Задача A. Тест

Максимальна кількість помилок дорівнює $b - c$.

Часова складність: $O(1)$.

Задача B. Середнє арифметичне

Для розв'язання даної задачі трансформуємо наше рівняння та виразимо з нього b:

$$\frac{a+b}{2} = c$$

$$a + b = 2 \cdot c$$

$$b = 2 \cdot c - a$$

Тому ми за даними a, c можемо знайти невідоме b, підставивши їх у формулу. Складність рішення $O(1)$.

Задача C. Гітара

Автор, розробник, автор розбору: Тимур Дегтеарі

Ця задача вимагає прямого моделювання умов, описаних у заявці. Ми просто проходимо через кожен момент часу і перевіряємо, чи є позиції пальців можливими.

Для кожної секунди i від 1 до n виконайте наступну перевірку:

1. Серед трьох значень $s_{1,i}, s_{2,i}, s_{3,i}$ визначте всі позитивні значення (тобто значення, більші за 0). Нехай x позначає мінімум цих позитивних значень, а y - максимум.
2. Якщо в момент часу i немає позитивних значень (всі струни або не грають, або грають відкрито), тоді пан Т може тривіально впоратися з цим моментом—перейдіть до наступної секунди.
3. В іншому випадку перевірте, чи задовольняється вимога розтягування: $y - x \leqslant 3$. Якщо ця умова виконується, продовжте до наступної секунди. Якщо вона не виконується, виведіть "Ні" разом з i як першим моментом, коли розтягування стає неможливим.

Якщо всі n секунд проходять перевірку, виведіть "Так"—пан Т може зіграти всю пісню.

Часова складність: $O(n)$, оскільки ми виконуємо перевірку постійного часу для кожної з n секунд.

Задача D. EVO проти STI

Автор, розробник, автор розбору: Тимур Дегтеарі

Ключове розуміння полягає в переформулюванні умов виграшу в термінах масивів різниць, а потім спостереженні, що оптимальний вибір трас відповідає префіксам відсортованих масивів.

Визначимо два допоміжні масиви:

- $c_i = e_i - s_i$ (позитивний, коли STI Guy швидший на трасі i)
- $d_i = s_i - e_i$ (позитивний, коли EVO Guy швидший на трасі i)

Нехай c' буде масивом c, відсортованим у невисхідному порядку, а d' буде масивом d, відсортованим у невисхідному порядку.

Оптимальний вибір трас для EVO Guy, щоб виграти, складається з **префікса** c'. Аналогічно, оптимальний вибір для STI Guy, щоб виграти, складається з **префікса** d'.

Для того, щоб EVO Guy виграв, нам потрібно:

$$\sum_{\text{трасу } i \text{ обрано}} c_i < 0$$

Припустимо, що S — найбільша підмножина трас, така що $\sum_{c_i \in S} c_i < 0$, і S не відповідає префіксу c'. Оскільки c' відсортований у невисхідному порядку, заміна S на префікс c' такої ж величини дасть суму, яка не перевищує початкову суму. Отже, префікс принаймні такий же хороший, як (і часто кращий за) будь-яку іншу підмножину такої ж величини.

За симетричним аргументом, те ж саме стосується перемоги STI Guy, використовуючи масив d'.

1. Обчисліть масиви c і d з вхідних даних.
2. Відсортуйте обидва масиви у невисхідному порядку.
3. Для кожного масиву знайдіть довжину найдовшого префікса, сума якого строго менша за 0.
4. Ці довжини дають R_E (для EVO Guy, використовуючи масив c') і R_S (для STI Guy, використовуючи масив d').

Примітка: Якщо жоден префікс не має негативної суми, відповідь дорівнює 0, що означає, що гонщик не може виграти незалежно від вибору трас.

Часова складність: $O(n \log n)$ через сортування. Обчислення префіксної суми — $O(n)$.

Задача E. Шифр

Автор, розробник, автор розбору: Тимур Дегтеарі

У цій задачі ми можемо скористатися малими обмеженнями на довжину імені m (не більше 10) та довжину повідомлення n (не більше 1000). Це дозволяє нам з комфортом використовувати рішення з складністю $O(n \cdot m)$ або навіть $O(n \cdot m^2)$.

Ми перебираємо всі можливі початкові позиції, де закодоване ім'я може з'явитися в повідомленні. Зокрема, ми пробуємо кожну позицію i від 1 до $n - m + 1$ як потенційний початковий індекс.

