік із трьох одиниць.

Мережа. Стиснемо всі значення l_i та $r_i + 1$ у точки $x_0 < x_1 < \ldots < x_{m-1}$. Додамо ланцюгові ребра $x_j \to x_{j+1}$ з пропускною здатністю 3 і вартістю 0 (очікування), а для кожної задачі — ребро $l_i \to r_i + 1$ з пропускною здатністю 1 і вартістю $-w_i$. Пустимо потік величини 3 з x_0 у x_{m-1}; відповідь дорівнює $-$(мінімальна вартість). Кінець ребра — саме $r_i + 1$, а не r_i: задачі $[1, 3]$ і $[4, 5]$ дають ребра $1 \to 4 \to 6$ і можуть іти одна за одною, а задачі $[1, 3]$ і $[3, 5]$ дають ребра $1 \to 4$ і $3 \to 6$, які не можуть лежати на одному шляху.

Чому потік величини 3 — це саме розклад. Усі ребра ведуть вперед, тож мережа є DAG, і цілочисельний потік величини 3 розкладається на три шляхи з x_0 у x_{m-1} (циклів немає). Ребра задач на одному шляху попарно не перетинаються в часі, бо шлях, що пройшов ребром $l_i \to r_i + 1$, продовжується з $r_i + 1$; отже, кожен шлях — коректний розклад одного учасника, а пропускна здатність 1 означає, що жодна задача не використана двічі. Навпаки, розклад кожного учасника є таким шляхом (між задачами він чекає вздовж ланцюга), і три шляхи проходять кожним ланцюговим ребром не більше 3 разів. Вартості збігаються, тож мінімальна вартість дорівнює мінус найкращій сумі. Потік мінімальної вартості з цілими пропускними здатностями має цілочисельний оптимум, і алгоритм нижче саме такий і знаходить.

Початкові потенціали. Вартості від'ємні, тож Дейкстру напряму застосувати не можна. Але мережа є DAG, вершини якого вже впорядковані топологічно, тому один прохід по $x_0, x_1, \ldots$ дає найкоротші відстані $\varphi(v)$ від x_0. Після цього кожна зведена вартість $c(u, v) + \varphi(u) - \varphi(v)$ невід'ємна.

Три запуски Дейкстри. Шукаємо послідовно найкоротші шляхи: запускаємо Дейкстру з x_0 на зведених вартостях залишкової мережі, додаємо відстані до φ і проштовхуємо потік найкоротшим шляхом до x_{m-1} (на його вузьке місце, не більше ніж 3 мінус поточний потік). Після доповнення зведені вартості лишаються невід'ємними (стандартний аргумент про потенціали), а кожна вершина лишається досяжною вздовж ланцюга, залишкова пропускна здатність якого не менша за 3 − потік ≥ 1, тож φ завжди скінченні. Кожне доповнення проштовхує хоча б одну одиницю, тому буде не більше 3 запусків Дейкстри на графі з $O(n)$ вершинами та ребрами.

Для малих n вистачило б Беллмана–Форда чи SPFA на кожному доповненні, але за цих обмежень це занадто повільно: у залишковій мережі є від'ємні зворотні ребра, і на ній ці алгоритми можуть працювати $\Theta(n^2)$. Відповідь сягає $2 \cdot 10^{14}$, тож потрібні 64-бітні цілі.

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

M Гірлянда UCPC

Скажімо, що лампа i грає роль j ($1 \le j \le 4$), якщо в деякому входженні `UCPC` маємо $i_j = i$, а лампу, що не грає жодної ролі, назвімо вільною. В узгодженому розфарбуванні всі входження мають одне розфарбування (c_1, c_2, c_3, c_4), тому кожна лампа, що грає роль j, має колір c_j, а вільні лампи нічим не обмежені. Навпаки, будь-які $c_1, \dots, c_4$ і будь-які кольори вільних ламп дають узгоджене розфарбування, якщо лише лампа, що грає дві ролі, не отримує двох різних кольорів. Літери `U`, `C`, `P` різні, тож спільну лампу можуть мати лише ролі 2 і 4. Отже, якщо вільних ламп f, то відповідь дорівнює 3^{f+3}, коли деяка лампа грає обидві ролі 2 і 4, і 3^{f+4} інакше; якщо `UCPC` не входить узагалі, відповідь 3^n.

Нехай $l_1 < l_2 < l_3 < l_4$ — входження, знайдене жадібно зліва (кожне l_j — перша відповідна літера після l_{j-1}), а $r_1 < r_2 < r_3 < r_4$ — знайдене жадібно справа. Оскільки жадібне зіставлення оптимальне, лампу i з j-ю літерою слова `UCPC` можна доповнити першими $j-1$ літерами зліва тоді й лише тоді, коли $l_j \le i$, і останніми $4-j$ літерами справа тоді й лише тоді, коли $i \le r_j$. Тож лампа i грає роль j тоді й лише тоді, коли має потрібну літеру і $l_j \le i \le r_j$. Зокрема, деяка лампа грає ролі 2 і 4 тоді й лише тоді, коли $l_4 \le r_2$: така лампа лежить у $[l_4, r_2]$, і навпаки, якщо $l_4 \le r_2$, то обидві ролі грає сама лампа l_4. Зауважте, що нерівність нестрога: у першій гірлянді прикладу $l_4 = r_2 = 4$.

Два жадібні проходи і один прохід для підрахунку вільних ламп займають $O(n)$ часу на гірлянду, тож загальна складність $O(\sum n)$.

N Щоденник погоди

Позначимо сонячний день літерою `S`, а дощовий — літерою `R`, і розіб'ємо послідовність на максимальні серії днів з однаковою погодою. Кожна з $u+v$ змін погоди відбувається між двома сусідніми серіями, а зміни чергуються: `SR`, `RS`, `SR` і так далі. Отже, умова $|u-v| \le 1$ необхідна.

Якщо $u+v \ge 1$, вона й достатня. Візьмемо $u+v+1$ серій довжини 1 з погодою, що чергується, починаючи із сонячного дня, якщо $u \ge v$, і з дощового інакше; вони дають рівно u пар `SR` і v пар `RS`. Є серії обох видів, тож подовжимо якусь сонячну серію на e днів і якусь дощову на g днів: кожен доданий день створює одну пару `SS` або `RR` і не змінює інших лічильників. Загальна довжина дорівнює $u+v+1+e+g=n$.

Якщо $u=v=0$, погода ніколи не змінюється, тобто всі дні сонячні або всі дощові. Це можливо тоді й лише тоді, коли $e=0$ або $g=0$; саме цей випадок легко пропустити.

Отже, відповідь `YES` тоді й лише тоді, коли $|u-v| \le 1$ і не виконується ($u=v=0$, $e>0$ і $g>0$). Складність — $O(1)$ на один підсумок.