Задача A. Монетки

Автор и разработчик задачи: Даниил Голов

Нам дана последовательность из n цифр 0 или 1. Затем, продолжается следующий процесс:

- Если последовательность не содержит 1, процесс останавливается.
- Иначе, пусть k — количество 1. Инвертируется элемент на позиции k (в 1-индексации).

Нужно либо сказать, за какое количество операций процесс остановится, либо, что он будет продолжаться бесконечно.

Рассмотрим инвертирование элемента на позиции k. Заметим, что до инвертирования существовал элемент на позиции $\geqslant k$, равный 1. Пусть он находился на позиции q. Тогда сначала $q-k$ раз мы заменим 0 на 1 на позициях $[k, q)$, затем элемент на позиции q поменяется с 1 на 0, и затем еще $q-k$ раз мы заменим 1 на 0 на позициях $[k, q)$. Таким образом, мы сделаем $(q-k)\cdot 2+1$ операций, после чего количество единиц уменьшится на 1. Отсюда следует, что процесс всегда конечный. А также, отсюда понятен алгоритм решения задачи за время $O(n \cdot \log(n))$. Нужно поддерживать множество позиций 1, и за один раз обрабатывать удаление одной единицы.

Задача B. Перестроение лемуров

Автор и разработчик задачи: Ильдар Гайнуллин

Нам нужно разделить перестановку на несколько подотрезков, минимизировав их суммарную стоимость. Цена одного подотрезка равна x плюс количество инверсий на этом подотрезке.

Обозначим количество инверсий на отрезке за $f_{l,r}$.

Заметим, что $f_{l,r} = f_{l+1,r} + f_{l,r-1} - f_{l+1,r-1} + (1, \text{если } p_l > p_r)$.

Значит, $f_{l,r} + f_{l+1,r-1} \geqslant f_{l+1,r} + f_{l,r-1}$, поэтому мы можем воспользоваться некоторыми оптимизациями ДП для решения этой задачи.

Будем оптимизировать $dp_i = \min(dp_j + f_{j+1,i})$.

Будем вычислять это ДП, используя метод Разделяй и властвуй.

Для начала, вычислим значения ДП для состояний $1, \ldots, \frac{n}{2}$ (рекурсивно).

Затем, нужно произвести пересчет значений $j > \frac{n}{2}$ через значения $i \leqslant \frac{n}{2}$. И затем, нужно будет рекурсивно вычислить значения для состояний $\frac{n}{2}+1, \ldots, n$.

Обратите внимание, что благодаря свойству нашей функции, оптимальное $i \leqslant \frac{n}{2}$ для j — монотонно возрастает при возрастании j.

Поэтому, можно воспользоваться методом Разделяй и властвуй для монотонных функций. Сначала найдем оптимальную точку для $i = \frac{(l+r)}{2}$ и затем рекурсивно разобьемся на две половины.

Но как вычислить $f_{l,r}$?

Во время Разделяй и властвуй, нам нужно будет переместить l и r только $\mathcal{O}(n \log n)$ раз суммарно. Поэтому, мы можем поддерживать $f_{l,r}$ и изменять его, когда нам нужно увеличить/уменьшить l/r. Это можно сделать с помощью дерева Фенвика, получив решение за **очень** быстрый $\mathcal{O}(n \log^3 n)$.

Также, можно предподсчитать некоторую информацию, используя SQRT декомпозицию. Можно заметить, что наши запросы выражаются через запросы вида «найти количество чисел $\leqslant x$ на отрезке $l \ldots r$». Используя sqrt декомпозицию, мы можем отвечать на эти запросы за $\mathcal{O}(1)$ с предподсчетом за $\mathcal{O}(n\sqrt{n})$, в итоге получится решение за $\mathcal{O}(n \log^2 n + n\sqrt{n})$.

Задача C. Электронный замок

Автор и разработчик задачи: Арсений Кириллов

Заметим, что мы всегда хотим получить более длинное число, так как длинное число больше короткого. Тогда для каждого n можно посчитать метод динамического программирования, какую максимальную длину можно получить, если включено ровно n сегментов. $len[n] = 1 + \max\limits_{x \in D}(len[n - cost[x]])$, где $cost[x]$ — количество горящих сегментов у цифры x, а D — множество доступных цифр.

Для вывода самого пароля, достаточно каждый раз выводить максимальную цифру, которая уменьшает возможную длину ровно на 1, а также не забыть, что первая цифра не должна быть нулевой.