Для кожної кандидатної позиції i ми розглядаємо підрядок $a[i], a[i+1], \ldots, a[i+m-1]$ і намагаємося зіставити кожне число з відповідною літерою з імені. Під час цього процесу ми перевіряємо, що:

- Кожна літера в імені відповідає точно одному числу.
- Жодні дві різні літери не відповідають одному й тому ж числу (тобто, відображення є ін'єктивним).

Якщо ця властивість виконується, ми знайшли дійсний шифр. Потім ми довільно призначаємо залишкові невикористані числа літерам, які не з'являються в імені (оскільки ці призначення не вплинуть на вже зіставлений підрядок) і виводимо результат.

Якщо властивість не виконується для всіх позицій, ми повідомляємо, що жоден дійсний шифр не існує.

Часова складність: $O(n \cdot m)$, де n — довжина зашифрованого повідомлення, а m — довжина імені.

Задача F. Пінгвіни

Автор: Тимур Дегтярі
Роздробник: Тимур Дегтярі
Розбір підготовано: Тимур Дегтярі, Костянтин Денисов

Ми стверджуємо, що відповідь $\leqslant 3$. Дійсно, розглянемо наступний жадібний підхід:

- проходимо через сім'ї, призначаємо поточну сім'ю команді найменшого розміру. Розглянемо, як змінюються найменший і найбільший розміри команди:
- нехай x буде мінімумом, а y — максимумом. Припустимо, що $y - x \leqslant 3$. Тоді після призначення поточної сім'ї найменшій команді новий мінімум становить принаймні x, а новий максимум — $\max(x + 3, y) \leqslant x + 3$. Отже, ця властивість зберігається протягом усього процесу призначення.

Отже, відповідь завжди одна з $0, 1, 2, 3$.

Нехай x — це кількість сімей з 3 пінгвінами, y — з 2 пінгвінами, а z — з 1 пінгвіном. Нехай $S = 3 \cdot x + 2 \cdot y + z$.

Ствердження: якщо $(x \ \% \ k) > 0$ і $(x \ \% \ k) + y + z < k$, відповідь дорівнює 3.

Нехай $t = \lfloor \frac{x}{3} \rfloor$. Спочатку зауважимо, що в оптимальному призначенні принаймні одна команда матиме $3t$ пінгвінів. Оскільки $(x \ \% \ k) + y + z < k$, за принципом Діріхле принаймні одна команда міститиме лише сім'ї з 3 пінгвінами. Якщо ця команда має розмір $3t + 3$, тоді, щоб відповідь не була 3, всі інші команди повинні мати розміри $\geqslant 3t + 1$, тому кожна з них повинна містити або принаймні одну сім'ю розміру 1, або 2, або принаймні $t+1$ сімей розміру 3, отже, $(x \ \% \ k) + y + z \geqslant k - 1$. Водночас команда, яку ми розглядаємо, також має $t + 1$ сімей з 3, отже, $(x \ \% \ k) + y + z \geqslant k$, що є суперечністю.

Якщо ця команда має розмір не більше $3t - 3$, тоді за принципом Діріхле найбільша команда має розмір принаймні $\frac{S}{k} > 3t$, що дає відповідь > 3, суперечність. Отже, завжди є принаймні одна команда з розміром $3t$.

Більше того, це єдиний випадок, коли відповідь дорівнює 3. Припустимо, що $(x \ \% \ k) + y + z \geqslant k$. Спочатку рівномірно розподіліть сім'ї з 3 між командами, так що $(x \ \% \ k)$ команд мають $3t+3$ пінгвінів, а решта команди мають $3t$ пінгвінів. Тепер розподіліть достатньо сімей з 1 і 2 пінгвінами, щоб усі команди мали розміри між $3t+1$ і $3t+3$. Оскільки залишилися лише сім'ї з 1 і 2, а різниця в розмірах становить не більше 2, ми можемо зберігати її не більше 2, жадібно додаючи нову сім'ю до найменшої команди, отже, оптимальна відповідь не перевищує 2.

Крім того, якщо $(x \ \% \ k) + y + z < k$, але $(x \ \% \ k) = 0$, очевидно, що відповідь дорівнює 2, якщо $y > 0$; 1, якщо $y = 0, z > 0$, і 0, якщо $y = z = 0$.

Отже, тепер нам потрібно лише розрізнити, чи відповідь дорівнює $0, 1$ або 2.

Цілі для відповідей 0 і 1.

Перевірка $ans = 0$**.** Якщо $S \bmod k \neq 0$, тоді $ans = 0$ неможливо. В іншому випадку всі k команд повинні мати однаковий розмір $T = S/k$.

Перевірка $ans = 1$**.** Нехай $L = \lfloor S/k \rfloor$, $U = L + 1$, а $x = S - kL$. Тоді точно x команд повинні мати розмір U, а решта $k - x$ команд повинні мати розмір L.

