Разбор задачи «Старик и шахматная доска»

У этой задачи было множество решений, и вот одно из них:

Переберем длину стороны итоговой доски — она точно не превосходит $\lceil\sqrt{n+m}\rceil$. Если перебранная длина x четная, и черных, и белых клеток в шахматной раскраске этой доски будет $x^2/2$, то есть каждое из чисел n и m должно быть не меньше этого числа. Если же x — нечетное, то черные и белые клетки делятся как $(x^2-1)/2$ и $(x^2+1)/2$, и нужно рассмотреть два случая на числа n и m.

Разбор задачи «Великий бой»

Заметим, что если мы поделим число x ($1 \leqslant x \leqslant 10^9$) 30 раз на число большее единицы, то оно станет равно нулю, так как после каждого деления число уменьшается хотя бы в два раза. Будем поддерживать в сете позиции чисел больших единицы и делить только на них.

1. Заведем структуру данных, которая умеет отнимать 1 на отрезке и находить максимум на отрезке, например, дерево отрезков. Сделаем $-=1$ на (l, r), а затем, пока $\min(s_l, s_{l+1}, \ldots, s_r) = 1$, удалим из сета позицию минимума.

2. Найдем двоичным поиском в сете первую позицию не единицы большую либо равную l. Затем будем переходить к следующей позиции в сете и делить на нее. Чтобы узнать значение в конкретной позиции, необходимо обратиться к структуре. Как только в результате деления был получен 0, мы останавливаемся.

Время работы:

1. Суммарно первый тип запросов будет работать за $\mathcal{O}(n \cdot \log(n) + q \cdot \log(n))$.

2. Второй тип запросов будет работать за $\mathcal{O}(q \cdot \log(n) \cdot \log(maxC))$

Разбор задачи «Ящик Пандоры»

Будем считать динамику dp_i — минимальное число действий, чтобы сделать неубывающим суффикс массива от i до $n-1$ (здесь и далее нумерация массивов с 0 до $n-1$).

Пусть $T(l, r) = 0$, если элементы с l по r одинаковые и 1 иначе. Пойдем **с конца массива** и заметим, что

$$dp_i = dp_j + T(i, j-1), \text{ если } a_{j-1} = a_i \text{ и } j > i.$$

Очевидно, для получения минимального количества действий, $dp_j + T(i, j-1)$ должно быть минимально возможным. Заметим, что $dp_j + T(i, j-1)$ минимально при минимальном dp_j, а если таких несколько, то при минимальном j.

Тогда для каждого числа a_i из входных данных будем поддерживать $p[a_i]$ — индекс минимального dp_j, такого, что $a_{j-1} = a_i$ и $a_{j-1} < a_j$. Это можно поддерживать, проходя с конца по исходному массиву в процессе подсчета динамики.

Для того, чтобы не добавлять в dp_i действие, когда мы хотим выравнять отрезок с i по j, но на нем элементы и так одинаковые, заведем массив c, при этом $c_i = 1$, если $a_i = a_{i+1}$, иначе 0. Теперь заведём массив sum, для которого sum_i — сумма чисел в массиве c на отрезке с 0 по i (массив префиксных сумм).

Теперь $q = sum[p[a[i]] - 1] - sum[i-1]$ — количество чисел, равных a_i, на оптимальном отрезке, начинающемся с i. Тогда, если q — не равно длине этого отрезка, значит на нем существует отличное от концов отрезка число, и нужно потратить действие для уравнивания и $dp_i = dp[p[a[i]] + 1$, иначе $dp_i = dp[p[a[i]]$.

Разбор задачи «Кружок стрельбы»

Заметим, что при выстреле i-го ученика снаряд долетит до следующего ученика тогда и только тогда, когда $x_i + r_i c_i \geqslant x_{i+1}$. Значит, если для какого-то номера i выполняется неравенство $x_i > x_{i-1} + r_{i-1} c_{i-1}$, то ученику с этим номером обязательно нужно дать команду, так как снаряд до

него долететь не может. Также очевидно, что до ученика с номером 1 никакой снаряд долететь не может, и ему нужно дать команду. Заметим теперь, что всем остальным ученикам давать команду не нужно, так как до каждого из них долетит снаряд.