Задача D. Помогите Прапору

Автор и разработчик задачи: Гайнуллин Ильдар

Чтобы найти идеальное паросочетание максимального веса, можно воспользоваться следующим жадным алгоритмом:

Выберем элемент наибольшего веса. Заметим, что ребер с весом равным этому элементу должно быть максимальное возможное количество. Поэтому, к ответу нужно добавить $a_i \cdot \min(i, (n+1-i))$. После чего, рекурсивно запуститься от большей половины с задачей «найти паросочетание максимального веса размера k для отрезка вершин».

Из этого алгоритма следует, что для фиксированной длины n, i-й элемент в отсортированном порядке прибавляет к ответу $a_i \cdot b_i$, где b_i — коэффициент, зависящий только от i. И тогда, для массива $a_1 \geqslant a_2 \geqslant \ldots \geqslant a_n$ ответом будет $\sum a_i \cdot b_i$.

Можно исследовать алгоритм, чтобы доказать, что b_i равно $(\frac{n+1}{2}!)^2 \cdot \binom{n-i-1}{\frac{n+1}{2}-1}$.

Задача E. Мосты

Автор задачи: Захаренко Артем, разработчик задачи: Михаил Анопренко

Построим граф, в котором вершинами будут компоненты реберной двусвязности исходного графа. Ребрами в полученном графе будут мосты исходного графа. Несложно заметить, что получившийся граф будет деревом. Пусть у него k листьев, тогда ответом является $\lceil \frac{k}{2} \rceil$. Понятно, что меньше ребер добавить нельзя, иначе бы в графе остался лист, а значит и мост. А сколько ребер достаточно, потому что можно соединять диаметрально противоположные вершины.

Задача F. Случайная задача

Автор и разработчик задачи: Ильдар Гайнуллин

Давайте выберем случайное простое число p в районе $\sqrt{n}$.

Для каждой тройки $(x_1 \bmod p, y_1 \bmod p, x_2 \bmod p)$ с $y_1 \bmod p > 0$ найдем $y_2 \bmod p$ с нужным значением $(x_1 \cdot y_1 + x_2 \cdot y_2) \bmod p$, давайте обозначим его за f_{x_1,y_1,x_2}.

После этого, для фиксированного i с $y_i \bmod p > 0$ зафиксируем $x_j \bmod p$, проверим все точки с таким значением x по модулю и $y_j \bmod p = f_{x_i,y_j,x_j}$.

Для точек с $y_i \bmod p$ проверим точки с $(x_j \cdot x_i) \bmod p = k \bmod p$.

Почему это работает быстро? Посмотрим на фиксированную пару $x_i \cdot x_j + y_i \cdot y_j \neq k$, но для которой выполняется равенство $\bmod p$. Есть $O(1)$ таких p, для которых это было бы верно. Так как если бы это выполнялось для большого количества простых модулей, по китайской теореме об остатках получилось бы, что равенство без модуля тоже верно. Поэтому, вероятность того, что фиксированная пара даст равенство по случайному модулю примерно равна $(\frac{1}{\sqrt{n}/logn})$. А значит, матожидание количества таких пар равно $n^2 \cdot \frac{\log n}{\sqrt{n}} = n \cdot \sqrt{n} \log n$.

Поэтому, если бы точки были не случайны, задачу можно было бы решить за $n \cdot \sqrt{n} \log n$.

Но точки случайны, поэтому $(x_i \cdot x_j + y_i \cdot y_j) \bmod p$ тоже случайно, поэтому есть примерно $n \cdot \sqrt{n}$ пар точек при фиксированном значении модуля.

Поэтому, всю задачу можно решить за $n \cdot \sqrt{n}$.

Задача G. Лемурьи вечеринки

Автор задачи и разработчик: Даниил Орешников

Для решения этой задачи достаточно переформулировать ее в терминах сочетаний и научиться считать частное факториалов по непростому модулю. Если C_a^b — число способов выбрать b элементов из a различных, то ответом на задачу будет

$$\sum_{t=0}^{k} C_k^t \cdot C_{k-t}^{n-2t}$$

Действительно, нам надо выбрать на n мест различными способоми какие виды лемуров будут представлены двумя лемурами, а какие — одним. Переберем количество «пар» лемуров t. Для фиксированного t: выбрать t пар можно C_k^t способами, а на оставшиеся $n-2t$ места надо разместить по одному лемуру из некоторых из оставшихся $k-t$ видов.

