Разбор задачи «Расследование убийства»

Если в формулу для $\beta(n,k)$ подставить формулу для $\beta(n-1,k)$, получится следующее:

$$\beta(n,k) = \frac{n-1+k}{n} \cdot \beta(n-1,k)$$

Раскрывая формулу выше рекурсивно, получается:

$$\beta(n,k) = \frac{(n-1+k)\cdot(n-1+k-1)\cdot\ldots\cdot k}{n\cdot(n-1)\cdot\ldots\cdot 2\cdot 1} \cdot \beta(0,k) = \frac{(n-1+k)!}{n!\cdot(k-1)!}$$

Предподсчитая заранее значения $n!$ и обратных к ним по модулю 998244353, можно узнавать значения $\beta(n,k)$ за $O(1)$.

Разбор задачи «Бомбы в Восточном Экспрессе»

Будем рассматривать бомбы подряд по одной. Рассмотрим бомбу с центром (x_b, y_b) и радиусом поражения r. Все двигатели, которые могут быть поражены этой бомбой, находятся в вертикальной бесконечной полосе, ограниченной лучами $x = x_b - r$ и $x = x_b + r$.

Переберем каждый x из этой полосы. По фиксированному x мы можем найти отрезок, который покрывает область поражения бомбы (то есть круг) просто решив уравнение $(y-y_b)^2 \leqslant r^2-(x-x_b)^2$. Для каждой вертикали сохраним все отрезки y-ов, которые мы для нее получили. Теперь нам достаточно найти количество двигателей на этой вертикали, которые не покрыты ни одним отрезком. Это можно с помощью сканирующей прямой за $O((s+c)\log(s))$, где s — количество отрезков на этой вертикали, а c — количество двигателей.

Итоговая асимптотика: $O((m\cdot r+n)\cdot(\log(m)+\log(n)))$. Также могли возникнуть проблемы с лимитом по памяти, для решения этих проблем можно было заметить, что для хранения данных в данной задаче достаточно использовать тип `short`, который в 2 раза меньше типа данных `int`.

Разбор задачи «Бюджет»

Решим задачу с помощью динамического программирования. Будем поддерживать массив $dp[i][j]$, в ячейке которого будем хранить значение «true», если на i-ом префиксе можно получить сумму j, «false» — если нельзя. База динамики: $dp[0][f[i]] = true$ (если $f[i]$ попадает в отрезок $[a,b]$), $dp[0][-f[i]] = true$ (аналогично, если $-f[i]$ попадает в отрезок $[a,b]$). Соответственно, во внешнем цикле будем перебирать все префиксы (по i), во внутреннем — суммы от a до b (по j). Если на предыдущем префиксе набралась сумма j, то есть $dp[i-1][j] == true$, значит на текущем префиксе набирается сумма $j+f[i]$, а также $j-f[i]$.

Если на последнем префиксе (то есть на всем массиве) набралась хотя бы одна сумма, лежащая на отрезке $[a,b]$, значит ответ на задачу существует. Чтобы восстановить ответ, необходимо знать, прибавляли ли мы или вычитали текущее число в массиве, будем также хранить эти данные в ячейке массива dp. Тогда, зная, какая сумма могла быть получена на последнем префиксе и какой знак был у последнего числа в массиве при получении этой суммы, мы можем переходить по суммам на меньших префиксах и получать знаки остальных чисел.

Асимптотика: $O(n\cdot(b-a))$.

Разбор задачи «Ключ к шифру»

Пусть подстрока t является ответом на задачу. Обозначим за t_1 и t_2 префикс и суффикс t максимальные по длине и не равные t. Обозначим за s_1 и s_2 суффиксы строки s, префиксами которых являются t_1 и t_2 соответственно. Тогда t_1 и t_2 являются наибольшим общим префиксом s_1 и s_2.

Построим суффиксный массив строки s и посчитаем наибольшие общие префиксы соседних суффиксов в массиве. Отсортируем $n-1$ пару по возрастанию lcp. Будем поддерживать отрезки суффиксов в суффиксном массиве с одинаковым префиксом на текущий момент. Очередная пара соседних суффиксов, с минимальной длиной lcp среди оставшихся, разбивает какой-то отрезок на два. Перед тем как разбивать отрезок, выберем на этом отрезке суффикс, который начинается раньше всего, и суффикс, который начинается позже всего. Обновим ответ подстрокой, начинающейся

там, где начинается первый суффикс, и заканчивающейся там, где начинается второй суффикс, плюс длина их lcp. После этого разобьем отрезок на два.

Итоговая асимптотика $O(n \cdot \log(n))$.

Разбор задачи «Проникновение в реликварий»

Пусть в исходной последовательности 1 имела индекс i. Тогда что бы проверить, возможно ли это, надо для каждого j от 1 до n проверить, что $|a_{(i+j-1) \mod n} - j| \leqslant 1$. Заметим, что для каждого элемента это условие выполнится не более трёх раз, поэтому весь алгоритм будет работать за линейное время.

Разбор задачи «Подозрительная строка»

То, что каждый символ перемещается не более чем на одну позицию, означает, что нам разрешается менять местами соседние символы, при этом каждый символ можно трогать не более одного раза.

Задача решается при помощи динамического программирования.

