Разбор задачи «Мясо и капуста»

Автор задачи:	Михаил Иванов
Подготовка тестов и решений:	Михаил Иванов
Автор разбора:	Михаил Иванов

Будем разбивать все продукты на блоки подряд идущих одинаковых. Тогда после каждого раза, как Алина и Ваня меняются местами, они могут съесть лишь крайние блоки, то есть уменьшить их количество не более чем на два. Разберёмся с тем, что происходит до первого обмена. Мысленно положим слева от всех продуктов кусок мяса, а справа — капусту: это не повлияет на ответ, поскольку Алина и Ваня могут сразу же съесть добавленные продукты. Если теперь на столе n блоков подряд идущих одинаковых продуктов, то n чётно, так как первый и последний блок состоят из разных продуктов. Тогда до первого обмена Алина и Ваня тоже могут уменьшить число блоков максимум на два. При этом до первого обмена, а также после каждого обмена можно действительно уменьшить число блоков на два, поскольку и перед Алиной, и перед Ваней будет тот тип продуктов, который они могут есть. Значит, после добавления куска мяса и кочана капусты надо вычислить число блоков n, и ответом будет $\frac{n}{2}$.

Разбор задачи «Глеб и задачи»

Автор задачи:	Глеб Лобанов
Подготовка тестов и решений:	Глеб Лобанов
Автор разбора:	Глеб Лобанов

Первым делом заметим, что если длина строки равна 1, то мы никак не сможем переставить задачи, а следовательно первый человек снова будет решать первую задачу. Если же длина строки больше 1, то так как все задачи различны, мы можем просто сделать циклический сдвиг массива. Тогда первый человек получит вторую задачу, второй — третью, и так далее, а следовательно, ни один человек не получит ту же задачу во второй раз.

Также можно было раз сделать случайную перестановку и проверить, подходит ли она. Вероятность того, что случайная перестановка подходит, стремится с ростом n к величине $\frac{1}{e}$.

Разбор задачи «Глеб и электрокар»

Автор задачи:	Глеб Лобанов
Подготовка тестов и решений:	Евгений Тушканов
Автор разбора:	Глеб Лобанов

Запустим алгоритм Дейкстры несколько раз: от каждой заправки, а также от начальной вершины и от конечной вершины. Теперь для каждой вершины мы знаем, до каких заправок мы можем добраться за k километров. Давайте построим новый граф, в котором вершины — заправки и города u и v (стартовый и конечный города), а любое ребро (a, b) означает, что от города a можно добраться до города b за k или меньше километров. Тогда путь в новом графе от u до v — путь между стартовой и конечной вершиной в изначальном графе, в котором все заправки расположены на расстоянии меньше k, при этом количество рёбер в таком пути — количество раз, когда мы должны заправиться. Найдём кратчайший путь в новом графе между стартовой и конечной вершиной, его длина и будет ответом. Оценим асимптотику: вторая часть работает за $O(M + N)$, первая же работает за $O(N \cdot M \log(M))$ (из каждой вершины мы запускаем алгоритм Дейкстры). Также существует альтернативное решение динамикой за $O(K \cdot M \log(M))$.

Для первой группы тестов можно просто перебрать все перестановки и для каждой узнать, можно ли добраться до конца и сколько раз надо будет заправиться.

Для третьей группы можно было сразу построить нужный нам граф, просто проверив рёбра между соседними вершинами.

Разбор задачи «Глеб и MTG»

Автор задачи:	Глеб Лобанов
Подготовка тестов и решений:	Глеб Лобанов
Автор разбора:	Глеб Лобанов

Давайте возьмём $s = -10^9$ и $d = 1$. Докажем, что это всегда ответ. Так как a_i — целое, то $|a_i|$ входит в арифметическую прогрессию с $s = 0$ и $d = 1$. Тогда a_i входит в $s = 0$, $d = 1$ или в $s = 0$, $d = -1$. У нас есть ограничение, что $-10^9 \leqslant a_i \leqslant 10^9$. Значит, вторую прогрессию можно заменить на $s = -10^9$, $d = 1$. Итоговый ответ: $s = -10^9$, $d = 1$.