Теперь вспомним, что $C_n^k = \frac{n!}{k!(n-k)!}$ (где $x!$ — произведение всех чисел от 1 до x. Предподсчитаем все факториалы от 0 до n, но каждое значение будем хранить как набор степеней простых, делящих m, и часть, взаимно простая с m. Иными словами, запишем число v как $(d_0, \ldots, d_r, X)$ такие, что m делится на $\frac{v}{X} = p_0^{d_0} \cdot \ldots \cdot p_r^{d_r}$ и $\gcd(X, m) = 1$ (где $\{p_i\}$ — все простые из разложения m на простые делители). Для того, чтобы работать с такими представлениями, надо в самом начале разложить m за простые, что можно сделать перебором делителей.

Пересчет таких значений можно осуществлять за $\mathcal{O}(r)$, которое с большим запасом можно оценить сверху как $\log n$. Деление v на w реализуем как поэлементное вычитание соотвествующих степеней простых и деление X_v на X_w по модулю m, которое можно реализовать с помощью расширенного алгоритма Евклида (который может найти обратное по модулю m к X_w). Остается только перебрать t и посчитать ответ, суммарное время работы — $\mathcal{O}(\sqrt{m} + n \log n)$.

Задача H. Дом в дереве

Автор задачи: Орешников Даниил, разработчик задачи: Николай Будин

Оптимальное расположение лестниц строится конструктивно. Во-первых нужно поставить n лестниц между центральными комнатами на каждом этаже. Затем, пока количество оставшихся лестниц хотя бы n, нужно соединять лестницами все комнаты на одной вертикали. И если осталось меньше n лестниц, их тоже нужно поставить на одну вертикаль подряд.

После чего, осталось посчитать ответ. Формулы можно посмотреть в авторском решении.

Задача I. Подсчет операций

Автор задачи: Даниил Орешников, разработчик: Ильдар Загретдинов

Заметим, что для того, чтобы обнулить значение индикатора в листе дерева, нам необходимо сделать модуль от веса листа операций, при этом все вершины на пути от листа до корня тоже невозможно не затронуть. Поэтому минимальное количество операций для обнуления веса всех листьев — сумма модулей весов листьев в начале. После этого ненулевыми (возможно) остались веса вершин на предпоследнем уровне, для их обнуления придется сделать аналогичные операции. Заметим, что выполняя поиск в глубину, мы понимаем на каждом шаге, сколько нам нужно сделать операций для обнуления значения в текущей вершине (значение в вершине после применений операций к детям изменилось на сумму весов детей). Таким образом мы понимаем, сколько нужно сделать операций применительно к этой вершине.

Задача J. Черные и белые

Автор идеи: Мухаммаджон Хакимов, разработчик задачи: Григорий Хлытин

Если будем считать формулу итеративно, получим вердикт TL, т.к. такое решение будет работать за время $\mathcal{O}(n)$, где n — количество ходов в игре ($1 \leqslant n \leqslant 10^{18}$).

Заметим, что если расписать формулу искомой вероятности как $p = \left(1 - \frac{1}{2^2}\right) \cdot \left(1 - \frac{1}{3^2}\right) \cdot \ldots \cdot \left(1 - \frac{1}{(n+1)^2}\right)$, что равно $\left(\frac{2^2-1}{2^2}\right) \cdot \left(\frac{3^2-1}{3^2}\right) \cdot \ldots \cdot \left(\frac{(n+1)^2-1}{(n+1)^2}\right)$. Если раскрыть числители как разность квадратов, получим

$$p = \left(\frac{(2-1)(2+1)}{2^2}\right) \cdot \left(\frac{(3-1)(3+1)}{3^2}\right) \cdot \ldots \cdot \left(\frac{((n+1)-1)((n+1)+1)}{(n+1)^2}\right)$$

Сокращаем знаменатели всех дробей, кроме крайних, с числителями соседних и получаем ответ

$$p = \frac{n+2}{2(n+1)}$$

Так как числа $n+2$ и $n+1$ всегда взаимно просты, то чтобы получить несократимую дробь, остается только рассмотреть случай $(n+2) \bmod 2 = 0$, когда надо разделить числитель и знаменатель на 2. Суммарное время работы — $\mathcal{O}(1)$.