$d[i][f_1][f_2]$, где $i \leqslant \lfloor \frac{n}{2} \rfloor$, $f_1, f_2 \in \{false, true\}$ равно $true$, если можно получить строку, в которой префикс длины i совпадает с развёрнутым суффиксом длины i, при этом f_1 означает, был ли последний символ этого префикса поменян местами со следующим за ним, f_2 — был ли первый символ суффикса поменян местами с предыдущим.

$d[i][f_1][f_2]$ вычисляется из $d[i-1][g_1][g_2]$ (g_1, g_2 перебираем). g_j может быть $true$, только если $f_j = false$. Этот переход в динамике можно сделать, только если последний символ префикса длины i и первый элемент суффикса длины j совпадут после свопов, которые задаются значениями f_j, g_j.

Если n чётно, то ответ — это $d[n/2][false][false]$.

Если n нечётно, то ответ — это $d[\lfloor n/2 \rfloor][false][false]$ *or* $d[\lfloor n/2 \rfloor][true][false]$ *or* $d[\lfloor n/2 \rfloor][false][true]$.

Разбор задачи «Шум»

Для начала заметим, что чтобы восстановить какой-то вид исходного массива нужно к данному применить такую же функцию шума. Затем отсортируем массив по возрастанию, и при рассмотрении каждого элемента будем пытаться получить из него наименьший возможный такой, что в мы его еще не получали.

Пусть x текущий элемент, а y, предыдущее значение, тогда возможны три случая:

- $x - r > y$, тогда предположим, что элемент x был равен $x - r$ в первоначальном массиве.
- $x + r < y$, тогда все числа из диапазона $[x - r; x + r]$, уже были получены на предыдущих элементах, а значит, элемент x никак не привести к уникальному.
- иначе можем предположить, что вместо x раньше было значение $y + 1$, поставим его в ответ.

Соответственно, ответ — это сумма первых и третьих случаев, однако можно было сначала поставить числа таким способом, а потом уже посчитать ответ.

Для доказательства оптимальности данного подхода рассмотрим два числа x_0 и x_1, такие что $x_0 < x_1$. Очевидно, что пересечение диапазонов $[x_0 - r; x_0 + r]$ и $[x_1 - r; x_1 + r]$ не выгодно занимать при рассмотрении x_0, так как мы уменьшим количество возможных вариантов для x_1. В случае же, если мы возьмем минимальное число из $[x_0 - r; x_0 + r]$, которое до этого не брали, мы оставим наибольшее возможное количество вариантов для x_1.

Сложность решения $O(n \log(n) + n)$.

Разбор задачи «Интересная загадка»

Формализуем условие задачи. Нужно найти максимальное d^2 такое, что граф, в котором вершины — это исходные точки, а ребра проведены между всеми парами точек расстояние между которыми менее d, не связен. Сделаем бинпоиск по d^2, проведем возможные ребра и запустим дфс, чтобы проверить граф на связность. Мы можем так сделать так, как с увеличением d старые ребра не исчезают. Итоговая асимптотика $\mathcal{O}(n^2 \cdot \log Ans)$

Разбор задачи «Безопасный пароль»

Заменим каждый третий повторяющийся символ на самый редко встречаемый в строке, не равный соседним. Очевидно, что количество вхождений таких символов никогда не станет больше половины длины. Подсчитаем, сколько раз встречается каждый из символов в строке. Пусть нашелся символ, который встретился чаще, чем в половине позиций. Тогда заменим минимальное количество вхождений этого символа на самые редко встречаемые в строке, не равный соседним.

Разбор задачи «Стабильность транзакций»

Для начала заметим интересный факт, для двух транзакций $a_i < a_j$, нет смысла разбивать транзакцию a_i, но не разбивать транзакцию a_j, так как максимум в такой ситуации точно не измениться, а минимум может уменьшится, то есть ухудшить ответ.

Второй полезный факт: разбивать всегда надо на две равные части по тем же самым соображениям.

Тогда отсортируем транзакции по возрастанию и переберем, начиная с какой позиции мы будем бить транзакции на две. Пусть первая транзакция, которую разбили имеет номер k. Тогда:

- $MIN = min(a_1, a_k/2)$
- $MAX = max(a_{k-1}, a_n/2)$

Переберем все k, найдем оптимальный ответ. Сложность решения $O(n \log(n) + n)$.

Разбор задачи «Вагоны Восточного Экспресса»

Будем решать данную задачу с помощью перебора. Давайте зафиксируем, сколько раз проводник будет освобождать вагон целиком. Пусть это число равно i. В таком случае, нужно освободить i вагонов так, чтобы максимальное количество людей в одном из оставшихся вагонов было минимально (пусть это количество равно x). Тога проводнику потребуется ровно x выпускать по одному пассажиру из вагонов, где еще остались люди. Следовательно, нужно минимизировать сумму $x + i$. Поступим следующим образом — отсортируем вагоны по количеству людей в них в невозрастающем порядке. Тогда, зафиксировав какое-то i мы будем целиком освобождать первые i вагонов, а x будет равно a_{i+1} или 0, если i было равно n. Итого, взяв минимум по всем i от 0 до n среди значений $i + a_{i+1}$ мы получим оптимальный ответ.