Задача A. Большой удой

По условию T численно равно количеству литров молока, надоенного победителем. Обозначив за S количество литров молока, надоенного проигравшим, получаем, что ответ равен $T-S$, с другой стороны, верно $S = L-T$, где L – количество литров молока, надоенного обоими финалистами. При подстановке этого выражения в предыдущее получаем, что ответ равен $T - (L - T) = 2T - L$.

Задача B. Сделай 100

Подход к решению этой задачи может быть разный. Проще всего было начать, посмотрев на ноль. Цифра 0 не очень помогает в арифметике, поэтому имеет смысл слить её с четвёркой. Получим выражение с числом 40 на конце.

Чтобы получить 100, можно из цифр 1, 2, 3 получить 60 и прибавить 40; или получить 140 и вычесть 40. Но эти три цифры слишком маленькие для этого. Деление на 40 тоже не помогает. Давайте рассмотрим вариант умножения на 40.

Для этого нам нужно из цифр 1, 2, 3 получить 2.5. Без деления тут точно не обойтись. Если начнём выражение с $1/2$, то до 2.5 нам останется прибавить 2. Но если мы начнём с $-1/2$, то прибавив 3 сможем получить ровно 2.5.

Итак, один из возможных ответов — `(-1/2+3)*40`. Все остальные ответы являются вариацией этого.

Задача C. Вендомат

Чтобы не было сдачи, достаточно выбирать только те пачки чипсов, которые стоят не больше рублей и не больше копеек, чем есть у Вадима. Из всех таких нужно выбрать самую дорогую.

Задача D. На планете Иворил...

Заметим, что можно рассматривать каждое слово отдельно, найти минимальное количество ошибок, а затем просуммировать. Заведём три счётчика — два для существительных (которые начинаются с гласной и которые начинаются с согласной) и для глаголов. Тогда за один проход по строке, смотря на чётность позиции и гласность буквы, можно найти количество ошибок для каждого из вариантов слова. Из всех вариантов достаточно выбрать минимум.

Задача E. История версий

В условии задачи описан процесс прибавления единицы к каждому разряду номера версии. Нетрудно реализовать функцию $f(x)$, добавляющую 1 к каждому разряду. Для этого нужно посчитать количество цифр в числе x и прибавить к нему $(10^d - 1)/9$, где d — это количество цифр. Но в задаче от вас просят обратить эту операцию.

Пусть у нас есть число x, состоящее из d цифр. Нам нужно найти такое y, что $f(y) = x$. Заметим, что у y может быть d или $d-1$ цифр. Предположив количество цифр в y, получим два кандидата:

$$y_1 = x - \frac{10^d - 1}{9}, y_2 = x - \frac{10^{d-1} - 1}{9}.$$

Тогда y_1 подходит, если y_1 положительное и имеет d цифр; а y_2 подходит, если y_2 положительное и имеет $d-1$ цифр.

Если подходит ровно одно из чисел y_1 и y_2, то мы нашли номер версии, предшествующий x-й. Наша цель — построить такую цепочку из чисел, начиная с N, и ответом на задачу будет длина цепочки. Цепочка оборвётся, если ни одно из чисел y_1, y_2 не подходит. Возможно, вам придётся обработать отдельный случай $x = 1$, которому не может предшествовать ни одно число, так как номер версии всегда натуральный.

Наконец, что делать, если из чисел y_1 и y_2 подходят оба? Оказывается, это невозможно. По их формулам нетрудно понять, что $y_2 > y_1$. Но мы требуем, чтобы в записи y_2 было на одну цифру меньше, чем в y_1. Таким образом, цепочка никогда не ветвится: по номеру версии мы либо однозначно определим предыдущую, либо сделаем вывод, что эта версия была первой.

Задача F. Туристы, достопримечательности и телескопы

Заметим, что при увеличении силы, количество видных достопримечательностей из города строго возрастает. Поэтому можно для каждого города запустить бинарный поиск по ответу. Для того, чтобы быстро пересчитать сумму на отрезке $[l; r]$, можно использовать префикс-суммы.

Пусть p_i — сумма s_j с 1-го по i-й. Для удобства положим $p_0 = 0$. Тогда если у i-го города поставить силу телескопа x, из него будет видно $p_{min(i+x,n)} - p_{max(i-x-1,0)}$ достопримечательностей.

