Задача A. Три пловца

Ответ равен $\min(\lceil \frac{t}{a} \rceil \cdot a, \lceil \frac{t}{b} \rceil \cdot b, \lceil \frac{t}{c} \rceil \cdot c) - t$.

Асимптотика: $O(1)$.

Задача B. Аквапарк

Для начала поймём, сколько сеансов будет до первой очистки бассейнов. Это $cnt_0 = min(\lceil m/a \rceil, \lfloor n/a \rfloor)$. После этого останется $max(0, n - cnt_0 \cdot a)$ времени.

Далее оценим сколько времени пройдёт до следующей очистки. Это $T = b + \lceil (m-b)/a \rceil \cdot a$, при этом за это время будет $cnt_1 = \lceil (m-b)/a \rceil$ сеансов.

Тогда, если разделить оставшееся время после первой очистки на T и домножить на cnt_1, получим сколько всего было сеансов между первой и последней очисткой бассейнов.

Далее оценим время после последней очистки бассейнов, вычтем оттуда b и разделим на a. Так мы получим сколько сеансов пройдёт после последней очистки.

Задача C. Максимальная ширина

Рассмотрим какую-то подпоследовательность строки s равную t. Обозначим за p_i позицию символа t_i в строке s. Для фиксированного i мы можем найти подпоследовательность, которая максимизирует $p_{i+1} - p_i$.

Пусть $left_i$ и $right_i$ — минимальное и максимальное возможное значение p_i среди всех допустимых p. Легко заметить, что максимально возможное значение $p_{i+1} - p_i$ равно $right_{i+1} - left_i$.

Чтобы вычислить $left_i$, нам просто нужно найти первый элемент после $left_{i-1}$, который равен t_i. Это можно сделать простым жадным алгоритмом или при помощи динамического программирования. $right_i$ можно найти таким же образом.

После нахождения $left$ и $right$, ответ легко считается как $\max_{i=1}^{i=n-1} right_{i+1} - left_i$.

Асимптотика: $O(n + m)$.

Задача D. Ход гения

В первой подгруппе можно было просто написать экспоненциальный перебор всех возможных чисел за $O(4^{a+b})$, или как-то умудриться перебрать ответ руками.

Все остальные группы представляют собой неоптимально написанные интерпретации полного решения.

Задача имеет несложное конструктивное решение. Пусть число $x = 111\ldots1100\ldots000$, зафиксируем его таким. Пусть $a > 0$ и $b > 1$, иначе конструкция ответа очевидна. Будем изменять y, пусть изначально $y = x$. Возьмем самую правую единичную цифру и будем двигать вправо. Далее обозначаем новую позицию последней единички как $last$.

Заметим, что для любого $k \in [0; a]$ его можно набрать, просто сдвигая последнюю единичку правее на k. Это видно из того, что на отрезке $[b; last]$ в числе $x - y$ будут стоять только единички, по причине переноса через разряд.

Если $k > a$, то сдвинем последнюю единичку в позицию $a + b$. Далее у нас есть префиксный отрезок из единичек. Пусть его размер больше одного и он не равен b. Тогда мы можем взять самую правую единицу из него и сдвинуть на одну позицию (ровно туда, где мы ранее не перешли через разряд при вычитании), таким образом мы сохраним переход через разряд в еще одной позиции, тем самым увеличив количество единичек в $x - y$ на один. Если k все еще больше числа, которое мы смогли получить - повторим сдвиг. Заметим, что таким образом мы можем получить любой ответ в отрезке $[0; a + b - 2]$.

Несложно доказать, что при $k > a + b - 2$ ответа не существует.

Задача E. Почти отказоустойчивая база данных

Если есть какой-нибудь ответ, то он отличается от всех n заданных массивов не более чем в одной позиции. Для нахождения ответа достаточно взять любой из этих n массивов и поменять в нём не более одного элемента, чтобы он стал ответом.

Возьмём первый массив из этих n массивов.

Если среди остальных $n-1$ массивов есть хотя бы один, отличающийся от первого массива более чем в двух позициях, то ответом будет «`No`», так как в таком случае ответ найти невозможно.

