Задача A. Сколько нулей?

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Вам даны n чисел: i-е число равно $a_i \cdot 10^{b_i}$. На сколько нулей оканчивается сумма всех этих чисел?

Формат входных данных

В первой строке дано целое число n — количество чисел ($1 \leqslant n \leqslant 2 \cdot 10^5$). В i-й из следующих n строк даны два целых числа a_i и b_i ($1 \leqslant a_i \leqslant 10^9$, $0 \leqslant b_i \leqslant 10^9$).

Формат выходных данных

Выведите одно целое число — ответ на задачу.

Система оценки

Тесты к этой задаче состоят из пяти групп. Баллы за каждую группу ставятся только при прохождении примера, а также всех тестов, подходящих под ограничения этой группы.

Во всех тестах первой группы $n \leqslant 100$, все $b_i = 0$, а сумма всех чисел $a_i \cdot 10^{b_i}$ не превосходит 10^9. За прохождение всех тестов первой группы можно получить 5 баллов. Ещё раз напомним, что пример для этого тоже надо пройти.

В тестах второй группы $n \leqslant 1000$, а сумма всех чисел $a_i \cdot 10^{b_i}$ не превосходит 10^9. За вторую группу можно получить ещё 6 баллов.

В тестах третьей группы $n \leqslant 1000$ и все $b_i \leqslant 1000$. За третью группу можно получить ещё 27 баллов.

В тестах четвёртой группы все $b_i \leqslant 2 \cdot 10^5$. За четвёртую группу можно получить ещё 28 баллов.

На тесты пятой группы не накладывается никаких дополнительных ограничений. За неё можно получить оставшиеся 34 балла.

Пример

стандартный ввод	*стандартный вывод*
4 2 1 20 0 6 1 100500 3	2

Пояснение к примеру

В примере получается число $2 \cdot 10 + 20 \cdot 1 + 6 \cdot 10 + 100\,500 \cdot 1000 = 100\,500\,100$.

Задача B. Бочка и песочные часы

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Это интерактивная задача.

Ваня — ретроград. Он предпочитает не спешить в гонке со временем, и это касается всех аспектов его жизни. В частности, у него дома нет никаких современных таймеров, электронных часов — чтобы отмерять время, он пользуется своими n песочными часами. В i-х из них песок полностью пересыпается из верхней половины в нижнюю за t_i секунд. Мы будем считать, что песок сыплется из одной части в другую с постоянной скоростью, не зависящей от того, как перевёрнуты часы; в частности, например, если после $s < t_i$ секунд преждевременно перевернуть часы, то песок пересыплется обратно за те же s секунд, за которые он туда засыпался. Для удобства пользования Ваня пронумеровал часы так, что $t_1 \leqslant t_2 \leqslant \ldots \leqslant t_n$. Эти песочные часы так глубоко проникли в Ванину жизнь, что он всё меряет своими песочными часами: скажем, крепкий сон — это $3562t_1$, а одна партия в его любимую настольную игру — $360\,(t_1 + t_2)$. Обычные же единицы измерения времени он уже позабыл, так что, в частности, значения t_i он восстановить не может. Ну и зачем?

На завтрак Ваня любит яичницу. Чтобы её пожарить, надо продержать сырые яйца на сковородке T секунд. Как водится, Ваня не помнит, чему равно T и сколько длится секунда, но зато он помнит, что $T = k_1t_1 + k_2t_2 + \ldots + k_nt_n$, где все k_i — целые неотрицательные числа. Поэтому дело осталось за малым — k_1 раз прождать промежутки по t_1 секунд, переворачивая каждый раз первые часы, когда в них песок досыпается донизу, потом k_2 раз прождать t_2 секунд при помощи вторых часов, и так далее.

Ванин шаловливый брат, пока тот спал, подшутил над ним и спрятал песочные часы в бочку, дополнительно приклеив их ко дну. Мало того, что теперь не видно, сколько песка осталось в половинках песочных часов, так ещё и переворачивать их теперь можно только синхронно! Чтобы не выбить Ваню полностью из привычного ритма жизни, он в бочку дополнительно встроил электронику, которая отслеживает момент, когда из одной из частей песок полностью пересыпался во вторую — тогда в бочке пищит специальный динамик. Более того, если в нескольких часах одновременно высыпался песок, то и динамик пищит с продолжительностью, пропорциональной числу этих часов. Тем не менее, даже максимальное возможное время писка пренебрежимо мало по сравнению с любым из t_i.