Задача G. Сапёр 1D

В этой задаче нужно угадать в каждой клетке первого ряда, находится ли там мина или нет. Нам понадобятся два ключевых утверждения:

1. Мы всегда можем узнать, есть ли мина в третьей слева клетке и в третьей справа клетке;
2. Если в некоторой клетке мы наверняка знаем, есть ли там мина или нет, то мы можем это узнать для клеток на 3 позиции левее и правее.

Этого достаточно, чтобы решить случаи $N = 3k$ и $N = 3k + 1$. То есть, если N при делении на 3 даёт остаток 0 или 1, то мы можем решить любое поле Сапёра, и ответ равен 2^N.

Если $N = 3k + 2$, то мы можем найти все мины в каждой третьей клетке, но этого недостаточно. Заметим, что если мы узнаем состояние ещё одной клетки, то это позволит решить головоломку. Посмотрим на все клетки, которые мы ещё не знаем: если две такие клетки подряд обе пустые или обе имеют мину, то мы сможем это определить, посмотрев на число под ними. Если же состояния неизвестных нам клеток чередуются, то мы не сможем решить головоломку; каждое число, которое мы откроем, будет указывать на две неизвестные клетки, и в них всегда будет в сумме 1 мина.

Построим общий вид нерешаемой головоломки. В ней $N = 3k + 2$, то есть N даёт остаток 2 при делении на 3. Каждая третья клетка может независимо друг от друга иметь или не иметь мину; всего 2^k вариантов. А все остальные клетки должны чередоваться, это дополнительный выбор из 2 вариантов: начинаем с мины или с пустой клетки. Итого, если $N = 3k+2$, то количество нерешаемых полей равно $2 \cdot 2^k$. Соответственно, количество решаемых полей равно $2^N - 2 \cdot 2^k$.

Так как ответ нужно вывести по модулю, рекомендуется использовать бинарное возведение в степень. Но N небольшое, поэтому можно возвести в N-ю степень с помощью цикла.

Вот пример поля, которое не решается однозначно. Можно узнать содержимое каждой третьей клетки в первой строке, но неоткрытые клетки могут быть как пустыми, так и заминированными.

Далее показаны два возможных расположения мин, которые могут быть на этом поле.

Задача H. Матч тысячелетия

Вначале нужно поделить все значения p_i на НОД. Очевидно, что количеством валунов в i-й куче в итоге станет $x \cdot p_i$, поэтому можно перебрать этот x. Посмотрим, как меняется время подготовки i-й кучи при увеличении x на 1. Есть три варианта:

1. Если $(x + 1) \cdot p_i \leqslant a_i$, то время уменьшится на p_i.

2. Если $x \cdot p_i \geqslant a_i$, то время увеличится на p_i.

3. В противном случае время увеличится на $((x+1) \cdot p_i - a_i) - (a_i - x \cdot p_i)$, что можно сократить до $p_i - 2 \cdot (a_i - x \cdot p_i) = p_i - 2 \cdot (a_i \bmod p_i)$

Это означает, что можно обновлять время сразу для всех куч суммарно, кроме переходных моментов в третьем случае. Обозначим эти моменты как x_i, тогда $x_i = a_i \operatorname{div} p_i$, и мы должны остановиться во всех значениях x из x_i и $x_i + 1$.

Отсортируем по значению x_i все кучи и запустим нечто вроде метода сканирующей прямой, чтобы перебрать все значения x по порядку, и для каждого узнать ответ. Будем поддерживать текущее число x, суммарное время T, и коэффициент d. Коэффициент d будет отвечать за изменение T при изменении x.

Начнём с $x = 0$, $T = \Sigma a_i$, и $d = -\Sigma p_i$. Это соответствует тому, что мы можем в каждой кучке собрать $0 \cdot p_i$ камней, на это уйдёт T времени, и при увеличении x наше время будет уменьшаться на Σp_i за каждую единицу сдвига x. Но это время подготовки не должно учитываться в ответе, потому что кучки должны быть непустые.