Если все остальные $n-1$ массивов отличаются от первого массива не более чем в одной позиции, то ответом будет «`Yes`» и первый массив без изменений.

Рассмотрим любой массив j, который отличается от первого в двух позициях. Нам нужно изменить первый массив так, чтобы массив j начал отличаться от первого только в одной позиции. Возможных вариантов для изменения всего два. После замены элемента нужно проверить, что ответ подходит (все массивы отличаются от первого не более чем в одной позиции).

Такое решение (с проверкой ответа) имеет сложность $O(n \cdot m)$.

Задача F. Древесные жабы (и лягушки)

Сначала формализуем задачу. За $d(a, b)$ обозначим крачайшее расстояние от вершины a до вершины b. Дано дерево на n вершинах. Надо ответить на запросы вида «найти сумму номеров всех вершин v, для которых $d(1, v) > d(u, v)$».

Подвесим дерево за вершину 1 (посмотрим на дерево как на корневое дерево с корнем в вершине 1). Пусть $w = lca(v, u)$ (w — наименьший общий предок вершин v и u, то есть самая глубокая вершина, являющаяся одновременно предком v и u).

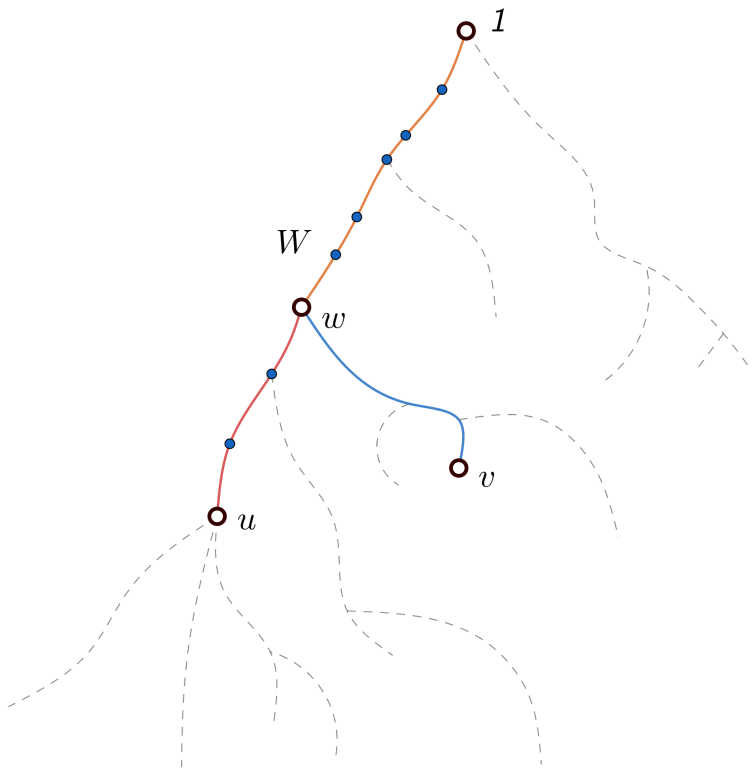

Рис. 1:

Мы хотим, чтобы выполнялось следующее неравенство:

$$d(u, v) < d(1, v)$$

Каждый из двух путей можем разбить на два пути (см. рис. 1): $d(u, v) = d(u, w) + d(w, v)$, $d(1, v) = d(1, w) + d(w, v)$. Значит, надо, чтобы выполнялось неравенство

$$d(u, w) + d(w, v) < d(1, w) + d(w, v)$$

Вычитая из обеих частей общее слагаемое $d(w, v)$, получаем, что

$$d(u, w) < d(1, w)$$

То есть, вершина w будет ближе к вершине u, чем к вершине 1. Найдём самую высокую вершину W такую, что $d(u, W) < d(1, W)$ (см. рис. 1). Из вышесказанного следует, что все вершины в поддереве вершины W ближе к вершине u, чем к вершине 1, а остальные – нет (см. рис. 2).