Разбор задачи «Маленькие знаменатели»

Авторы задачи:	Михаил Иванов Владислав Макаров
Подготовка тестов и решений:	Михаил Иванов
Автор разбора:	Михаил Иванов

Как известно, все положительные числа лежат в ряду Фарея между дробями $\frac{0}{1}$ и $\frac{1}{0}$ (для подробностей обратитесь к статье на Википедии). Будем хранить в приоритетной очереди набор непересекающихся интервалов, на которых лежат наши дроби (изначально это один интервал $\left(\frac{0}{1}; \frac{1}{0}\right)$). Приоритетом интервала $\left(\frac{a}{b}; \frac{c}{d}\right)$ будет пара $(b + d, a + c)$ (из теории ряда Фарея известно, что у всех таких интервалов приоритеты будут различными). Пока мы не нашли нужное число дробей, рассмотрим интервал $\left(\frac{a}{b}; \frac{c}{d}\right)$ с наименьшим приоритетом и рассмотрим дробь $\frac{a+c}{b+d}$. Если у неё слишком большой знаменатель, то процесс заканчивается. Если она лежит в данном по условию отрезке, то мы её добавим к ответу, а интервал заменим на два интервала $\left(\frac{a}{b}; \frac{a+c}{b+d}\right)$ и $\left(\frac{a+c}{b+d}; \frac{c}{d}\right)$. Если же не лежит, то пусть, не умаляя общности, наш отрезок полностью справа от дроби $\frac{a+c}{b+d}$. Тогда мы заменим наш интервал на $\left(\frac{a+kc}{b+kd}; \frac{c}{d}\right)$, где k — наибольшее положительное целое число, для которого дробь $\frac{a+kc}{b+kd}$ всё ещё левее отрезка (его можно найти с помощью линейного уравнения). Нетрудно понять, что так всякая дробь, содержавшаяся в ответе и в нашем интервале, останется в нашем интервале.

Для того, чтобы сдать задачу на полный балл, поиск такого k и сравнения дробей надо производить с использованием 128-битного целочисленного типа данных или длинной арифметики.

Разбор задачи «Произведение случайных чисел»

Автор задачи:	Иван Казменко
Подготовка тестов и решений:	Иван Казменко
Автор разбора:	Иван Казменко

Meet-in-the-middle.

Мы хотим найти пару множеств с произведениями (u, v), для которой $u \cdot v = x \pmod p$.

Будем идти по числам последовательности слева направо и поддерживать все множества двух типов: множества первого типа состоят только из элементов последовательности с чётными индексами, а второго типа — только из элементов с нечётными индексами.

Например, изначально есть по одному множеству каждого типа: это пустое множество. После рассмотрения a_1 в первом типе только $\varnothing$, а во втором — $\varnothing$ и $\{a_1\}$. После рассмотрения a_2 в первом типе $\varnothing$ и $\{a_2\}$, а во втором — $\varnothing$ и $\{a_1\}$. После рассмотрения a_3 в первом типе $\varnothing$ и $\{a_2\}$, а во втором — $\varnothing$, $\{a_1\}$, $\{a_3\}$ и $\{a_1, a_3\}$. И так далее.

Каждый раз, когда мы добавляем новое множество S какого-то типа — пусть произведение в нём равно u — нужно проверить, нет ли среди множеств другого типа множества T с произведением $(x \cdot u^{-1}) \bmod p$. Если есть, то множество $S \cup T$ — искомое.

Проверка делается хранением всех u и всех v в двух структурах с быстрым поиском — например, `set`. По парадоксу дней рождения (о нём можно почитать, например, в Википедии) количество множеств, которые нам придётся рассмотреть, пока не найдём искомое, будет порядка $\sqrt{p}$.

