

Задача А. Шоколад

Рассмотрим произвольные две соседних дольки первой плитки. Если на второй плитке они не стоят рядом, причём в таком же порядке, то в месте соединения этих квадратиков первую плитку так или иначе придётся разломать. Вместе с тем, разломав плитку во всех таких местах и только в них, получим части, каждая из которых является фрагментом последовательных долек второй плитки. Поэтому, сложив их правильным образом, получим тот же порядок рисунков, что и на второй плитке.

Следовательно, нам осталось для каждого числа выяснить, идёт ли перед ним на первой и на второй плитке одно и то же значение или нет. Чтобы это сделать, можно для всех чисел проводить отдельный поиск по одной или обоим плиткам за $\mathcal{O}(n^2)$.

Для полного решения можно создать для одной из плиток индексный массив $p[i]$, где для каждого i , хранить число, которое на данной плитке идёт перед i (или, например, 0, если i — первое число на плитке). Далее, можно обработать дольки второй шоколадки, не выполняя поиск. Получаем решение за $\mathcal{O}(n)$.

Задача В. Логарифмическая последовательность

Поскольку $a_{1.1} = a_1 + a_1$, то a_1 должно быть равно 0. Тогда при изменении первого элемента последовательность не может снова стать логарифмической. Заметим, что $a_{xy} = a_x + a_y$ эквивалентно $a_m = \alpha_1 a_{p_1} + \alpha_2 a_{p_2} + \dots + \alpha_k a_{p_k}$, где $m = p_1^{\alpha_1} \dots p_k^{\alpha_k}$. Таким образом, любая логарифмическая последовательность однозначно определяется по элементам, индексы которых — простые числа. Если индекс p изменённого элемента простое число, то один из возможных способов «исправить» последовательность — изменить все элементы с индексами, кратными p . Таких элементов $\lfloor \frac{n}{p} \rfloor - 1$. Так как для любого индекса x надо изменить хотя бы один его простой делитель, то минимальное количество изменений для этого индекса равно $\lfloor \frac{n}{\text{максимальный простой делитель } x} \rfloor - 1$. Докажем, что это действительно является минимальным количеством изменений.

Воспользуемся индукцией по n и покажем, что для простых чисел p , больших двух, изменение кратных p , является единственным минимальным решением.

Если n не делится на p , то количество изменений остаётся неизменным, а решение единственно.

Если n делится на p , то очевидно, что решение для $n - 1$ нужно расширить, изменив n -й элемент, и мы получаем, что $\lfloor \frac{n}{p} \rfloor - 1$ — наименьшее число изменений. Предположим, что существует другое минимальное решение, которое не меняет n -й элемент и необходимо изменить другой нечётный простой индекс q (для $q = 2$ можно отдельно показать, что он требует больше число изменений), для которого по предположению индукции есть единственное решение для первых $n - 1$ элемента. Поскольку $2q$ не делится на p , это решение будет иметь по крайней мере $\lfloor \frac{n}{p} \rfloor$ изменений, поэтому оно не является минимальным.

Для $p = 2$ не обязательно, чтобы минимальное решение было единственным (для простых $\frac{n}{3} < q \leq \frac{n}{2}$ можно поменять q -й вместо $2q$ -го), но все ещё верно, что минимальное число необходимых изменений равно $\lfloor \frac{n}{2} \rfloor - 1$. Доказательство этого утверждения аналогично предыдущему.

Тогда в решении достаточно найти максимальный простой делитель x и вычислить по формуле ответ, это стандартная задача. Итоговое время работы $\mathcal{O}(q\sqrt{n})$. Также существуют альтернативные решения, использующие решето Эратосфена с линейным временем работы и вычислением факторизации числа за время порядка размера факторизации, но этого не требовалось в задаче.

Задача С. Машина с шариками

Так как шары из машины удаляются по одному, а максимум в машине может находиться не более n шаров, то суммарно будет добавлено не более $\mathcal{O}(q)$ шаров.

Найдем порядок, в котором будут добавляться шары, если операций удаления нет. Для этого найдем минимум в каждом поддереве, после этого обойдем поддерева в порядке post-order (LRN) ([https://ru.wikipedia.org/wiki/Обход_дерева#Обратный_обход_\(LRN\)](https://ru.wikipedia.org/wiki/Обход_дерева#Обратный_обход_(LRN))). Теперь для каждой вершины известен ее порядковый номер добавления (приоритет), можно хранить в любой структуре, позволяющей искать минимум, вершины и их приоритеты. Например, `std::set`. Таким образом суммарно запросы добавления работают за $\mathcal{O}(q \log n)$.

После удаления шара, вся цепочка из подряд идущих шаров в предках, скатится вниз, это равносильно тому, что самый верхний шар переместится вниз, искать последний шар в цепочке можно с помощью двоичных подъемов аналогично идее поиска LCA.

Задача D. Игра с массивом

Заметим, если каждому числу k сопоставить число Фибоначчи с таким номером F_k , то операция сжатия не меняет сумму подотрезка, из этого следует что каждый отрезок сжимается только в одно конкретное число или не сжимается вообще. Из этого же факта можно оценить максимальное возможное значение, которое можно получить: $n \cdot F_{40} < F_{80}$.

Пусть $dp[i][x]$ — длина отрезка с началом в i , который сожмется в x . Тогда для получения x предварительно были $x - 1$ и $x - 2$. Переберем оба варианта расположения и пересчитаем динамику $dp[i][x] = dp[i][x - 1] + dp[i + dp[i][x - 1]][x - 2]$ (или $dp[i][x] = dp[i][x - 2] + dp[i + dp[i][x - 2]][x - 1]$).

Далее насчитаем динамику $len[i]$ минимальная длина суффикса, если начать рассматривать операции только на суффиксе с позиции i . Для пересчёта переберем первый элемент суффикса и рассмотрим оставшийся суффикс суффикса $len[i] = 1 + \min_x len[i + dp[i][x]]$. Тогда ответ находится в $len[0]$, запоминая элементы по которым был произведен пересчет можно восстановить итоговый массив.