Разбор задачи «Составление МОШ»

Минимальное количество задач равно $\max(n - k * b, 0)$, максимальное количество задач – $n - a$.

Разбор задачи «Ещё одна акция»

Пусть $n = (m + 1) \cdot q + r$.

Заметим, что акцией выгодно пользоваться если $a \cdot m \leqslant b \cdot (m + 1)$. В таком случае q раз купим картошку по акции. Оставшуюся картошку (или всю, если акция не выгодна) можно купить по цене $\min(a, b)$ за килограмм.

Тогда ответ:

$$q \cdot \min(a \cdot m, b \cdot (m + 1)) + r \cdot \min(a, b)$$

Асимптотика решения: $\mathcal{O}(1)$

Разбор задачи «Ещё одна акция»

В этой версии задачи $n \leqslant 10^6$. Давайте переберем, сколько мы купим килограмм в первый день. Пусть это число равно k. Тогда общая цена такой покупки будет равна $k \cdot a + max(n - (k + \lfloor \frac{k}{m} \rfloor), 0) \cdot b$. После чего возьмем минимум по этим знакичениям для всех k.

Асимптотика решения: $\mathcal{O}(n)$.

Разбор задачи «Федя и массив»

Заметим, что локальные минимумы и максимумы будут чередоваться, и их будет одинаковое количество k. Обозначим i-й локальный максимумум за a_i, i-й локальный минимум за b_i. Без потери общности считаем, что a_i идет раньше b_i. Чтобы перейти от a_i к b_i надо выписать $a_i - b_i$ чисел, от b_i к $a_{(i+1) \bmod k}$ надо $a_{(i+1) \bmod k} - b_i$.

Тогда

$$(a_1 - b_1) + (a_2 - b_1) + (a_2 - b_2) + \ldots + (a_k - b_k) + (a_1 - b_k) = 2 \cdot (a_1 + a_2 + \ldots + a_k) - 2 \cdot (b_1 + b_2 + \ldots + b_k) = 2 \cdot (A - B) = n$$

В качестве массива подойдет $[B, B + 1, B + 2, \ldots, A - 1, A, A - 1, A - 2, \ldots, B + 1]$.

Разбор задачи «Проверка на списывание»

Если строки состоят только из символов ‘a’, ‘b’, то ответ всегда «YES» и нужно только проверить, заменяется ли на каких-то позициях ‘a’ на ‘b’ и ‘b’ на ‘a’.

В общем случае есть следующий критерий. Рассмотрим каждый символ, хотя бы раз входящий в первую строку. Обозначим рассматриваемый символ за x и найдем все позиции, на которых x в нее входит. Посмотрим на множество различных символов не равных x на тех же позициях во второй строке. Если в этом множестве есть хотя бы два символа, то получить из первой строки вторую невозможно. Если множество пустое, то символ не участвует в заменах. А если в множестве ровно один символ y, то пара x y будет одной из нужных замен.

Если после проверки для какого-то символа выяснилось, что преобразовать строки нельзя, то нужно вывести «NO». Иначе ответ «YES» и нужно вывести все найденные замены.

Разбор задачи «Преобразование строки»

Если строки состоят только из символов ‘a’, ‘b’, то ответ всегда «YES» и нужно только проверить, заменяется ли на каких-то позициях ‘a’ на ‘b’ и ‘b’ на ‘a’.

В общем случае есть следующий критерий. Рассмотрим каждый символ, хотя бы раз входящий в первую строку. Обозначим рассматриваемый символ за x и найдем все позиции, на которых x в нее входит. Посмотрим на множество различных символов не равных x на тех же позициях во второй строке. Если в этом множестве есть хотя бы два символа, то получить из первой строки вторую невозможно. Если множество пустое, то символ не участвует в заменах. А если в множестве ровно один символ y, то пара x y будет одной из нужных замен.

Если после проверки для какого-то символа выяснилось, что преобразовать строки нельзя, то нужно вывести «NO». Иначе ответ «YES» и нужно вывести все найденные замены.

Разбор задачи «Даша и поиски»

Подгруппы решались разными версиями полного перебора всех отрезков.

Давайте рассмотрим ключевой факт для решения. Предположим мы хотим проверить удовлетворяет ли весь массив искомому требованию. Если это так, то мы можем вывести весь массив как ответ. Иначе один из двух крайних элементов не удовлетворяет нашим требованиям. Из этого можно сделать вывод, что все отрезки содержащие элемент, который не удовлетворяет нашим требованиям будут также некорректными, потому что этот крайний элемент будет оставаться минимумом/максимумом.

Из факта выше следует алгоритм: давайте посмотрим на отрезок $[l; r]$, который изначально равен $[1; n]$. Если $a_l = \min(a_l, a_{l+1}, \ldots, a_r)$ или $a_l = \max(a_l, a_{l+1}, \ldots, a_r)$, то перейдем к отрезку $[l+1; r]$. Также необходимо аналогичное рассуждения для a_r. Таким образом мы либо через сколько-то иттераций получим требуемый отрезок, либо получим $l == r$ и ответом будет -1.

