Разбор задачи «Аккуратная табличка»

Так как количество знаков должно быть одинаково, то во всех случаях, когда количество знаков в номере пользователя увеличивается на 1, должен поступать новый запрос (с количеством нулей в префиксе, на единицу меньшим). Таким образом, получаем, что обязателен новый запрос между 9 и 10, между 99 и 100, между 999 и 1000 и между 9999 и 10000.

Так как в первых двух случаях количество логинов меньше 500, то для чисел, меньших 10, нужен один запрос, для чисел от 10 до 99 — два запроса. Далее числа от 100 до 599 можно сгенерировать за три запроса, а вот для чисел от 699 до 999 уже понадобятся четыре (из-за лимита в 500 логинов на генерацию за один запрос). Дополнительный запрос понадобится для перехода границы между 999 и 1000, то есть получаем 5 запросов для чисел от 1000 до 1499 включительно. Далее уже особых случаев нет, так как переход следующей границы совпадает с границей блока по 500.

Таким образом, задача решается через несколько операторов `if`.

Разбор задачи «Бордеры»

Рассмотрим несколько утверждений, необходимых для решения.

Утверждение 1 Пусть у строки a есть бордеры s и t и $|s| < |t|$. Тогда s – бордер t.

В самом деле, так как t и s – бордеры a, то a начинается и с t, и с s, но длина s – меньше, поэтому s – префикс t. Аналогично, a заканчивается на t и s и s – суффикс t. Но это значит, что s – бордер t.

Утверждение 2 Пусть у строки a есть бордеры s и t, $|s| < |t|$ и t является также бордером строки b. Тогда s также является бордером строки b.

Действительно, из первого утверждения следует, что s – бордер t. Но t – бордер b, то есть оканчивается и начинается на b, значит, будет оканчиваться и начинаться и на s, то есть s – бордер t.

Из этого утверждения следует, что, для того чтобы найти $LCB(a, b)$, достаточно найти наименьший общий бордер в a, а затем проверить, является ли он бордером b, и если является, то он и $LCB(a, b)$, иначе $LCB(a, b)$ – это пустая строка. Это так, поскольку, если есть бОльший общий бордер, то наш будет являться бордером этого бОльшего бордера(из утверждения 1) и, следовательно, бордером обеих строк.

Для решения задачи найдем с помощью префикс-функции и несложной динамики длину наименьшего бордера для каждого префикса b. Пусть она равна ℓ для какого-то i. Тогда началом и концом для подходящей подстроки могут выступить все суффиксы в строке a, у которых длина наибольшего общего префикса с нашим бордером хотя бы ℓ.

Для вычисления количества таких суффиксов построим z-функцию по строке $b\#a$ заранее и посчитаем для каждого x, сколько суффиксов имеют наибольший общий префикс с b длины ровно x. Затем строим массив суффиксных сумм по этому массиву. Теперь у нас все готово для того, чтобы посчитать ответ.

Сложность – $O(n)$.

Разбор задачи «Внимание к деталям!»

Построим все римские числа в соответствии с таблицей, после чего для каждого числа посчитаем количество деталей, из которых оно собирается. Сложим все эти тройки в `set` *triples*. Общее количество различных троек будет равно 341. Далее решаем задачу динамическим программированием: `dp[i][j][k]` — минимальное количество римских чисел для набора из i элементов первого типа, j элементов второго типа и k элементов третьего типа. Инициализируем бесконечно большим в рамках задачи значением и для каждой тройки из *triples*, для которой соответствующее количество элементов $i1$, $j1$ и $k1$ не больше i, j или k проверяем, верно ли, что `1+dp[i-i1][j-j1][k-k1]` меньше текущего значения `dp[i][j][k]`; если верно — обновляем.

После вычисления таблицы просто отвечаем на запросы, беря соответствующее значение из неё за $O(1)$.

Общее количество операций при вычислении тем самым равно $341 \cdot 10^6$, что в принципе укладывается в ограничения задачи. Однако заметив, что во всех цифрах присутствует чётное количество наклонных элементов, можно сократить количество операций вдвое, сразу оставляя «бесконечность» для нечётных значений j.

Разбор задачи «Готовим интенсив...»

