Оглавление

Коллекция фотографий

У Егора есть a больших и b маленьких фотографий. В одном альбоме n страниц. На страницу можно поместить либо одну любую фотографию, либо две маленькие. Найти минимальное количество альбомов.

Каждая страница вмещает максимум 2 маленькие фотографии или 1 большую. Переведём все фотографии в эквивалент маленьких: одна большая занимает место двух маленьких. Общее количество "маленьких единиц"равно $2a + b$.

Одна страница вмещает 2 такие единицы. В одном альбоме n страниц, значит, вместимость альбома в единицах равна $2n$.

Количество альбомов: $\left\lceil \frac{2a+b}{2n} \right\rceil$.

Сумма расстояний

Дан массив целых чисел $a_1, a_2, \ldots, a_n$. Для каждого индекса i определим сумму расстояний до всех остальных элементов: $S_i = \sum_{j=1}^{n} |a_i - a_j|$. Требуется найти индекс i, для которого S_i минимальна (вывести значение a_i). Гарантируется, что n нечетно.

Для массива чисел сумма абсолютных отклонений от некоторой точки минимальна, когда эта точка является медианой. При нечетном n медиана — это элемент, стоящий на позиции $(n + 1)/2$ после сортировки. Таким образом, нужно отсортировать массив и взять средний элемент.

Саша и разнообразные числа - легкая версия

Натуральное число называется разнообразным, если все его цифры различны. Для заданного n найдите наименьшее разнообразное число, большее n. Если такого числа не существует, выведите -1. Здесь $n \leq 10^6$.

Будем последовательно увеличивать n на 1 и проверять, являются ли все цифры числа различными. Для проверки можно перевести число в строку и сравнить длину строки с количеством уникальных символов (например, через множество). Так как $n \leq 10^6$, а следующее разнообразное число не может быть слишком далеко, такой перебор работает быстро.

Саша и разнообразные числа - сложная версия

Натуральное число называется разнообразным, если все его цифры различны. Для заданного n найдите наименьшее разнообразное число, большее n. Если такого числа не существует, выведите -1. Здесь $n \leq 10^9$.

Все разнообразные числа можно сгенерировать заранее. Количество таких чисел равно сумме $\sum_{k=1}^{10} P(10, k)$, где $P(10, k)$ — число размещений без повторений из 10 по k. Это около $9 \cdot 9! + 8 \cdot 8! + \ldots$ — примерно несколько миллионов, что допустимо.

Генерируем все подмножества цифр, для каждого подмножества перебираем все перестановки, отбрасывая варианты с ведущим нулём. Сохраняем полученные числа в массив и сортируем.

Остается для каждого запроса находить верхнюю границу (первое число, большее n) с помощью бинарного поиска. Если такого числа нет, вывести -1.

Беспорядок в библиотеке - легкая версия

На полке стоят n книг, каждая из трёх типов: '1', '2', '3'. За одну операцию можно выбрать три книги с номерами $i < j < k$ такие, что i-я книга — '1', j-я — '2', k-я — '3', и забрать их. Книги удаляются, порядок оставшихся сохраняется. Найти максимальное количество книг, которое можно забрать. Здесь $n \leq 10^3$.

Жадно ищем первую слева книгу '1', затем после неё первую '2', затем после неё первую '3'. Как только нашли тройку, помечаем эти три позиции как забранные и повторяем процесс сначала. Такой жадный подход даёт максимальное количество троек, так как использование самой левой возможной тройки не блокирует другие тройки.

Беспорядок в библиотеке - сложная версия

На полке стоят n книг, каждая из трёх типов: '1', '2', '3'. За одну операцию можно выбрать три книги с номерами $i < j < k$ такие, что i-я книга — '1', j-я — '2', k-я — '3', и забрать их. Книги удаляются, порядок оставшихся сохраняется. Найти максимальное количество книг, которое можно забрать. Здесь $n \leq 10^6$.

Пройдём по строке слева направо, поддерживая два счётчика: количество доступных книг '1' и количество пар "1-2 ожидающих '3'. При встрече '1' увеличиваем счётчик единиц. При встрече '2', если есть хотя бы одна единица, используем её для формирования пары: уменьшаем счётчик единиц и увеличиваем счётчик пар. При встрече '3', если есть хотя бы одна пара, завершаем тройку: уменьшаем счётчик пар и увеличиваем ответ на 3. Такой жадный подход позволяет собрать максимальное количество троек за один проход $O(n)$.