Итоговая ассимптотика: $\mathcal{O}(n \log n)$ или $(O)(n)$ в зависимости от реализации.

Разбор задачи «Московские гориллы»

Обозначим за pos_x индекс числа x в перестановке. Подотрезки с MEX > 1 имеют вид $1 \leqslant l \leqslant pos_1 \leqslant r \leqslant n$.

Введем обозначения: $l_x = \min[pos_1, pos_2, \ldots, pos_x]$, $r_x = \max[pos_1, pos_2, \ldots, pos_x]$.

Подотрезки с MEX $> x$ имеют вид $1 \leqslant l \leqslant l_x \leqslant r_x \leqslant r \leqslant n$. Давайте определим вид подотрезков с MEX $= x$.

Если $pos_{x+1} < l_x$, тогда подотрезки с MEX $= x + 1$ имеют вид $pos_{x+1} < l \leqslant l_x \leqslant r_x \leqslant r \leqslant n$

Если $l_x \leqslant pos_{x+1} \leqslant r_x$, тогда не существует подотрезка с MEX $= x + 1$

Если $r_x < pos_{x+1}$, тогда подотрезки с MEX $= x + 1$ имеют вид $1 \leqslant l \leqslant l_x \leqslant r_x \leqslant r < pos_{x+1}$

Осталось всего лишь пересечь множества таких подотрезков для p и q, что делается тривиально.

Разбор задачи «Велепин и маркетинг»

Давайте отсортируем людей по их требованию размера группы. Предположим у нас есть такой человек i, что он не доволен, и есть человек $j > i$, который доволен. Тогда мы можем заменить j человека в его группе на i и ответ для нас не ухудшится. Отсюда следует, что для конкретного k ответ это какой-то префикс людей, которых мы можем сделать довольными.

Давайте также докажем, что существует такая расстановка групп, которая покрывает тот же самый префикс, и каждая группа это непрерывный отрезок. Давайте возьмем какое-нибудь корректное разбиение по группам. Тогда каждая группа будет представлять из себя набор несвязных отрезков. Давайте возьмем самый левый из таких отрезков. Заметим, что мы можем его про swap'ать до ближайшего отрезка такой же группы справа, при чем ничего не сломав.

Таким образом мы получили, что мы можем искать решение в виде разбиения каждого префикса на валидные группы, которые являются отрезками. Будем решать эту задачу с помощью динамического программирования.

Пусть $dp[i]$ – максимальное количество групп, на которые можно разбить i-й префикс, так чтобы все были довольны (при чем нельзя использовать элементы за префиксом). База динамики: $dp[0] = 0$ (пустой префикс максимум можно разбить на 0 групп). Переход: для i-го человека его группа должна иметь размер хотя бы $a[i]$, поэтому переход выглядит следующим образом $dp[i] = \max\limits_{0 \leqslant j \leqslant i - a[i]} dp[j] + 1$. Но что если $a[i] > i$? Тогда мы не можем набрать i-й префикс. Тогда положим $dp[i] = -\infty$. Эту динамику можно посчитать с помощью префиксных максимумов. Эта часть решения работает за $\emptyset(n)$.

Ранее мы сказали, что ответом будет являться какой-то префикс людей, которые будут довольны. Если мы можем разбить префикс на какое-то количество групп, то этот ответ может являтся префиксом для всех $k \leqslant dp[i] + n - i$. (мы разбиваем наш префикс соотвественно dp, а остальных людей расскидываем по одному в группу)

Если мы не можем сделать весь префикс довольным ($dp[i] = -\infty$), то нам надо докидывать людей извне. Таким образом максимальное количество групп, на которые мы можем разбить, если i-й префикс полностью доволен, это $n - a[i] + 1$.

Заметим, что если каким-то префиксом мы можем набрать k, то тогда можем набрать и $k - 1$ (объединив две группы в одну). Тогда нам нужно найти самый большой префикс, который подходит

для данного k в запросе. Это можно сделать массивом суффиксных максимумом за $O(q)$ суммарно. Итоговая асимптотика решения: $\emptyset(n \log n + q)$.

Разбор задачи «Ребрендинг»

Ниже будет приведено два решения, которые проходили все тесты.