Исходя из условий задачи, «подсказанный» выбор всегда равен суммарному номеру хода $+1$, то есть если оставлять возможность выбрать соответствующую букву, то Боб всегда её будет выбирать. Переформулируем условие как «подсказка системы должна быть использована ровно $k' = \lfloor n/2 \rfloor - k$ раз». Тогда первые k' ходов выбираем буквы задач по порядку, в этом случае Боб делает детерминированный выбор, соответствующий подсказке. Затем всякий раз выбираем букву, номер которой на 2 больше текущего количества задач в наборе (если такая буква выбрана, выбираем как угодно), гарантируя, что Бобу придётся отказаться от подсказки.

Заметим, что при действиях в другом порядке программа жюри может начать сама выбирать буквы, которые блокируют автовыбор (то есть если вы заблокировали автовыбор на ходе x, то программа жюри заблокирует автовыбор на ходе $x + 2$ и так далее, то есть возможности автовыбора просто не будет), тем самым исключая возможность получить ровно k ситуаций, в которых автовыбор не работал.

Разбор задачи «Делимость и займы»

Давайте попробуем вывести некоторое наблюдение на основании данных нам свойств. Из того, что x_k – максимальное слагаемое в разбиении следует, что $x_k \geqslant \frac{n}{k}$.

n делится на x_k, поэтому можно представить n как $n = q_k \cdot x_k$ для некоторого натурального q_k.

Получим, что $x_k \geqslant \frac{q_k \cdot x_k}{k}$, откуда $q_k \leqslant k$.

Тогда давайте переберем q_k от 1 до k(на самом деле, если $k \neq 1$, можно перебирать от 2) и, если n делится на q_k, мы получим новый вариант для x_k. И мы получили ту же самую задачу, однако уже для меньших параметров! k уменьшается на 1, из n вычитается x_k. Теперь мы можем перебирать q_{k-1} до $k - 1$, у нас, правда, появилось дополнительное ограничение, которое мы должны запоминать и учитывать – что x_{k-1} не превосходит x_k. В итоге задача может быть решена рекурсивной функцией перебора $rec(n, k, last)$(здесь $last$ – предыдущий поставленный элемент, идем мы от наибольших слагаемых к наименьшим) за время $O(k!)$.

Разбор задачи «Естественная видимость»

В этой задаче просто требуется аккуратно реализовать всё то, что написано в условии. Проще всего это сделать, заведя два булевых массива 8×8 видимости фигур, после чего последовательно пройти по всем фигурам, отмечая поля, до которых фигура может дотянуться (при этом каким-то образом решив вопрос выхода за границы массива; проще всего это решить, написав отдельную функцию, которая проверяет, свободно ли поле (i, j) для данного цвета, и выдаёт `false` в случае, если координаты выходят за рамки диапазона $[1, 8]$ или же на поле стоит фигура (если фигура другого цвета — поле помечается как видимое) и `true`, если поле свободно, при этом поле помечается, как видимое).

Разбор задачи «Ё322»

Сделаем 297 раз `get`; если не получено -1, то в структуре было ровно 297 элементов, то есть она гарантированно пуста. Далее кладём два разных числа (например, 1 и 2) и делаем `get`: если выдалось второе — это стек, в противном случае это очередь.

Разбор задачи «Журналистское исследование»

Построим граф, вершинами которого являются площади, а рёбрами — трамвайные линии.

Маршрут найдётся тогда и только тогда, когда обе вершины находятся в одной компоненте связности. Таким образом, выделим компоненты связности с помощью обходов в глубину (dfs), для каждой компоненты посчитаем количество различных пар вершин из этой компоненты (оно равно $x(x - 1)/2$, где x — количество вершин в компоненте, просуммируем эти значения по всем компонентам, получив количество пар s, при которых маршрут найден. Общее количество всех пар равно $n(n-1)/2$. Посчитав наибольший общий делитель d найденной суммы и общего количества пар, разделим оба числа на него и выведем $p = s/d$ и $q = n(n-1)/2d$.

Разбор задачи «Залипающие кнопки»

Давайте разделим все 10^5 кодов на классы по количеству уникальных цифр, используемых в коде:

- 1 уникальная цифра: 10;
- 2 уникальных цифры: 1350;
- 3 уникальных цифры: 18000;
- 4 уникальных цифры: 50400;
- 5 уникальных цифр: 30240;

Заметим, что если использовать все коды из какого-то одного класса, то суммарное число использований цифры в этом классе будет совпадать для любых двух цифр из-за симметрии.