Пока Ваня спал, в каждых часах весь песок успел просыпаться вниз. Когда Ваня проснулся, он, конечно, очень расстроился, а электронику в бочке воспринял как дополнительную издёвку над своей ретроградской сущностью. Однако одной досадой голод не утолишь, и Ваня решил всё равно пожарить яичницу, несмотря на возникшие препятствия. Поможете ему?

Протокол взаимодействия

Ваша программа будет отправлять команды специальной программе жюри — *интерактору* — и получать от неё ответы в следующем режиме. Почти всё время готовки Ваня будет просто стоять и смотреть на плиту. Однако в моменты *событий* — в стартовый момент, а также в момент, когда какие-то часы пищат — Ваня может совершить некоторое действие.

В такие моменты нужно сначала прочитать информацию о событии. А именно, в начальный момент прочитайте из первой строки целое число n — количество песочных часов ($1 \leqslant n \leqslant 100$). Затем прочитайте из второй строки целые числа $k_1, k_2, \ldots, k_n$, разделённые пробелами — количества переворотов каждых песочных часов для того, чтобы отмерить T секунд ($0 \leqslant k_i \leqslant 100$, $1 \leqslant \sum\limits_{i=1}^{n} k_i \leqslant 100$). Напомним, что само число $T = \sum\limits_{i=1}^{n} k_it_i$, а все числа t_i неизвестны — известно лишь, что $t_1 \leqslant t_2 \leqslant \ldots \leqslant t_n$. Известно, что в начальный момент в каждых часах песок полностью находится внизу.

В остальные моменты — когда часы пищат — прочитайте одну строку. Она имеет вид

«Beee...ep!», где букв «e» вдвое больше, чем количество песочных часов, в которых в этот момент полностью пересыпался песок из верхней половины в нижнюю.

В перечисленных выше двух случаях Ваня может выбрать, что делать дальше. Вы должны вывести одну из трёх команд:

- «Wait» — тогда Ваня будет ждать следующего события;
- «Flip and wait» — тогда Ваня сначала перевернёт бочку (и, следовательно, синхронно перевернёт все песочные часы), а затем будет ждать следующего события;
- «Stop» — тогда Ваня выключит плиту, снимет яичницу со сковородки и начнёт её есть. Выведя эту команду, ваша программа должна завершить работу. Если к этому моменту яичница пробыла на сковородке ровно T секунд, ваша программа получит вердикт OK.

После каждой команды надо делать перевод строки и *сбрасывать буфер вывода*. В таблице приведены примеры на нескольких языках (если в четвёртом столбце есть вариант, то его можно использовать вместо обоих предыдущих).

Язык	Перевод строки	Сброс буфера вывода	И то, и другое

C	`#include <stdio.h>`
	
	`printf("\n");`
		`#include <stdio.h>`
		
		`fflush(stdout);`
			—

Python	`print()`	`import sys`
		
		`sys.stdout.flush()`
			`print(flush=True)`

Java	`System.out.println();`	`System.out.flush();`	—

C++	`#include <iostream>`
	
	`cout << '\n';`
		`#include <iostream>`
		
		`cout.flush();`
			`#include <iostream>`
			
			`cout << endl;`

Delphi/Pascal	`WriteLn;`	`Flush(Output);`	—

D	`import std.stdio;`
	
	`writeln;`
		`import std.stdio;`
		
		`stdout.flush();`
			—

Таблица 1: перевод строки и сброс буфера вывода в популярных языках

Кроме вышеперечисленных, есть ещё три *терминальных* события:

- яичница сгорела, то есть во время очередного ожидания суммарное время пребывания яичницы на сковородке превзошло T секунд, в таком случае интерактор выведет «Burn»;
- Ване вывели суммарно более 30 000 команд типа «Wait» и «Flip and wait», и он устал от готовки, в таком случае интерактор выведет «Tired». Команда «Stop» **не считается** утомительным действием; таким образом, разрешается сделать 30 000 команд типа «Wait» и «Flip and wait», а в качестве 30 001-го действия завершить готовку.
- ваша программа вывела некорректное действие, в таком случае интерактор выведет «Fail».