Очередная задача про хорошие строки - легкая версия

Даны n строк из цифр. Строка называется хорошей, если её цифры идут в неубывающем порядке. Посчитать количество пар (i, j), $i < j$, таких что конкатенация $s_i + s_j$ является хорошей. Здесь $n \leq 10^3$.

Чтобы конкатенация $s_i + s_j$ была хорошей, необходимо, чтобы строка s_i сама по себе была хорошей, строка s_j была хорошей, и последняя цифра s_i была не больше первой цифры s_j. Можно запомнить для каждой хорошей строки её первую и последнюю цифру: $(first, last)$. Затем перебираем все пары индексов и проверяем условие $last_i \leq first_j$.

Очередная задача про хорошие строки - сложная версия

Даны n строк из цифр. Строка называется хорошей, если её цифры идут в неубывающем порядке. Посчитать количество пар (i, j), $i < j$, таких что конкатенация $s_i + s_j$ является хорошей. Здесь $n \leq 10^6$.

Чтобы конкатенация $s_i + s_j$ была хорошей, необходимо, чтобы строка s_i сама по себе была хорошей, строка s_j была хорошей, и последняя цифра s_i была не больше первой цифры s_j. Идём по строкам в порядке их поступления - для каждой хорошей строки s_j с первой цифрой $first$, добавляем к ответу количество ранее встреченных хороших строк s_i, у которых последняя цифра $\leq first$. Затем запоминаем s_j по её последней цифре (увеличиваем счётчик $end[last]$). Таким образом, обрабатываем строки по одной, поддерживая массив из 10 счётчиков. Сложность $O(n \cdot 10)$.

Очередная задача про игру с камнями

Даны n кучек. Из i-й кучки нужно взять от l_i до r_i камней включительно. Всего нужно взять ровно s камней. Для каждой кучки i определите, сколько значений x ($l_i \le x \le r_i$) можно выбрать так, чтобы существовал способ добрать камни из остальных кучек, удовлетворяющий условиям.

Обозначим $L = \sum l_i$, $R = \sum r_i$ — минимальная и максимальная суммарная возможность.

Для фиксированной кучки i обозначим $L' = L - l_i$, $R' = R - r_i$ — минимальная и максимальная сумма в остальных кучках. Выбрав x камней из i-й кучки, остатку $s - x$ нужно лежать в отрезке $[L', R']$. Отсюда:

$$x \in [s - R',\ s - L'] \cap [l_i,\ r_i]$$

То есть количество подходящих x равно $\max(0,\ \min(r_i,\ s - L') - \max(l_i,\ s - R') + 1)$.

Автомагистраль

Дан массив h из n чисел и q запросов: для отрезка $[l, r]$ и числа k посчитать количество $i \in [l, r]$, таких что $k \mid \gcd(h_i, i)$.

Условие $k \mid \gcd(h_i, i)$ означает, что k делит и i, и h_i одновременно. Таким образом, для фиксированного k нас интересуют только те позиции i, которые делятся на k, и при этом h_i тоже делится на k. Предподсчитаем для каждого k от 1 до 1000 (так как $k \le 1000$) отсортированный список всех таких индексов $i = k, 2k, 3k, \cdots \le n$, что $h_i \% k == 0$ (сохраним в $I[k]$).

Тогда на запрос (l, r, k) ответ — количество элементов в $I[k]$, попадающих в отрезок $[l, r]$. Это можно найти с помощью бинарного поиска:

$$\text{ans} = \text{upper\_bound}(I[k], r) - \text{lower\_bound}(I[k], l)$$

Сладкая жизнь

Имеется n предметов, у каждого стоимость c_i и удовольствие h_i. Бюджет m. Нужно выбрать набор предметов с максимальной суммой удовольствия, не превышая бюджет.

Классическая задача о рюкзаке. Пусть $dp[j]$ — максимальное удовольствие при суммарной стоимости ровно j. Изначально $dp[0] = 0$, остальные $-\infty$. Для каждого предмета обновляем dp в порядке убывания j, чтобы каждый предмет использовался не более одного раза:

$$dp[j + c_i] = \max(dp[j + c_i],\ dp[j] + h_i)$$

После обработки всех предметов ответ — $\max_{0 \le j \le m} dp[j]$.

КотТВ

Дан неориентированный взвешенный граф из n узлов и m рёбер. Требуется найти длину кратчайшего пути от s до f. Если пути не существует, вывести "Meow meow meow...".

Запускаем алгоритм Дейкстры из вершины s, так как все веса рёбер. Если после работы алгоритма $dist[f]$ остался равным бесконечности, пути нет. Иначе выводим $dist[f]$.