Любой код, состоящий из 5 уникальных цифр, задается битовой маской используемых цифр, а также перестановкой цифр. Коды из этого класса легко генерировать, соблюдая условие равномерности — если мы взяли какой-то код из 5 уникальных цифр, то можно взять битовую инверсию его маски и сгенерировать код из 5 других цифр. Таким образом каждая цифра от 0 до 9 была использована 1 раз. Так как количество перестановок для любой битовой маски совпадает и равно 5!, то можно одинаковое целое число раз использовать все маски и потом по одному разу использовать какой-то префикс масок.

Если нам нужно сгенерировать до 49600 кодов или более 50400, то можно полностью взять какие-то классы кодов и равномерно сгенерировать коды из уникальных чисел. Но в иных случаях придется использовать коды из 4 уникальных цифр. Так как таких кодов нужно немного, можно сгенерировать коды из 4 уникальных цифр какой-то простой формы. Например коды, где на первых двух позициях стоит дублирующаяся цифра, а на остальных позициях уникальные цифры в отсортированном порядке. Таких кодов будет $10 \cdot C_9^3 = 840$, и из-за симметрии каждая цифра используется в них одинаковое число раз.

Разбор задачи ««И» побитовое»

Сначала сформулируем важное свойство: для любого k $2^k > 2^0 + 2^1 + ... + 2^{k-1}$. Также заметим, что наша операция означает "флип"всех битов числа до его старшего бита – то есть нули заменятся на единицы, а единицы – на нули.

Давайте рассуждать над задачей "побитово". То есть, посмотрим на каждый бит и зададимся вопросом, может ли он быть в итоговом «and». А он будет тогда и только тогда, когда у всех чисел есть этот бит.

Проблема может быть в том, что, добавив в ответ некоторые биты, мы можем тем самым повлиять на другие, тем самым исключив их. Но обратимся к упомянутому в начале разбора свойству. Из него следует, что лучше точно в ответ взять бит k, чем взять туда все более младшие биты. Поэтому давайте найдем самый старший бит, который мы точно сможем взять в ответ – это такое минимальное r, что все числа меньше 2^{r+1}. Найдя этот бит, мы сможем понять, какие операции нам необходимо выполнять. В самом деле, если в некотором числе бит r равен нулю, нужно сделать операцию, иначе не стоит. Получив последовательность операций, мы можем сформировать итоговый массив a и найти его «and». Сложность решения – $O(n \cdot B)$, где B – максимально возможный номер бита в числах(в сложности n домножается на B, так как для каждого числа мы пройдемся во его битам, чтобы найти старший присутствующий).

Разбор задачи «Йода принимает экзамен»

Первыми двумя запросами узнаем буквы для 1 и $maxw$ (обозначим их за $f(1)$ и $f(maxw)$). Затем до тех пор, пока очередная буква не будет равна $f(1)$, запускаем двоичный поиск границы самой правой известной нам буквы (то есть находим самую правую букву l_1, не равную $f(maxw)$, затем самую правую букву l_2, не равную l_1 и так далее, пока не получим, что очередная l_i равна $f(1)$). Общее количество запросов тем самым не превосходит $2 + 25\log_2(10^{18}) < 2 + 25 \cdot 60 < 1600$.

Разбор задачи «Красим и дополняем»

Обозначим $dp(v, 0)$ – мин стоимость покраски поддерева v, если стоимость цвета вершины v равна x, и $dp(v, 1)$ – аналогично для y.

Пусть cnt_x – сколько раз x встречается в массиве c и cnt_y – сколько раз встречается y.

Пусть нам нужно посчитать $dp(v, 0)$. Предположим, все потомки покрашены в цвет x. Если мы перекрасим какой-то в цвет y, получим дельту в $dp(u, 1) - dp(u, 0)$. Нужно, чтобы не более $cnt_x - 1$ потомков были покрашены в цвет x и не более cnt_y потомков были покрашены в цвет y. Отсортируем потомков по разнице и будем перебирать префикс, который одновременно означает количество потомков, покрашенных в цвет y.

$dp(v, 1)$ посчитаем аналогично.

Сложность – $O(nlogn)$

Разбор задачи «Леопольд и множества»

Множество должно состоять только из тех вершин, которые лежат в каком-то цикле(компоненты сильной связности размера больше 1). Таким образом, находим эти компоненты, ответ – $2^m - 1$, где m – суммарный размер сильно связных компонент размера больше 1. Поправка: стоит отдельно рассмотреть циклы длины 1,чтобы учесть этот случай. Они входят в ответ.