Разбор задачи «Отмерь и отрежь»

Автор задачи:	Андрей Райский
Подготовка тестов и решений:	Андрей Райский Владислав Макаров
Автор разбора:	Владислав Макаров

Есть такая простая динамика за $O(n^3)$: $dp[\ell][r]$ есть наименьшая длина строки, которая может быть получена из строки $s[\ell, r) = s_\ell s_{\ell+1} \dots s_{r-1}$ за ноль или более операций (ответ — $dp[0][|s|]$). Тогда $dp[\ell][r] = \min\limits_d \min(dp[\ell][r-d], dp[\ell+d][r])$, где d пробегает множество всех таких d от 1 до $r-\ell-1$, что строки $s[\ell, \ell+d)$ и $[r-d, r)$ совпадают (если же таких d нет вообще, то $dp[\ell][r] = r-\ell$, так как невозможно сделать ни одной операции). В дальнейшем, если строка q является префиксом и суффиксом строки p, а также $0 < |q| < |p|$ (то есть строка q непуста и не равна всей строке p), будем говорить, что q – **бордер** p.

Для того, чтобы получить решение за куб, достаточно вычислять значения $dp[\ell][r]$, скажем, в порядке увеличения длины (то есть $r-\ell$), перебирать все d от 1 до $r-l-1$ и проверять строки $s[l, l+d)$ и $s[r-d, r)$ на равенство с помощью хэшей, z-функции или других стандартных строковых алгоритмов, о которых можно почитать на википедии, e-maxx.ru, викиконспектах ИТМО и других подобных сайтах (ну или в хорошем учебнике по алгоритмам).

Такое решение позволяет пройти первую группу тестов, но для того, чтобы получить полный балл, его недостаточно. Есть несколько подходов, приводящих к решению за $O(n^2)$, опишем самый простой из них с точки зрения реализации. В вышеописанном решении мы перебираем очень много разных значений d, когда пытаемся посчитать $dp[\ell][r]$. Хотелось бы как-то ограничить перебор d на какое-то меньшее множество возможных значений. Наверное, самая естественная идея – всегда отрезать самый длинный бордер. Однако, как показывает второй пример из условия, такая стратегия не всегда оптимальна. Более того, не работают идеи вида «попытаться отрезать 10 самых длинных бордеров» и большинство вариаций на эту тему.

Чтобы прийти к решению, нужно сделать, на первый взгляд, контринтуитивный шаг: нужно отрезать не самый длинный бордер, а **самый короткий**. Покажем, что это всегда работает. Для этого предположим, что у строки $s[\ell, r)$ есть бордеры длин d_1 и d_2, где $0 < d_1 < d_2 < r-\ell$, при этом бордер длины d_1 – самый короткий из бордеров строки $s[\ell, r)$. Пусть мы отрезали бордер длины d_2, не умаляя общности, с конца. Тогда покажем, что мы могли вместо этого сперва отрезать бордер длины d_1 с конца, а потом бордер длины $d_2 - d_1$, опять с конца. Назовём наш бордер длины d_1 буквой q, а бордер длины d_2 – буквой p. Так как p и q – префиксы $s[\ell, r)$ и $|q| < |p|$, то q – префикс p. Аналогично, q – суффикс p, то есть q – бордер p.

Заметим, что любой бордер строки q, является также префиксом и суффиксом строки $s[\ell, r)$, а следовательно, и её бордером.

Теперь, если $2|q| > |p|$ (удвоенная длина q больше длины p), то префикс и суффикс p длины $|q|$ (напомним, что они оба равны q как строки) имеют непустое пересечение. Более того, строка, образованная этим пересечением, является одновременно префиксом и суффиксом строки q, а значит, и $s[\ell, r)$, что противоречит минимальности q.

