Задача A. Борги

Простежимо за гаманцем Аліси.

Стартова сума — a гривень. Боб і Світлана повертають свої борги, тому додаються b і c гривень. Далі Аліса повертає борг Дмитру, і d гривень віднімається.

Разом отримуємо

$$a + b + c - d.$$

Умова гарантує, що підсумок невід'ємний, тому окремо обробляти випадок «не вистачило грошей» не потрібно. Усі величини не перевищують 100, тож ані переповнення, ані дійсних чисел тут немає.

Задача B. Сир для Анфіси

Погляньмо на задачу з боку кількості шматків, а не розрізів.

Спочатку є рівно один шматок сиру. У кінці має вийти $a \cdot b \cdot c$ одиничних кубиків. Кожен розріз ділить якийсь один шматок на два, тобто збільшує загальну кількість шматків рівно на одиницю — ні більше, ні менше.

Отже, щоб перейти від 1 шматка до abc шматків, потрібно рівно

$$a \cdot b \cdot c - 1$$

розрізів. Цю кількість не можна ані зменшити (кожен розріз додає лише один шматок), ані обійти хитрішим порядком дій — відповідь не залежить від стратегії взагалі.

Технічна деталь: сторони сягають 1000, тому добуток може досягати 10^9. Це на межі 32-бітного знакового типу, тому надійніше взяти 64-бітний.

Задача C. Картини на стіні

Автор: Андрій Холод
Розробник: Володимир Балачевцев
Розбір підготовано: Андрій Холод

Усі картини разом займають $n \cdot w$ одиниць довжини стіни. Отже, для проміжків залишається $s = l - n \cdot w$ одиниць.

Між сусідніми картинами є $n - 1$ проміжків, а біля країв стіни — ще два. Таким чином, усього маємо $n + 1$ однакових проміжків. Довжина кожного з них має дорівнювати $s/(n + 1)$.

За умовою $n \cdot w \leqslant l$, тому s невід'ємне. Залишається перевірити, чи ділиться s на $n + 1$ без остачі. Якщо так, усі проміжки матимуть однакову невід'ємну цілу довжину, тому виведемо YES. Інакше виведемо NO.

Випадок $s = 0$ також підходить: усі проміжки мають нульову довжину, тобто картини торкаються одна одної та країв стіни.

Складність алгоритму — $O(1)$ за часом та пам'яттю.

Задача D. Дубль

Автор: Володимир Балачевцев
Розробник: Тарас Деркач
Розбір підготовано: Андрій Холод

Перевірятимемо випадки в порядку зменшення кількості балів.

Якщо $a = b = c$, усі три числа однакові, тому відповіддю буде 30.

Інакше, якщо виконується хоча б одна з рівностей $a = b$, $a = c$ або $b = c$, однаковими є рівно два числа. У цьому випадку відповідь — 20.

Залишився випадок, коли всі числа різні. Помітимо, що отримання одного числа відніманням двох інших можна записати як рівність із додаванням. Наприклад, $a - b = c$ рівносильне $b + c = a$. Тому достатньо перевірити три рівності: - $a + b = c$; - $a + c = b$; - $b + c = a$.

Якщо хоча б одна з них виконується, відповіддю буде 10, інакше — 0.

Порядок перевірок важливий: наприклад, для чисел 1, 1, 2 виконується рівність $1 + 1 = 2$, але за правилами потрібно нарахувати 20 балів за два однакові числа.

Складність алгоритму — $O(1)$ за часом та пам'яттю.

Задача E. Двоколірна карта

Автор: Анна Ярош
Розробник: Тарас Деркач
Розбір підготовано: Андрій Холод

Пофарбуємо перший рядок повністю в червоний колір, а останній — повністю в синій. У кожному з решти рядків чергуватимемо кольори: клітинки непарних стовпців будуть червоними, а парних — синіми.

Перевіримо вимоги задачі.

Кожен стовпець містить червону клітинку в першому рядку та синю в останньому. Отже, в кожному стовпці присутні обидва кольори.