Таким образом, чтобы получить ответ, достаточно посчитать количество чисел x_i, для которых $x_i > x_{i-1} + r_{i-1}c_{i-1}$, и увеличить на единицу. Асимптотика времени работы решения составляет $\mathcal{O}(n)$.

Разбор задачи «Древнегреческий изоморфизм»

Рассмотрим клетку (i, j) и кратчайшие расстояния от неё до углов решетки. Тогда рассмотрим расстояния до углов $(1, 1)$ и $(1, m)$. Эти расстояния равны, соответственно, $i + j$ и $i + (m - j)$. Тогда мы можем найти i как $\frac{((i+j)+(i+(m-j))-m}{2}$. И можем найти j как $\frac{((i+j)-(i+(m-j))+m}{2}$.

Таким образом, если мы знаем кратчайшие расстояния до углов решетки, то мы можем однозначно восстановить координаты клетки.

Углам решетки соответствуют вершины степени два данного графа, и если он изоморфен решетке, то таких вершин ровно четыре. Переберем 4! вариантов какая вершина, какому углу соответствует.

Тогда мы для каждой вершины можно найти по формулам выше соответствующую вершину решетки, а затем проверить, что полученная перестановка вершин действительно является изоморфизмом решетки и данного графа.

Таким образом, мы получили решение за $\mathcal{O}(4!nm)$, что есть $\mathcal{O}(nm)$.

Разбор задачи «Гонки на колесницах»

Пусть на победу колесничего было поставлено a монет. Тогда для того, чтобы Кратос гарантированно остался в выигрыше, должны выполняться два неравенства:

1. $a \cdot x + a > n \Rightarrow a \cdot (x + 1) > n \Rightarrow a > \frac{n}{x+1}$
2. $(n - a) \cdot y + (n - a) > n \Rightarrow (n - a) \cdot y > a \Rightarrow a < \frac{n \cdot y}{y+1}$

Получаем ограничение на a: $\frac{n}{x+1} < a < \frac{n \cdot y}{y+1}$.

Для рассмотрения максимального потенциального выигрыша сравним максимальный a, умноженный на $x + 1$ и максимальный $n - a$ (при минимальном a), умноженный на $y + 1$.

Разбор задачи «Таинственный ритуал»

Заметим, что если $10 \leqslant n$, то после нескольких операций, значение n станет строго меньше. Это верно, т. к. до ближайшего деления на 10 можно максимум 9 раз прибавить по 9. Путь $\frac{n+9 \cdot 9}{10} \geqslant n$, тогда $n + 81 \geqslant 10 \cdot n$, тогда $81 \geqslant 9 \cdot n$, тогда $9 \geqslant n$, противоречие.

Будем бежать по цифрам от младших к старшим, и поддерживать значение переноса. Когда мы стоим на очередной цифре, сложим значение переноса и эту цифру. Тогда, если взять остаток по модулю 10 от этого числа, мы получим текущую последнюю цифру числа n. Зная последнюю цифру, мы можем понять, сколько раз ее нужно прибавить к n, чтобы n стало делиться на 10. Прибавим к переносу эту цифру, умноженную на нужное количество раз. Затем делим перенос на 10 и переходим к следующей цифре. Так продолжаем, пока остались нерассмотренные цифры, или перенос больше 9, или перенос не взаимно прост с 10. Когда процесс остановится, ответ будет лежать в переносе.

Разбор задачи «Покраска»

Будем для каждого значения t от 1 до $n + 1$ поддерживать, сколько клеток нужно перекрасить, чтобы все клетки до t-го ряда (не включая его) имели черный цвет, а после него — белый цвет. Для изначальной конфигурации эти значения легко посчитать за линейное время от размера поля (для каждого t искомая стоимость — это сумма количества клеток белого цвета до t-го ряда и количества клеток черного цвета после t-го ряда). Также будем хранить в массиве все цвета клеток.

