Оглавление

Чтение книги

В книге n страниц, на каждой m строк. Найти номер страницы, на которой находится k-я строка (нумерация строк сквозная).

Страницы нумеруются с 1. На первой странице строки с 1 по m, на второй — с $m+1$ по $2m$, и так далее. Номер страницы для k-й строки вычисляется как $\lceil k/m \rceil$, то есть $(k+m-1)/m$.

Цифровая сумма

Назовём число интересным, если сумма его цифр делится на 3. Найти сколько существует k-значных чисел (без ведущих нулей) с таким свойством.

Среди всех k-значных чисел ровно треть имеет сумму цифр, делящуюся на 3, поэтому количество интересных чисел равно $9 \cdot 10^{k-1}/3 = 3 \cdot 10^{k-1}$.

Так как k может быть очень большим (до 10^7), заранее вычислим массив степеней десятки $pow10[i] = 10^i \bmod M$ для всех i от 0 до 10^7 (максимальное возможное k). Это делается одним циклом: $pow10[0] = 1$, $pow10[i] = (pow10[i-1] \cdot 10) \bmod M$. Тогда ответ для каждого запроса k равен $3 \cdot pow10[k-1] \bmod M$.

Игра в поедание

За круглым столом сидят n игроков, у каждого a_i блюд. Ходят по часовой стрелке, съедая по одному блюду (без блюд — пропуск). Победитель — съевший последнее блюдо. Сколько игроков могут стать победителями при оптимальном выборе стартового?

Победителем может стать только игрок с максимальным количеством блюд, так как он будет есть чаще остальных и в итоге съест последнее блюдо. Игроки с меньшим количеством блюд не могут победить. Ответ — количество игроков, у которых a_i равно максимуму.

Три кота

Даны n чисел — размеры предметов под покрывалами. Три котёнка спрятались под тремя покрывалами. Котята имеют разный рост, и ни один размер коробки не совпадает с ростом котёнка. Нужно найти номера покрывал с котятами, если это можно сделать однозначно.

Заметим, что если два предмета имеют одинаковый размер, то ни под одним из них не мог спрятаться котёнок. Следовательно, котята могут находиться только под покрывалами с уникальными размерами. Таких позиций может быть несколько. Если их ровно три, то это и есть искомые номера. Если их больше или меньше трёх, то ответ неоднозначен (или невозможен), и выводится "Meow meow meow...".

Прикол - очень легкая версия

Для двух целых положительных чисел l и r определим $f(l, r)$ как количество упорядоченных пар натуральных чисел (a, b) таких, что $l \le a + b \le r$ и $\gcd(a, b)$ — простое число. Дано $q = 1$ запрос, $l, r \le 1000$.

Перебираем все пары (a, b) от 1 до r. Для каждой пары вычисляем $\gcd(a, b)$ и проверяем, является ли он простым. Проверка простоты числа x выполняется перебором делителей от 2 до $\sqrt{x}$. Сложность алгоритма: $O(r^2 \cdot (\log r + \sqrt{r}))$.

Прикол - легкая версия

Для двух целых положительных чисел l и r определим $f(l, r)$ как количество упорядоченных пар натуральных чисел (a, b) таких, что $l \le a + b \le r$ и $\gcd(a, b)$ — простое число. Дано $q = 1$ запрос, $l, r \le 10^4$.

Предвычисляем все простые числа до 10^4 с помощью решета Эратосфена. Затем перебираем все пары (a, b) от 1 до r, проверяя условие на сумму и простоту $\gcd(a, b)$. Сложность алгоритма: $O(r^2 \log r)$.

Прикол - средняя версия

Для двух целых положительных чисел l и r определим $f(l, r)$ как количество упорядоченных пар натуральных чисел (a, b) таких, что $l \le a + b \le r$ и $\gcd(a, b)$ — простое число. Дано $q \le 10^6$ запросов, $l, r \le 10^4$.

Заметим, что условие зависит только от суммы $s = a+b$. Для фиксированной суммы s количество пар (a, b) с $a, b \ge 1$, $a + b = s$, у которых $\gcd(a, b)$ простое, можно предпосчитать. Для каждого s перебираем a от 1 до $s-1$, находим $b = s - a$, вычисляем $\gcd(a, b)$ и проверяем простоту. Полученное количество пар для суммы s обозначим $cnt[s]$.