Усі червоні клітинки утворюють одну зв'язну область. Справді, червоні клітинки кожного непарного стовпця з'єднані між собою по вертикалі та доходять до першого рядка. Перший рядок з'єднує всі ці стовпці.

Аналогічно, сині клітинки парних стовпців з'єднані по вертикалі та доходять до останнього рядка, який об'єднує їх в одну зв'язну область.

Залишилося перевірити квадрати 2×2. Якщо такий квадрат містить рядок, відмінний від першого та останнього, то дві сусідні клітинки цього рядка мають різні кольори. Тому квадрат містить обидва кольори. При $n = 2$ єдиний квадрат складається з червоного та синього рядків, тож вимога також виконується.

Таким чином, побудована карта задовольняє всі умови.

Складність алгоритму — $O(n^2)$ за часом. Якщо виводити карту по одному рядку, достатньо $O(n)$ додаткової пам'яті.

Задача F. Копії без оригіналів

Автор: Андрій Холод
Розробник: Андрій Холод
Розбір підготовано: Андрій Холод

Помітимо, що кожна операція лише копіює вміст картки. Тому для кожної картки можна визначити, з якої початкової картки походить її кінцевий вміст.

Позначимо через p_i номер такої початкової картки. Спочатку $p_i = i$. Під час операції (x, y) виконаємо присвоєння $p_y = p_x$, оскільки тепер вміст обох карток походить з одного джерела. Обробимо таким чином усі операції в заданому порядку.

Якщо після цього $p_i = p_j$, на картках i та j мають бути однакові числа: обидві містять копію початкового вмісту тієї самої картки. Тому якщо $a_i \neq a_j$, отримати заданий кінцевий вміст неможливо, і відповіддю буде -1.

Інакше відповідь дорівнює кількості різних значень серед $p_1, p_2, \ldots, p_n$. Справді, кожна з відповідних початкових карток мусила бути непорожньою, адже її вміст присутній наприкінці. Водночас достатньо записати на кожній із них потрібне число, а решту карток залишити порожніми. Після виконання операцій отримаємо саме задані числа.

Для перевірки заведемо масив b, спочатку заповнений нулями. Переберемо всі кінцеві картки i. Якщо $b_{p_i} = 0$, присвоїмо $b_{p_i} = a_i$ та збільшимо відповідь на один. Інакше перевіримо, чи $b_{p_i} = a_i$. Якщо рівність не виконується, виведемо -1.

Однакові числа на картках із різними джерелами враховуємо окремо: кожне з цих джерел мало бути непорожнім.

Складність алгоритму — $O(n + m)$ за часом та $O(n)$ за пам'яттю.

Задача G. Спільна остача: видалення відрізка

Автор: Тарас Деркач
Розробник: Андрій Холод
Розбір підготовано: Андрій Холод

Помітимо, що всі числа мають однакову остачу від ділення на d тоді й лише тоді, коли різниця між кожними двома сусідніми числами ділиться на d. Отже, для фіксованого відрізка видалення найбільше можливе d дорівнює найбільшому спільному дільнику сусідніх різниць у послідовності, що залишилася.

Нехай ми видаляємо елементи з індексами від l до $r = l + k - 1$. Після видалення залишаються префікс $x_1, \ldots, x_{l-1}$ та суфікс $x_{r+1}, \ldots, x_n$. Сусідні різниці всередині цих частин не змінюються. Якщо обидві частини непорожні, з'являється ще одна сусідня пара: x_{l-1} та x_{r+1}. Її різниця дорівнює $x_{r+1} - x_{l-1}$.

Щоб швидко знаходити відповідь для кожного відрізка, попередньо обчислимо два масиви: - p_i — найбільший спільний дільник сусідніх різниць серед перших i елементів; - s_i — найбільший спільний дільник сусідніх різниць серед елементів з індексами від i до n.

Покладемо $p_0 = p_1 = 0$ та $s_n = s_{n+1} = 0$. Решту значень обчислимо послідовно: $p_i = \gcd(p_{i-1}, x_i - x_{i-1})$ для $i = 2, \ldots, n$; $s_i = \gcd(s_{i+1}, x_{i+1} - x_i)$ для $i = n-1, \ldots, 1$.