Когда поступает запрос на изменение цвета клетки (a, b), посмотрим, какого цвета была эта клетка. Пусть она была черного цвета. Посмотрим, как изменятся значения, которые мы храним для каждого t. Легко заметить, что все значения до a-го включительно уменьшатся на 1, а все значения после него увеличатся на 1. Аналогично, если клетка перекрашивается из белого в черный цвет, то все значения до a-го включительно увеличатся на 1, а после a-го уменьшатся на 1. Также после

каждого запросы искомый ответ — это минимальное из значений, которые мы храним. Значит, нам нужно уметь прибавлять ко всем значениям на отрезке фиксированное число и брать минимальное среди всех значений. Для этих операций можно хранить значения в дереве отрезков на минимум с массовыми обновлениями.

Асимптотика времени работы решения составляет $\mathcal{O}(nm + q \log n)$.

Разбор задачи «Деревни лесорубов»

Подвесим дерево за вершину с номером 1. Задача решается методом динамического программирования по поддеревьям. Ответ для вершины — наибольшее количество кораблей, которое может произвести поддерево этой вершины. Рассмотрим два случая.

Если вершина v не находится в подчинении ни у одного наместника, то ответ для поддерева вершины v равен $a_v + \sum\limits_{u \in Ch_v} ans_u$, где Ch_v — множество детей вершины v, а ans_u — ответ для вершины u.

Второй случай — если в вершине v находится наместник. В таком случае, ответ для вершины v находится с помощью модифицированной задачи о рюкзаке. Изначально есть рюкзак вместимостью $|St_v| - |Ch_v|$, где St_v — множество потомков вершины v, включая саму вершину v. Вместимость рюкзака обозначает, сколько деревень могут поставлять лес. Изначально мы знаем, что любая вершина, кроме вершин из Ch_v, может поставлять лес. Также, есть $|Ch_v|$ предметов. Каждый предмет можно использовать для одного из трех действий:

- Положить в рюкзак. Вес предмета равен b_u, а стоимость — c_u. Это действие соответствует постройке мастерской в вершине $u \in Ch_v$.
- Увеличить вместимость рюкзака на 1. Это действие соответствует тому, что вершина $u \in Ch_v$ будет поставлять лес.
- Ничего не делать. Тогда предмет не кладется в рюкзак, но суммарная стоимость все равно увеличивается на a_u. Это действие соответствует тому, что вершина $u \in Ch_v$ будет строить корабли сама по себе.

Такая задача решается методом динамического программирования за $O(|St_v| \cdot |Ch_v|)$. В итоге, мы получили для каждого числа k от 0 до $(|St_v| - |Ch_v|)$ максимальное количество кораблей, которое может быть суммарно построено вершинами из Ch_v, если ровно k вершин из $St_v \setminus Ch_v$ будут поставлять лес ($St_v \setminus Ch_v$ означает множество вершин, находящихся в поддереве v, за исключением ее детей). Чтобы получить максимальное количество количество кораблей, которые могут суммарно построить все вершины из St_v при фиксированном k, нужно прибавить к посчитанному значению $(|St_v| - |Ch_v| - k)$ максимальных значений a_u, где $u \in St_v \setminus Ch_v$. Теперь ответ для вершины — максимум посчитанных значений по всем k.

Ассимптотика времени работы равна:
$O(\sum |St_v| \cdot |Ch_v|) = O(\sum n \cdot |Ch_v|) = O(n \cdot \sum |Ch_v|) = O(n \cdot n) = O(n^2)$.

Разбор задачи «Игра в строки»

Пусть массив a_i — количество i-х букв алфавита, которое есть во второй строке. Зафиксируем подстроку длины k первой строки. Пусть массив b_i — количество i-х букв алфавита, которое есть в этой подстроке. Тогда надо проверить, что существует подстрока, для которой $b_i \leqslant a_i$ для всех i от 1 до 26.

Посчитаем массив a за $O(|t|)$. Также посчитаем массив b для подстроки длины k первой строки, начинающейся в первом символе. Заметим, что при переходе от одной подстроки к следующей, в массиве b изменяется не более двух элементов, а значит этот переход делается за $O(1)$.

Тогда построить массив b для всех подстрок длины k первой строки можно за $O(|s|)$. Проверить каждую из них занимает $O(26)$ времени, где 26 — размер алфавита. Тогда всё решение работает за $O(26|s| + |t|)$.