Задача A. Квнщики

В этой задаче нужно сделать именно то, что требуется в условии: посчитать, сколько суммарно времени провели за просмотром видео Макс и Мел, и сравнить эти два числа.

Задача B. Га-га 3

Если остаток a_1 по модулю 3 равен нулю или единице — то ответ 0 или 1 соответственно, вне зависимости от остальных чисел.

Если остаток a_1 по модулю 3 равен 2 — то посмотри на четность a_2: если a_2 четно — вне зависимости от остальных чисел, степень, в которую нужно возвести a_1 будет четная, значит ответ 1, если же a_2 нечетно, то результат также не зависит от остальных чисел, и ответ 2.

Задача C. Дедлайны

Первоначально преобразуем данные о каждой задаче в отрезки дней, в каждый из которых задача уже может быть сдана, то есть отрезки $tasks$ вида $[s_i + c_i, f_i]$ — от момента решения задачи до дня, по прошествии которого задача должна быть сдана. Затем отсортируем все отрезки по левому концу.

Задача решается жадностью: в каждый свободный день выбираем из всех задач, которые к этому дню уже решены, задачу, которая должна быть сдана раньше остальных. Затем переходим к следующему дню и проделываем то же самое.

Реализовывать будем следующим образом. Каждый правый конец отрезка по мере возможности сдачи соответствующей ему задачи в текущий день будем добавлять в множество. Текущий день храним в переменной j, еще не рассмотренный отрезок - в переменной i. Первоначально j соответствует минимальному левому концу среди левых концов всех отрезков. На каждой итерации цикла сначала добавляем все доступные для сдачи в текущий день j задачи (увеличивая при этом переменную i), а затем вынимаем из множества минимальный правый конец среди всех правых концов доступных задач. И если вынутый правый конец оказался меньше j, то есть текущего свободного дня, выводим `NO` и завершаем программу.

Иначе идем дальше. Если после извлечения элемента множество оказалось пусто, присваиваем переменной j значение левого конца еще не рассмотренного отрезка, то есть $j = tasks[i].left$. В ином случае, если множество не оказалось пустым, увеличиваем j на один. Затем переходим к следующей итерации.

После цикла выводим `YES`.

Задача D. Игрушки

Нужно проверить, что в массиве есть два либо ноль элементов, четность которых не совпадает с индексами, на которых они стоят.

Если таких элементов два, то проверяем, что четности их разные, и меняем местами.

Если таких элементов ноль, то меняем элементы, стоящие на 1 и 3 позициях. Частным случаем является случай при $n = 2$, для него невозможно произвести обмен в этом случае.

Задача E. Книжный шкаф

Выпишем условия для книги размера $size$ стоящей на полки с высотой h_i:
$\frac{height}{k} \leqslant h_i \leqslant height \Leftrightarrow h_i \leqslant height \leqslant h_i \cdot k$
$\frac{height}{m_1} \leqslant size \leqslant \frac{height}{m_2}$
Подставляя неравенство на $height$, получаем, что должны выполняться два условия:
$\frac{h_i}{m_1} \leqslant size \Leftrightarrow h_i \leqslant size \cdot m_1$
$size \leqslant \frac{h_i \cdot k}{m_2} \Leftrightarrow size \cdot m_2 \leqslant h_i \cdot k$
Проверяя эти два условия независимо для каждой книги, находим ответ.

Задача F. Прогулка по Новоуральску

Маршруты, о которых говорится в данной задаче — гамильтоновы пути. То есть пути, которые проходят через каждую вершину данного графа ровно по одному разу. Заметим, что если путь

w подходит, то «развернутый» путь $\widetilde{w}$ (вершины в котором следуют в обратном порядке) — тоже подходит, так как пути считаются различными, если последовательности вершин в них неодинаковы.

Тогда общее количество таких путей всегда четно. Кроме случая когда $n = 1$, тут ответ, очевидно, равен 1.

Задача G. Супер чемпион

Преобразуем неравенство из условия: $a_i + i < a_j + j$. Теперь нужно найти такое максимальное количество элементов массива, что для каждой пары элементов выполняется следующее: сумма значения и индекса одного элемента меньше, чем сумма значения и индекса другого.

Прибавим к каждому элементу массива его индекс и получим новый массив b. В полученном массиве нужно найти такое максимальное количество элементов, что в каждой паре значение одного элемента строго меньше значения другого, то есть необходимо просто посчитать длину наибольшей возрастающей последовательности в массиве b.