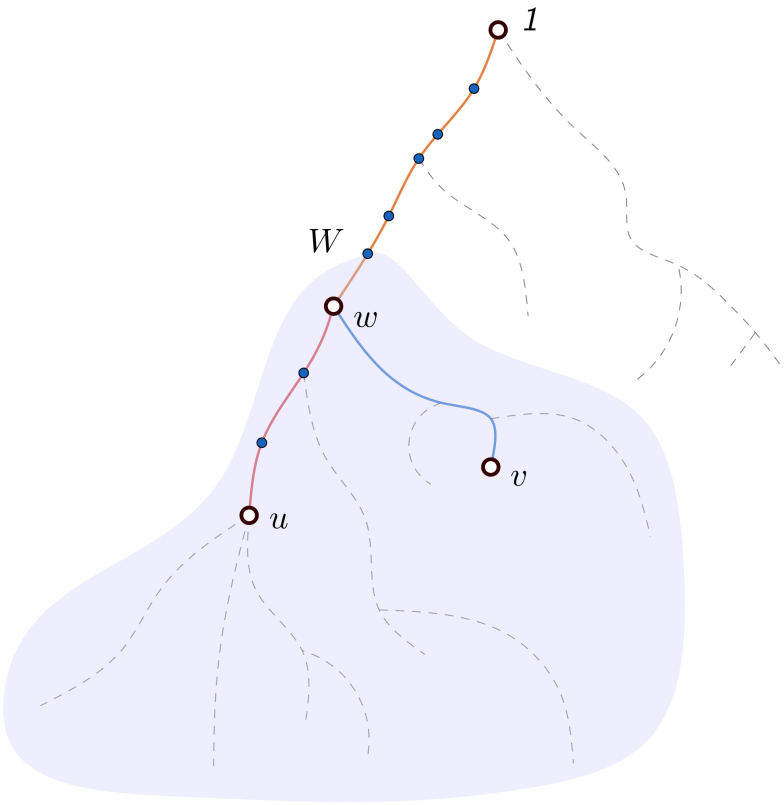

Рис. 2:

Значит, если для каждой вершины подсчитать сумму номеров вершин в её поддереве (делается с помощью тривиальной динамики по поддеревьям за $\mathcal{O}(n)$), а также для каждой вершины u найти соответствующую ей вершину W, то на один запрос можно будет отвечать за $\mathcal{O}(1)$. Единственный вопрос в том, как находить вершину W — для этого достаточно запустить обход в глубину и заметить, что вершина W всегда будет лежать на середине пути из корня в вершину u, то есть, если параллельно с обходом в глубину поддерживать путь из первой вершины в текущую рассматриваемую вершину u, то вершиной W будет являться вершина, лежащая в центре этого пути (с поправкой не более чем на одну позицию).

Задача G. Сложная задача

Воспользуемся методом динамического программирования.

Обозначим за dp_i минимальную необходимую высоту листа, в который можно вписать первые i формул. Тогда для пересчета такой динамики может использоваться следующая формула:

$dp_i = \min_j(dp_j + \max\limits_{x=j+1}^{i} h_x)$, где j принимает все возможные значения левой границы следующей строки, с учетом суммарной ширины всех прямоугольников. Иначе говоря, мы будем поддерживать окно индексов, через которые можно делать пересчет динамики.

Внутри этого окна назовем рекордами те индексы x, для которых h_x является максимумом на суффиксе в окне. Получается, что слева от каждого рекорда есть подотрезок элементов, для которых переход в динамике будет стоить $dp_j + h_x$. Пользуясь неравенством $dp_j \leqslant dp_{j+1}$, получаем, что у каждого рекорда есть подотрезок, у которого однозначно определена стоимость — это высота рекорда и прошлое значение динамики от левой границы отрезка.

Получаем такую задачу: есть окно, в котором хранятся подотрезки. У каждого подотрезка есть своя стоимость. Нужно находить самый дешевый подотрезок. Если посмотреть на пересчет окна, то можно заметить, что нам понадобится удалять рекорды с конца, добавлять рекорды в конец, удалять элементы из начала. Удаление элементов из начала будет либо удалять рекорд, либо изменять стоимость самого левого отрезка. Таким образом, достаточно реализовать такую структуру данных, как deque с поддержкой минимума (например, на двух стеках).