Затем будет постепенно увеличивать x, рассматривая только интересные нам точки x_i и $x_i + 1$. Найдя следующую точку $x_i > x$, присвоим $T = T + d \cdot (x_i - x)$. Затем нужно будет перейти в $x_i + 1$ и пересчитать T и d, так как у нас теперь камней в i-й куче станет больше текущего количества, а значит потраченное на эту кучу время будет расти. Значение d увеличится на $2p_i$, а T можно посчитать вычтя время для x_i и прибавив для $x_i + 1$, или по вышеуказанной формуле.

Докажем, что для ответа нам достаточно посмотреть T во всех x_i и $x_i + 1$. Пусть это не так, и оптимальный ответ T_{ans} был в $x_l + 1 < x_{ans} < x_r$. Но между $x_l + 1$ и x_r значение d не менялось — на протяжении всего интервала размер каждой кучки либо монотонно приближался к a_i, либо удалялся. Значит, величина T на этом интервале изменялась линейно, а линейная функция на отрезке всегда достигает максимума в одном из концов. А именно, при положительном d время $T_l < T_{ans}$, при отрицательном d — время $T_r < T_{ans}$, а при $d = 0$ — время $T_l = T_{ans} = T_r$.

В итоге, суммарно переход (третий вариант) сделается ровно один раз для каждой кучки, а поэтому сложность алгоритма равна $O(N \cdot \log N)$.

Альтернативное решение:

Аналогично предыдущему решению, поделим все p_i на их НОД, и будем искать лучшее значение x. Пусть $f_i(x) = |a_i - p_i \cdot x|$ — время, нужное чтобы привести i-ю кучу в порядок при заданном коэффициенте x. Заметим, что все функции f_i битонные; то есть, до $x = \frac{a_i}{p_i}$ они строго убывают, затем строго возрастают. Ответ равен сумме $\Sigma f_i(x)$. Можно показать, что эта сумма является выпуклой функцией от x. А значит, можно найти минимум с помощью тернарного поиска.

Задача I. Кусочно-линейные функции

Для удобства будем считать, что все $a_i \geqslant 0$. Если $a_i < 0$, то можно заменить знак у a_i и b_i на противоположный, и значение $|a_i x + b_i|$ не поменяется.

Введём обозначение

$$c_i = -\frac{b_i}{a_i}.$$

Выражение с модулем $|a_i x + b_i|$ можно записать как $|a_i(x - c_i)|$. Если $x \geqslant c_i$, то модуль раскроется как $a_i \cdot x - a_i \cdot c_i$, иначе — как $-a_i \cdot x + a_i \cdot c_i$.

Пока забудем про свободные члены $\pm a_i \cdot c_i$ и посмотрим на коэффициент при x. На отрезке между x_i и x_{i+1} коэффициент при x должен быть равен

$$\frac{y_i - y_{i+1}}{x_i - x_{i+1}}.$$

Давайте создадим $n - 1$ слагаемых $\pm|a_i(x - c_i)|$, в качестве c_i взяв значения $x_1, x_2, \ldots, x_{n-1}$. Раскроем все модули. Тогда на отрезке $x \in [x_1, x_2]$ только первое выражение $|a_1(x - c_1)|$ раскроется с плюсом, а остальные — с минусом. На отрезке $x \in [x_2, x_3]$ первые два выражения $|a_1(x - c_1)|$ и

$|a_2(x - c_2)|$ раскроются с плюсом, а остальные — с минусом. Продолжим это, и получим систему уравнений относительно коэффициентов при x:

$$\begin{aligned}
a_1 - a_2 - a_3 - \ldots - a_{n-1} &= \frac{y_1 - y_2}{x_1 - x_2}, \\
a_1 + a_2 - a_3 - \ldots - a_{n-1} &= \frac{y_2 - y_3}{x_2 - x_3}, \\
a_1 + a_2 + a_3 - \ldots - a_{n-1} &= \frac{y_3 - y_4}{x_3 - x_4}, \\
&\ldots \\
a_1 + a_2 + a_3 + \ldots + a_{n-1} &= \frac{y_{n-1} - y_n}{x_{n-1} - x_n}.
\end{aligned}$$

Нужно найти из этой системы все значения a_i. Это нетрудно сделать для $a_2, \ldots, a_{n-1}$, посмотрев на разности соседних строчек. Например, вычтя из второй строки первую, получим

$$2a_2 = \frac{y_2 - y_3}{x_2 - x_3} - \frac{y_1 - y_2}{x_1 - x_2}.$$

Это позволит нам явно найти все значения a_i, кроме a_1. Подставив их в одно из уравнений, можно выразить a_1.