Тепер переберемо всі можливі ліві межі l. Для кожної з них знайдемо $r = l + k - 1$ та спочатку обчислимо $g = \gcd(p_{l-1}, s_{r+1})$. Якщо $l > 1$ та $r < n$, додатково замінимо g на $\gcd(g, x_{r+1} - x_{l-1})$.

Таким чином, g враховує всі сусідні різниці після видалення, тому є найбільшим допустимим значенням d для цього відрізка. Відповіддю буде найбільше отримане g.

Оскільки залишаються принаймні два різні числа, хоча б одна різниця додатна, тож отриманий найбільший спільний дільник також додатний.

Алгоритм виконує $O(n)$ обчислень найбільшого спільного дільника. Складність за часом — $O(n \log M)$, де $M = x_n - x_1$, за пам'яттю — $O(n)$.

Задача H. Краплі на таці

Автор: Анна Ярош
Розробник: Андрій Холод
Розбір підготовано: Андрій Холод

Розглянемо рух крапель по горизонталі та вертикалі окремо. Горизонтальні команди не змінюють рядки крапель, а вертикальні — їхні стовпці. Тому всі краплі зберуться в одній клітинці тоді й лише тоді, коли вони опиняться в одному стовпці та в одному рядку.

Спочатку розглянемо лише горизонтальний рух. Замінимо команду R на $+1$, L на -1, а U та D — на 0. Нехай $q_0 = 0$, а q_i — сума перших i значень у вибраному фрагменті.

Позначимо через t різницю між найбільшим і найменшим значеннями серед цих префіксних сум. Тоді відстань між крайньою лівою та крайньою правою краплями після виконання фрагмента дорівнює $\max(0, w - 1 - t)$.

Пояснимо це спостереження. Спочатку крайні краплі розділяють $w - 1$ клітинок. Поки краплі не зібралися в одному стовпці, кожне збільшення максимуму або зменшення мінімуму префіксних сум означає рух до відповідної межі: одна крайня крапля залишається на місці, а інша наближається до неї. Відстань зменшується на один. Решта рухів лише зсувають обидві крайні краплі, зберігаючи відстань між ними. Після злиття краплі вже не розділяються.

Отже, для збирання в одному стовпці необхідно й достатньо, щоб розмах горизонтальних префіксних сум був не меншим за $w - 1$. Аналогічно, для збирання в одному рядку розмах вертикальних префіксних сум має бути не меншим за $h - 1$.

Обчислимо префіксні суми для всього рядка команд: - $x_0 = 0$, а x_i враховує команди R як $+1$, L як -1; - $y_0 = 0$, а y_i враховує команди D як $+1$, U як -1.

Для фрагмента з позиції l до позиції r потрібно розглядати значення з індексами від $l - 1$ до r включно. Віднімання початкових значень x_{l-1} та y_{l-1} не змінює розмахів. Тому фрагмент підходить, якщо одночасно виконуються дві умови: - різниця максимуму та мінімуму x на відрізку $[l - 1, r]$ не менша за $w - 1$; - різниця максимуму та мінімуму y на цьому відрізку не менша за $h - 1$.

Побудуємо для масивів x та y розріджені таблиці (sparse table), які дозволяють знаходити мінімум і максимум на відрізку за $O(1)$.

Тепер скористаємося бінарним пошуком за довжиною відповіді. Для перевірки довжини k перебе-ремо всі фрагменти такої довжини та перевіримо наведені умови. Одна перевірка займає $O(m)$ часу.

При розширенні фрагмента розмахи префіксних сум не зменшуються. Тому якщо існує придатний фрагмент довжини k, то існує придатний фрагмент кожної більшої довжини до m. Це дозволяє знайти найменшу придатну довжину бінарним пошуком. Якщо навіть увесь рядок не підходить, виведемо -1.

Побудова розріджених таблиць займає $O(m \log m)$ часу та пам'яті. Бінарний пошук виконує $O(\log m)$ перевірок по $O(m)$ часу. Загальна складність — $O(m \log m)$ за часом та пам'яттю.