Таким образом, $2|q| \leqslant |p|$ и строка q имеет вид qwq, где w – какая-то, возможно, пустая, строка. Тогда у строки $s[\ell, r)$ есть префикс и суффикс, равные $p = qwq$. Мы удалили p с конца, но вместо этого мы могли сперва удалить q, а потом qw и попасть в ту же самую строку. Значит, не было смысла торопиться и сразу удалять длинный бордер, можно было удалить и самый короткий.

Итак, мы почти добились того, чего хотели, только остался один вопрос: как быстро находить самый маленький бордер? Для этого нужно воспользоваться **префикс-функцией** (места, где о ней можно почитать, уже упоминались выше в разборе): для каждого ℓ найдём префикс функцию строки $s[\ell, |s|)$. Так мы получим длину **самого длинного** бордера для каждой из строк $s[\ell, r)$. Чтобы

получить длину **самого короткого** бордера, заметим, что этот самый короткий бордер будет либо самым коротким бордером самого длинного бордера $s[\ell, r)$ (если у самого длинного бордера $s[\ell, r)$ вообще есть бордеры), или просто самым длинным (и единственным) бордером строки $s[\ell, r)$ (если у самого длинного бордера $s[\ell, r)$ нет своих бордеров). Таким образом, длины **самых коротких** бордеров для всех строк $s[\ell, r)$ можно насчитать динамикой за $O(|s|-\ell+1)$ одновременно с подсчётом префикс-функции для строки $s[\ell, |s|)$. Итого, мы получили решение за $O(n^2)$.

Замечание для хорошо разбирающихся в префикс-функции: если вы привыкли думать о префикс-функции, как об ориентированном дереве, то самый короткий бордер для префикса строки, соответствующего вершине v дерева, соответствует первой вершине после корня (пустой строки) на пути вниз от корня до v.

Тем, кому понравилась эта задача, предлагаю порешать её усложнённую версию: при равенстве итоговых длин минимизировать ещё и количество сделанных операций. Её тоже можно решить за $O(n^2)$ и решение по-своему красивое, пусть и более сложное в реализации.

Разбор задачи «Посольства»

Автор задачи:	Михаил Иванов
Подготовка тестов и решений:	Владислав Макаров
Автор разбора:	Михаил Иванов

В каждом государстве находится $n-1$ посольств — по одному от каждого государства. Всего государств n, итого посольств $\underbrace{(n-1)+(n-1)+\ldots+(n-1)}_{n \text{ слагаемых}} = n(n-1)$. Так как ответ может быть около 10^{18}, но не больше, умножение в 64-битном целочисленном типе даёт правильный ответ, а вот 32-битного типа не хватает (и решения с 32-битным умножением получали неполный балл за задачу).

Разбор задачи «Компоненты связности»

Автор задачи:	Иван Казменко
Подготовка тестов и решений:	Иван Казменко
Автор разбора:	Иван Казменко

Вероятность того, что количество компонент связности больше 20, очень мала. Поэтому просто выведем граф из стольких связных компонент, сколько их в исходном графе.

Для этого, например, найдём компоненты связности и для каждой запишем порядок обхода вершин, полученный поиском в ширину или в глубину заодно с самим поиском компонент. Далее из каждого такого списка выберем первую вершину, затем вторую (если она есть), и так далее, пока не наберётся 20 вершин.

При таком решении во втором запуске можно использовать ровно тот же алгоритм, что и в первом — различия будут только при выводе ответа.

Разбор задачи «Конструктивное доказательство»

Автор задачи:	Иван Казменко
Подготовка тестов и решений:	Иван Казменко
Автор разбора:	Иван Казменко

Решение 1: пронумеровать объекты. Преобразовать объект в номер и номер в объект. Требуется только `int64`.

Решение 2: построить естественную биекцию. Либо в сочетании есть n, и тогда оставшаяся часть — левый ответ. Либо нет n, и тогда оно целиком — правый ответ. Восстановление: добавим n к левому ответу, а правый ответ выведем без изменений.