В процессе мы можем получить отрицательные значения a_i. Но в начале мы договорились, что все $a_i \geqslant 0$. Вместо того, чтобы использовать отрицательное a_i, возьмём выражение $-|a_i(x - c_i)|$ со знаком минус. Тогда a_i раскроется с противоположным знаком.

Итак, мы получили модульную функцию, которая имеет те же коэффициенты при x, что и кусочно-линейная функция, на всех подотрезках $[x_i, x_{i+1}]$. Это значит, что текущая функция отличается от искомой на константу. Можно явно посчитать значение в одной точке, затем прибавить последнее слагаемое $|0x + b_n|$ или $-|0x + b_n|$, чтобы завершить решение.

Альтернативное решение

Пусть $f(x)$ — данная нам кусочно-линейная функция, $g(x)$ — модульная функция, которую мы строим, $h(x) = f(x) - g(x)$. Давайте постепенно строить $g(x)$ с целью сделать $h(x)$ тождественно равной нулю.

Посмотрим на производные кусочно-линейной функции $f(x)$ на её отрезках. Имеется $n-1$ отрезков непрерывности, обозначим производные на них как $k_1, k_2, \ldots, k_{n-1}$. Аналогично предыдущему решению, получим формулу

$$k_i = \frac{y_i - y_{i+1}}{x_i - x_{i+1}}.$$

Посмотрим на точку излома x_i, $2 \leqslant i \leqslant n - 1$. В окрестности слева от неё производная равна k_{i-1}, справа — k_i. Мы хотим создать такой же излом в функции $g(x)$. Это не всегда можно сделать с помощью одного слагаемого, но мы может обеспечить скачок производной на величину $k_i - k_{i-1}$. Если $k_i \geqslant k_{i-1}$, добавим в $g(x)$ слагаемое

$$|0.5(k_i - k_{i-1}) \cdot (x - x_i)|,$$

а если $k_i < k_{i-1}$, то добавим слагаемое

$$-|0.5(k_i - k_{i-1}) \cdot (x - x_i)|.$$

После добавления всех $n - 2$ таких слагаемых, получим следующее: $g(x)$ до x_1 имеет какие-то значения и какую-то производную, это нами не учтено. Но в каждой точке излома $x_2, x_3, \ldots, x_{n-1}$ производная функций $f(x)$ и $g(x)$ изменяется на одно и то же значение. Из этого следует, что на данный момент функция $h(x) = f(x) - g(x)$ является линейной на отрезке $[x_1, x_n]$.

Осталось добавить к $g(x)$ два слагаемых, который будут тождественно равны $h(x)$ на отрезке $[x_1, x_n]$. Точное значение $h(x)$ можно узнать, явно посчитав $f(x) - g(x)$ в точках x_1 и x_n. Теперь нужно с помощью двух слагаемых $\pm|a_{n-1}x + b_{n-1}| \pm |a_n x + b_n|$ построить произвольную линейную функцию на отрезке $[x_1, x_n]$. Это можно сделать, например, слагаемыми вида

$$\pm|a_{n-1}(x - x_1)| \pm |0x + b_n|.$$

Подробности мы оставляем на раздумье читателям.

Задача J. Мой дед

Заметим, что если в k-й день ответ «YES», то существует путь от 1 до N, что для каждой тропинке i на этом пути соблюдается неравенство $s_i \cdot a_k > w_i \cdot b_k \Leftrightarrow \frac{a_k}{b_k} > \frac{w_i}{s_i}$. Давайте для каждого ребра запомним величину $\frac{w_i}{s_i}$. По этой величине можно определить, существует ли путь в k-й день: мысленно удалим все рёбра, для которых $\frac{w_i}{s_i} \geqslant \frac{a_k}{b_k}$. Если после этого путь от 1 до N существует, то ответ в этот день «YES», иначе «NO».