Затем построим массив префиксных сумм $pref[t] = \sum_{s=1}^{t} cnt[s]$. Тогда количество пар с суммой в отрезке $[l, r]$ равно $pref[r] - pref[l-1]$. Это и есть ответ на запрос. Сложность алгоритма: $O(r^2 \log r + q)$.

Победа бездействием

Дан массив a. За ход игрок выбирает непустую подпоследовательность, в которой хотя бы один элемент не равен gcd всех её элементов, и заменяет все элементы на этот gcd. Кто не может сделать ход — побеждает. Кто выигрывает при оптимальной игре?

Заметим, что если в массиве есть элемент, равный общему gcd всего массива d, то его можно оставить неизменным, а все остальные элементы со временем станут равны d. Игра сводится к тому, кто сделает последний ход. Если количество элементов, равных d, не равно 0 и не равно $n-1$, то выигрывает Алиса. Если их $n-1$, то выигрывает Боб.

Если же их 0, то нужно проверить, можно ли за один ход сделать так, чтобы появился элемент, равный 1 (если это возможно, то Алиса выигрывает, иначе Боб). Для этого вычислим префиксные gcd ($pref[i] = \gcd(a_1, \ldots, a_i)$) и суффиксные gcd ($suf[i] = \gcd(a_i, \ldots, a_n)$). Если существует позиция i, такая что $\gcd(pref[i-1], suf[i+1]) = 1$, то Алиса может выбрать подпоследовательность из всех элементов, кроме a_i, и после замены a_i станет равным 1. Тогда Алиса выигрывает. Иначе выигрывает Боб.

Хорошие индексы - легкая версия

Дан массив a длины n. Индекс i ($1 \le i < n$) разделяет массив на левую и правую часть. Назовём i хорошим, если можно изменить не более одного элемента массива (на любое положительное число) так, чтобы gcd левой части стал равен gcd правой. Изменение делается независимо для каждого i. Найти количество хороших индексов. Здесь $n \le 5000$.

Вычислим префиксные gcd: $pref[i] = \gcd(a_1, \ldots, a_i)$ и суффиксные: $suf[i] = \gcd(a_i, \ldots, a_n)$. Для фиксированного разделителя i левый gcd равен $pref[i]$, правый — $suf[i+1]$. Чтобы сделать их равными, можно изменить один элемент в любой позиции. Изменение элемента в левой части не влияет на правый gcd, и наоборот. Условие: либо в левой части есть не более одного элемента, не делящегося на $suf[i+1]$, либо в правой части есть не более одного элемента, не делящегося на $pref[i]$. Проверяем это для каждого i за $O(n)$ и получаем ответ.

Хорошие индексы - сложная версия

Дан массив a длины n. Индекс i ($1 \le i < n$) разделяет массив на левую и правую часть. Назовём i хорошим, если можно изменить не более одного элемента массива (на любое положительное число) так, чтобы gcd левой части стал равен gcd правой. Изменение делается независимо для каждого i. Найти количество хороших индексов. Здесь $n \le 2 \cdot 10^5$.

Воспользуемся идеей легкой версии, но для быстрой проверки предподсчитаем для каждого возможного значения gcd (их немного - порядка логарифма, так как со временем gcd либо не меняется, либо уменьшается хотя бы в два раза) массивы префиксных и суффиксных количеств элементов, не кратных этому gcd. Тогда для каждого i проверка выполняется за $O(1)$.

Прикол - сложная версия

Для двух целых положительных чисел l и r определим $f(l, r)$ как количество упорядоченных пар натуральных чисел (a, b) таких, что $l \le a + b \le r$ и $\gcd(a, b)$ — простое число. Дано $q \le 10^6$ запросов, $l, r \le 5 \cdot 10^4$.

Пусть $d = \gcd(a, b)$. Тогда $a = d \cdot x$, $b = d \cdot y$, где $\gcd(x, y) = 1$. Условие на сумму: $l \le d(x + y) \le r$. Количество пар (x, y) с $\gcd(x, y) = 1$ и фиксированной суммой $s = x + y$ равно $\varphi(s)$ (функция Эйлера).