Получив сообщение о любом из терминальных событий, ваша программа должна завершить работу; она получит вердикт WA (Wrong Answer); впрочем, при выводе некоторых некорректных действий программа может даже не дождаться слова «Fail» и получить, например, IL (Idleness Limit Exceeded). Кроме того, вы получите вердикт WA, если яичница

окажется недожаренной (если она к моменту, как вы вывели команду «`Stop`», пробыла на сковороде меньше T секунд). Обратите внимание, что если ваша программа будет выводить буквы не в том регистре, в каком это указано в списке команд, это не будет считаться ошибкой.

Система оценки

Чтобы решение было принято на проверку, оно должно пройти все примеры. Кроме примеров, в задаче есть 100 тестов — по одному на каждое значение n в пределах от 1 до 100. Каждый тест оценивается независимо и стоит 1 балл.

Примеры

стандартный ввод	***стандартный вывод***
1 1 Burn	 Wait
5 3 0 0 0 0 Beeeeeeeep! Beeeeeeeeeep! Beeeeeeeep!	 Flip and wait Flip and wait Flip and wait Stop
2 0 1 Beep! Beep!	 Flip and wait Wait Stop

Пояснения к примерам

В первом примере Ваня не перевернул бочку и просто стоял у плиты и ждал. Он дождался, что его завтрак сгорел, и вы в такой ситуации получили бы вердикт `WA`. Напомним, что если вы получите `WA` на этом примере, равно как и на любом другом, то ваше решение не будет тестироваться дальше и получит ноль баллов.

Во втором примере $t_1 = t_2 = t_3 = t_4 = 60$ секунд, $t_5 = 120$ секунд, $T = 3t_1 = 180$ секунд. Поэтому через минуту после того, когда Ваня перевернул бочку, одновременно высыпался весь песок в четырёх часах (поэтому в слове «`Beee...ep!`» оказалось $4 \cdot 2 = 8$ букв «`e`»), а в пятых он высыпался наполовину. Затем Ваня перевернул бочку, и через минуту во всех пяти часах одновременно песок вернулся в исходную половинку (поэтому в слове «`Beee...ep!`» оказалось $5 \cdot 2 = 10$ букв «`e`»). Наконец, Ваня снова перевернул бочку, дождался, когда одновременно высыпался весь песок в четырёх часах, и готовка завершилась. Этот пример пройден.

В третьем примере, поскольку $T = t_2$, Ваня проигнорировал сигнал, поступивший от первых песочных часов, и дождался, пока досыплется песок во вторых. Этот пример также пройден.

Задача C. Половина информации

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Дана строка из n двоичных цифр. Замените в ней не менее $\lfloor n/2 \rfloor$ цифр на знаки вопроса, а затем по результату восстановите исходную строку.

Протокол взаимодействия

В этой задаче ваше решение будет запущено на каждом тесте два раза. В конце каждой строки входных данных следует символ перевода строки.

При первом запуске решение заменяет цифры на знаки вопроса. В первой строке записано целое число t — количество тестовых случаев ($1 \leqslant t \leqslant 100\,000$). Каждая из следующих t строк содержит тестовый случай — строку из двоичных цифр. Длина каждой строки — от 2 до 200 000 символов. Гарантируется, что суммарная длина строк не больше 200 000 символов.

В ответ на каждый тестовый случай выведите заданную строку, в которой не менее $\lfloor n/2 \rfloor$ цифр заменены на знаки вопроса, где n — длина строки, а $\lfloor x \rfloor$ — округление вниз числа x.

При втором запуске решение восстанавливает исходные строки. В первой строке записано целое число t — количество тестовых случаев ($1 \leqslant t \leqslant 100\,000$). Каждая из следующих t строк содержит тестовый случай — строку из двоичных цифр и знаков вопроса, ровно ту, которую вывело решение при первом запуске.

В ответ на каждый тестовый случай выведите исходную двоичную строку без знаков вопроса.

Пример

На каждом тесте входные данные при втором запуске зависят от того, что вывело решение при первом запуске.

Далее показаны два запуска какого-то решения на первом тесте.