Решение за $\mathcal{O}(n\sqrt{n} + q \log q)$:

Пусть длина отрезка из запроса равна $\mathcal{O}(k)$. Тогда ответ не превосходит $\mathcal{O}(\frac{n}{k})$. Действительно, пусть ответ равен $x > \mathcal{O}(\frac{n}{k})$, тогда посмотрим на худший случай. Будем расставлять числа жадно – $1, x+1, 2x+2, \ldots, n$. Всего чисел будет $\mathcal{O}(\frac{n}{x}) < \mathcal{O}(\frac{n}{n/k}) = \mathcal{O}(k)$. Значит чисел будет меньше чем мы предполагали – получили противоречие. Таким образом получаем утверждение, что если длина отрезка $\geqslant \mathcal{O}(k)$, то ответ $\leqslant \mathcal{O}(\frac{n}{k})$.

Давайте возьмем $k = \sqrt{n}$. Для всех отрезков длины $\leqslant k$ посчитаем их ответ за $\mathcal{O}(k \log k)$ или $\mathcal{O}(k)$ (первое делается сортировкой всего отрезка). Для остальных отрезков нас будут интересовать только такие i, j, что $|a_i - a_j| \leqslant \mathcal{O}(\sqrt{n})$. В зависимости от реализации обработки всех этих пар и релаксированние для них ответов ваше решение могло получать от 62 до 90 баллов.

Решим сначала задачу для запросов с длиной отрезка $\leqslant \sqrt{n}$. Будем идти по элементам слева направо, и для каждого элемента поддерживать $dp[i]$ – минимальная разность элемента i с элементами правее, которые мы успели рассмотреть. Более формально, пусть r – текущая правая граница, тогда $dp[i] = \min\limits_{i<j\leqslant r} |a[i]-a[j]|$. При переходе к $r+1$ нам нужно пересчитать $dp[i]$ только для $r+1-\sqrt{n} \leqslant i \leqslant r$. Используя $dp[i]$ можно легко отвечать на запросы, длина которых $\leqslant \sqrt{n}$ Действительно, давайте для i отрезка, когда мы будем в позиции l_i посмотрим на $\min\limits_{l_i\leqslant j<r} dp[j]$. Это и будет ответом. Эта часть решения работает за $\mathcal{O}(n\sqrt{n})$.

Давайте теперь обработаем отрезки, длина которых $\geqslant \sqrt{n}$. В этом случае ответ $\leqslant \sqrt{n}$. Для каждого числа есть только $2\sqrt{n}$ чисел, которые образуют с ним разницу $\leqslant \sqrt{n}$. Давайте будем идти точно также слева направо и для каждого ответа от 1 до $\sqrt{n}$ поддерживать максимальную левую границу, при которой мы можем набрать эту разницу. Обозначим это за $dp2[i]$. Пусть мы посчитали $dp2[i]$ для r, перейдем к $r+1$. Нам нужно обработать все пары, в которые входит $a[r+1]$ и разность в которых меньше $\sqrt{n}$. Это можно сделать явно суммарно за $\mathcal{O}(n\sqrt{n})$. Для каждого отрезка мы можем явно найти первый ответ ans, для которого $dp2[ans] \leqslant l_i$. Но мы несложно заметить, что если $l_{i-1} \leqslant l_i$, то $ans_{i-1} \leqslant ans_i$. Мы можем двигать указатель и получить ответ для всех отрезков такого типа за $\mathcal{O}(n\sqrt{n} + q \log q)$ (потому что нужно еще посортировать отрезки). При должном старании и подборе k это решение проходило все тесты.

Решение за $\mathcal{O}(n \log^2 n + q \log n)$:

Давайте будем также идти по всем элементам слева направо. Основной задачей будет поддержка актуальной версии $dp[i]$ – минимальной разности a_i с элементами справа от него, который мы успели рассмотреть. Пусть мы корректно посчитали dp для первых r элементов. Перейдем к $r+1$. Давайте покажем как обновить ответ для всех $j < i$, таких что $a[j] > a[i]$. Для $j < i$, таких что $a[j] < a[i]$ решается аналогично.

Давайте возьмем первый элемент $a[j]$ слева от i, такой что $a[j] > a[i]$. Заметим, что если есть $l < j < i$, такой что $a[l] > a[j] > a[i]$, то для него мы $dp[l]$ не будем обновлять, потому что $|a[l] - a[j]| < |a[l] - a[i]|$. Также мы не будем обновлять ответ для l таких что $|a[l] - a[j]| < |a[l] - a[i]| \to a[l] > a[i] + \frac{a[j]-a[i]}{2}$. То есть дальше нас будут интересовать только числа с отрезка $\left[a[i], a[i] + \frac{a[j]-a[l]}{2}\right]$.

Давайте заметим, что мы уменьшили длину отрезка в 2 раза. То есть таких иттераций будет не больше $\mathcal{O}(\log n)$. Находить самое правое число, принадлежащее отрезку, можно с помощью дерева отрезков. Ответом для отрезка l_i, r_i будет $\min\limits_{l_i\leqslant j<r} dp[l]$ в момент r_i. Это также можно эффективно находить с помощью дерева отрезков. Итоговая асимптотика решения $\mathcal{O}(n \log^2 n + q \log n)$.