Задача H. Вилочные вычисления по ложечному модулю

Для начала узнаем, какие биты являются единичными в $x \oplus y$. Тогда мы сможем представить результат в виде суммы 2^i, по всем битам i, равным 1 в $x \oplus y$.

Не умоляя общности предположим, что $x > y$. Затем получим $2^0 = 1$, деля число y целочисленно на 2, пока не получим 1. На это у нас уйдет не более $O(\log(y))$ операций. Теперь достаточно с помощью операции умножить на 2, получить из 2^0 нужные степени двойки и сложить их. Нужных степеней двоек будет порядка $O(\log(x))$.

Для удобства обращения к памяти можно было сначала получить все степени двойки, а уже потом выполнять сложение.

Всего нужно будет хранить порядка $O(\log x)$ переменных.

Задача I. Подопытные крысы

Заметим, что описанная в условии конструкция представляет из себя граф, а именно — дерево. А две искомые вершины это его листы.

С помощью поиска в глубину, найдем для каждой вершины наименее глубокий лист среди ее потомков. Для это корнем дерева нужно взять не лист (число соседей больше одного).

Рассмотрим пути между листами. Утверждение: такой путь единственен (дерево) и сначала идет вверх до какого-то предка (наименьшего общего предка двух листов), а потом идет вниз, до второго листа.

Таким образом можно найти для каждой вершины два наименее удаленных друг от друга листа, путь между которыми «перегибается» в ней. Для этого посмотрим уже посчитанные наименее глубокие листы по каждому ребенку и если детей хотя бы два, то ответ это сумма расстояний до найденных листов.

Ответом же на задачу будет минимум по всем вершинам из найденных наименьших расстояний. Итоговая сложность $O(n)$.

Замечание: возможна сложность $O(n \log n)$, если для упрощения поиска двух минимумов из детей использовать быструю сортировку.

Задача J. Путь в неизвестность

Для решения задачи переберем ширину w чемодана и найдем по фиксированной ширине максимальную длину.

Будем делать это с помощью структуры данных очередь с максимумом, поддерживающей максимум среди всех элементов, находящихся в данный момент в ней. Заведем две таких очереди: up и $down$, при этом в up в любой момент времени будут хранится $w + 1$ (почему не w будет объяснено позже) последовательных a_i, в $down$ — b_i. Эти очереди будут отвечать за отрезок ширины пещеры, покрытый нашим чемоданом.

Пусть сейчас чемодан занимает столбцы с i-го по $i + w$, при сдвиге чемодана вправо, следует перестать учитывать столбец i и начать учитывать $i + w + 1$-й. Это легко поддерживать в очереди. Кроме того, в любой позиции, чемодан не может быть длиннее, чем $n - max(up) - max(down)$,

где $max(up), max(down)$ - максимумы в очередях. Иначе чемодан будет пересекать стену пещеры в каком-то из столбцов.

Тогда max - максимальная сумма $max(up) + max(down)$ на всех отрезках длины $w + 1$, и максимальной площадью чемодана для фиксированного w будет $n - max$.

В очереди следует каждый раз хранить сведения о $w + 1$ подряд идущих столбцах, так как при сдвиге чемодана вправо и вниз или вверх, мы **сначала** двигаем вниз или вверх и важно, чтобы после такого сдвига чемодан не выходил за пределы стен пещеры. То есть важно, чтобы чемодан не пересекал пещеру как на отрезке $[i, ..., i + w - 1]$, так и на отрезке $[i + 1, ..., i + w]$.

Задача K. Строка для друга

Составим граф из всех букв и проведем ребра между разными буквами минимальной ценой замены. Посчитаем минимальное расстояние в графе изменений алгоритмом Флойда-Уоршелла за $O(a^3)$, где a — размер алфавита. Расстояние между вершиной a и b в графе соответствует минимальному количеству монет, которое нобходимо, чтоб получить из символа a символ b.

Рассмотрим делители n, только такие числа являются кандидатами на k-строку из n символов. Таких подходящих k будет порядка $O(\sqrt[3]{n})$.

Будем решать для каждого k отдельно. Рассмотрим букву на позициях i, для каждого $i \, mod \, k$ посчитаем количество букв стоящих на таких позициях. Решаем независимо для каждого остатка от деления на k. Для этого переберем букву, которая будет стоять на этих позициях и посчитаем суммарную цену замены на необходимую букву.