Теперь рассмотрим для каждого дня j значение $d_j = \frac{a_j}{b_j}$. Отсортируем все дни по этому значению. Если рассматривать их в порядке возрастания и так же мысленно удалять рёбра, то можно считать, что в этом порядке в графе появляются новые рёбра, а старые всегда остаются. Это значит, что для каких-то первых нескольких дней в этом порядке (возможно, ни для каких) ответ «NO», а для всех остальных (возможно, ни для каких) — «YES». Поэтому можно сделать бинарный поиск по дням и найти эту границу. Для проверки нужно строить новый граф, удалять не подходящие рёбра и проверять существование пути от 1 до N поиском в глубину или в ширину. Мы получим величину $\frac{a}{b}$ такую, что на все дни с $\frac{a_j}{b_j} \leqslant \frac{a}{b}$ ответ «NO», а для остальных — «YES». После этого выведем ответ в исходном порядке. Сложность такого решения равна $O((N + M + Q) \cdot \log Q)$.

Альтернативное решение

Отсортируем рёбра по величине $\frac{s_j}{w_j}$. Поскольку на каждой полянке грибы и ягоды продаются по одной и той же цене, то для проверки корректности пути достаточно запомнить ребро с самым маленьким соотношением $\frac{s_j}{w_j}$.

Воспользуемся динамическим программированием $dp[v]$ — наименьшее ребро на лучшем пути до v. Под наименьшим ребром мы подразумеваем номер ребра с наименьшим значением $\frac{s_j}{w_j}$ в порядке сортировки. Массив $dp[v]$ можно заполнить в порядке топологической сортировки графа, это похоже на алгоритм Дейкстры на минимум.

Пусть $dp[N] = e$, то есть в лучшем пути до N-й вершины обязательно встретится ребро e или хуже. Это значит, что мы гарантированно пройдём по ребру с соотношением $\frac{s_j}{w_j}$ не лучше $\frac{s_e}{w_e}$. По формуле из первого решения получим, что для k-го дня ответ «YES», если $\frac{a_k}{b_k} > \frac{s_e}{w_e}$. Это позволяет нам сразу отвечать на все запросы. Сложность такого решения равна $O(N + M \cdot \log M + Q)$.

Задача K. Старобарский рэп

Заметим несколько ключевых фактов:

1. Можно гарантировать ответ не меньше 0.5 — взять суффикс длины 1.
2. Не имеет рассматривать большие суффиксы, если у нас уже есть $d = 1 + \log_2 |s_i|$ различий. Даже если взять всю строку с d различиями, получим величину рифмы $\frac{|s_i|}{2^d} <= 0.5$.
3. Если рассмотреть два таких суффикса с длинами $l_1 < l_2$, что количество различий в них совпадает, то всегда выгоднее взять более длинный.

Из этого следует основная идея решения: сравнивая два слова s_i и s_j, имеет смысл рассмотреть только $1 + \log_2 min(|s_i|, |s_j|)$ наименьших суффиксов, следующая буква после которых различается. Среди них посчитать наибольшую величину рифмы и вывести.

Можно развернуть все строки, и для каждой строки $s = c_1 c_2 c_2 \dots c_l$ посчитать полиномиальные хэши для всех префиксов: $(c_1 + c_2 A + c_3 A^2 + \dots + c_k A^{k-1}) \mod M$. Теперь символы отрезаются

с начала строк, а не с конца, и можно легко проверять две подстроки с одинаковым количеством отрезанных символов на равенство. Чтобы сравнить префикс длины a строк s_i и s_j с c отрезанными символами, нужно у обеих строк посчитать разность хэшей $(a + c)$-го префикса и c-го префикса. Если они равны, то и соответствующие подстроки с большой вероятностью равны.

Осталось у двух строк начиная с c-го символа найти первые d отличий. Каждое такое отличие ищется бинарным поиском: начиная с позиции сразу после предыдущего отличия, найдём подстроку наибольшей длины, совпадающую у обеих строк. Как было описано ранее, это позволяет нам найти ответ на запрос. Так как мы ограничили d логарифмом длины, сложность такого решения равна $O(N + \Sigma|s_i| + Q \cdot \log^2(\max(|s_i|)))$.