Для каждого простого d и для каждой суммы S от 1 до $5 \cdot 10^4$, кратной d, добавляем $\varphi(S/d)$ в $cnt[S]$. Функцию Эйлера $\varphi(m)$ можно вычислять на лету за $O(\sqrt{m})$ разложением на простые множители. Заметим: $m \le r \le 5 \cdot 10^4$, а количество вызовов φ равно $\sum_{p \le r} r/p \approx r \log\log r$. Затем строим префиксные суммы и отвечаем на запросы за $O(1)$. Сложность алгоритма: $O(r \log\log r \cdot \sqrt{r} + q)$

Прикол - очень сложная версия

Для двух целых положительных чисел l и r определим $f(l, r)$ как количество упорядоченных пар натуральных чисел (a, b) таких, что $l \le a + b \le r$ и $\gcd(a, b)$ — простое число. Дано $q \le 10^6$ запросов, $l, r \le 10^6$.

Пусть $d = \gcd(a, b)$. Тогда $a = d \cdot x$, $b = d \cdot y$, где $\gcd(x, y) = 1$. Условие на сумму: $l \le d(x + y) \le r$. Количество пар (x, y) с $\gcd(x, y) = 1$ и фиксированной суммой $s = x + y$ равно $\varphi(s)$ (функция Эйлера).

Для каждого простого d и для каждой суммы S от 1 до 10^6, кратной d, добавляем $\varphi(S/d)$ в $cnt[S]$. Для этого предвычисляем φ для всех чисел до 10^6 с помощью линейного решета, а затем вторым решетом для каждого простого d перебираем кратные $S = d \cdot s$ и добавляем $\varphi(s)$ в $cnt[S]$. Получаем массив $cnt[S]$ — количество пар с суммой S. Затем строим префиксные суммы и отвечаем на запросы за $O(1)$. Сложность алгоритма: $O(r \log\log r + q)$.

Циклические блоки - легкая версия

Дан массив a длины n. Разбиваем на блоки размера k (где k делит n). Разрешены операции: циклический сдвиг всех блоков влево (первый блок становится последним) и обмен соседних блоков. Сначала выполняются все сдвиги, затем все обмены. Нужно минимальное количество операций, чтобы отсортировать массив. Здесь $n \le 5000$.

Для каждого k, делящего n, проверим, что каждый блок уже отсортирован по возрастанию (иначе такой k не подходит). Если условие выполнено, заменим каждый блок на пару (первый элемент, последний элемент). Теперь массив блоков нужно упорядочить так, чтобы пары шли в порядке неубывания первых элементов, а внутри одинаковых первых — по вторым.

Перебираем все циклические сдвиги (их n/k). Для каждого сдвига i считаем количество инверсий inv в получившейся последовательности, ведь это и есть минимальное число обменов для сортировки (искать можно, например, сортировкой слиянием). Стоимость для сдвига i равна $i + inv$. Осталось лишь выбрать минимум по значениям сдвиги + инверсии. Ответ — минимум по всем k.

Циклические блоки - сложная версия

Дан массив a длины n. Разбиваем на блоки размера k (где k делит n). Разрешены операции: циклический сдвиг всех блоков влево (первый блок становится последним) и обмен соседних блоков. Сначала выполняются все сдвиги, затем все обмены. Нужно минимальное количество операций, чтобы отсортировать массив. Здесь $n \leq 2 \cdot 10^5$.

Для каждого k, делящего n, проверим, что каждый блок уже отсортирован по возрастанию (иначе такой k не подходит). Если условие выполнено, заменим каждый блок на пару (первый элемент, последний элемент). Теперь массив блоков нужно упорядочить так, чтобы пары шли в порядке неубывания первых элементов, а внутри одинаковых первых — по вторым.

Разрешается циклически сдвигать весь массив блоков (стоимость сдвига $= 1$) и менять местами соседние блоки (стоимость обмена $= 1$). Поскольку сдвиги делаются до обменов, можно перебрать все возможные циклические сдвиги (их n/k), для каждого поддерживать количество инверсий:

Перебирать все сдвиги и считать инверсии заново за $O(n/k \log n/k)$ слишком долго. Заметим, что при переходе от сдвига s к сдвигу $s+1$ мы переносим s-ый элемент из начала в конец. Количество инверсий меняется предсказуемо: $\text{invnew} = \text{invlast} - \text{less} + \text{more}$, где less — количество элементов, меньших s-ого, а more — количество элементов, больших s-ого. $less$ и $more$ можно предпосчитать для каждого элемента.

Осталось лишь выбрать минимум по значениям сдвиги + инверсии. Ответ — минимум по всем k.