В обох випадках ми запускаємо ту ж процедуру перевірки **TryBuild**, яка намагається побудувати команди з необхідними цільовими розмірами, використовуючи доступні сім'ї.

Жадібна процедура перевірки TryBuild.

Нам дано мультимножину цільових розмірів команд (або всі T, або $k - x$ копій L і x копій U). Ми також підтримуємо залишкові кількості (c_1, c_2, c_3) і намагаємося побудувати кожну команду незалежно.

Перший прохід (жадібний на основі шаблону). Для кожного цільового розміру t (обробляємо цілі в невисхідному порядку) ми намагаємося заповнити одну команду з загальною сумою точно t за наступними пріоритетами:

1. Якщо t непарне, розмістіть одну сім'ю розміру 3, якщо це можливо; в іншому випадку розмістіть одну сім'ю розміру 1. Зменшіть t на 3 або 1 відповідно. Якщо жоден з них недоступний, цей прохід зазнає невдачі.
2. Поки це можливо, розміщуйте пари $(3, 3)$ (віднімемо 6 від t), поки $t \geqslant 6$.
3. Потім, поки це можливо, розміщуйте пари $(3, 1)$ (віднімемо 4), але лише після завершення етапу $(3, 3)$.
4. Потім розмістіть якомога більше сімей з 2 (віднімемо 2 щоразу).
5. Нарешті, розмістіть пари $(1, 1)$ (віднімемо 2).

Якщо нам вдасться зменшити кожну цільову t до 0, перший прохід успішний; в іншому випадку він зазнає невдачі.

Другий прохід (найбільша сім'я в найменшу команду). Якщо перший прохід зазнає невдачі, ми починаємо з початкових кількостей і запускаємо ще один жадібний:

- Зберігайте k команд з їхніми поточними сумами (спочатку всі 0) у структурі, яка може повертати команду з найменшою сумою.
- Постійно беріть найбільшу залишкову сім'ю (надаючи перевагу розміру 3, потім 2, потім 1) і додавайте її до найменшої команди.

Після того, як усі сім'ї призначені, ми сортуємо отримані суми команд і порівнюємо їх з необхідною цільовою мультимножиною. Якщо вони збігаються, другий прохід успішний.

Складність.

Підрахунок c_1, c_2, c_3 — це $O(n)$. Другий жадібний прохід можна реалізувати за допомогою пріоритетної черги над k командами і виконується за $O(n \log k)$. Загальна складність на тестовий випадок становить $O(n \log k)$.

Задача G. Множення запитів

Автор, розробник, автор розбору: Тимур Дегтеарі

Цю задачу можна ефективно вирішити, використовуючи два ключові спостереження.

1. **Монотонність значень:** Оскільки $x_i \geqslant 1$, як тільки $a_j \geqslant Y$ після i-го оновлення, воно залишиться таким після всіх наступних оновлень. Тому достатньо визначити, для кожного індексу j, перше оновлення, після якого $a_j \geqslant Y$. Якщо ми позначимо номер цього оновлення як $f(j)$, то остаточна відповідь буде простою: $b_j = q - f(j) + 1$.
2. **Обмежена кількість значущих оновлень:** Оновлення з $x_i = 1$ взагалі не змінюють масив a, тому ми можемо їх пропустити. Для оновлень з $x_i \geqslant 2$, кожен елемент, на який це впливає, принаймні подвоює своє значення. Відповідно, будь-який елемент може бути суттєво змінений максимум $\log_2(Y)$ таких оновлень, перш ніж досягти або перевищити Y.

Ми підтримуємо множину S, що містить усі індекси j, для яких $a_j < Y$. При обробці оновлення з $x_i \geqslant 2$:

- Перебираємо всі індекси $j \in S$, які лежать у діапазоні $[l_i, r_i]$.
- Множимо a_j на x_i.
- Якщо $a_j \geqslant Y$ після множення, записуємо $f(j) = i$ і видаляємо j з S (оскільки всі майбутні оновлення збережуть $a_j \geqslant Y$).

На перший погляд, цей підхід може здаватися $O(q \cdot n)$. Однак, згідно з спостереженням (2), кожен елемент може бути відвіданий максимум $\log_2(Y)$ разів протягом усіх оновлень (оскільки кожне відвідування принаймні подвоює його значення). Використовуючи збалансовану структуру даних (таку як `std::set`) для ефективного запиту елементів у діапазоні, загальна складність стає:

$$O\big(q \cdot \log(n) + n \cdot \log_2(Y)\big)$$

Це достатньо ефективно з урахуванням обмежень задачі.