стандартный ввод	*стандартный вывод*
2 1011 01100	1??? ?1?00
2 1??? ?1?00	1011 01100

Система оценки

Тесты состоят из примера и четырёх групп, каждая из которых даёт 25 баллов.

В первой группе длины строк — от 2 до 10 символов и при этом **нечётные**.

Во второй группе длины строк — от 2 до 10 символов и при этом **чётные**.

В третьей группе длины строк — от 100 до 200 000 символов и при этом **нечётные**.

В четвёртой группе длины строк — от 100 до 200 000 символов и при этом **чётные**.

Чтобы получить баллы за группу, нужно пройти **пример** и все тесты этой группы.

Обратите внимание: во всех тестах n либо не больше 10, либо не меньше 100. Тем не менее, существует решение, правильно работающее для всех строк длиной от 2 до 200 000 символов.

Задача D. Красивые множества

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Назовём множество (возможно, пустое) из целых чисел от 1 до n *красивым*, если оно не содержит двух последовательных чисел. Для того, чтобы потренироваться в работе с разными системами счисления, Миша выписал для каждого красивого множества квадрат произведения чисел из этого множества (произведение чисел в пустом множестве считаем равным единице). Затем он просуммировал все выписанные числа, а результат записал в системе счисления с основанием $998\,244\,353$ (это число простое).

Помогите Мише проверить эти вычисления. Посчитайте количество нулей на конце результата, а также его последнюю ненулевую цифру.

Формат входных данных

На вход дано единственное целое число n ($1 \leqslant n \leqslant 10^{18}$).

Формат выходных данных

Выведите два целых числа — ответ на задачу.

Система оценки

Задача содержит семь подзадач. Баллы за каждую подзадачу будут начислены, если пройдены все тесты этой и всех предыдущих подзадач, а также пример. Стоимости подзадач и ограничения в них перечислены в таблице ниже.

Подзадача	Баллы	Ограничение на n
1	10	$n \leqslant 10$
2	10	$n \leqslant 18$
3	10	$n \leqslant 10^4$
4	10	$n \leqslant 10^6$
5	20	$n \leqslant 10^8$
6	20	$n \leqslant 998\,244\,351$
7	20	$n \leqslant 10^{18}$

Пример

стандартный ввод	*стандартный вывод*
3	0 24

Пояснение к примеру

Для $n = 3$ множества и квадраты произведений в них таковы: $\varnothing \to 1$, $\{1\} \to 1^2 = 1$, $\{2\} \to 2^2 = 4$, $\{3\} \to 3^2 = 9$, $\{1, 3\} \to 1^2 \cdot 3^2 = 9$. Сумма $1 + 1 + 4 + 9 + 9 = 24$.

Задача E. N станков

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	6 секунд
Ограничение по памяти:	512 мебибайт

Яна решила заняться бизнесом. Она придумала следующее: есть $n + 1$ типов деталей, которые мы будем называть деталями нулевого, первого, второго, ..., n-го типа. Все эти детали так или иначе востребованы на рынке. Идея бизнеса в том, что у её знакомого есть завод с n современными станками, i-й из которых способен преобразовывать детали $(i - 1)$-го типа в детали i-го типа. У станков могут быть разные скорости: i-й станок делает v_i преобразований в день (однако можно подать в станок меньше v_i деталей, и изготовит он столько же новых деталей, сколько в него подали старых). Суточная аренда любого станка стоит одну монету, и любой станок можно арендовать в любой момент на любое целое число суток.

Не все станки одинаково инновационные, поэтому не все преобразования деталей приведут к прибыли. Поэтому Яна решила попробовать преобразовывать детали типа a в детали типа $b > a$. Для пробы она купила d деталей a-го типа, и хочет изготовить d деталей b-го типа. Найдите наименьшее число монет, которых хватит на нужное количество аренд станков с $(a+1)$-го по b-й, чтобы выполнить эти преобразования. Поскольку Яна ещё не приняла окончательное решение, необходимо найти ответ для нескольких различных a_i, b_i, d_i.

Формат входных данных

В первой строке находится целое число n — количество станков ($1 \leqslant n \leqslant 300\,000$). Во второй строке находится n целых чисел v_i, разделённых пробелами — скорости станков ($1 \leqslant v_i \leqslant 300\,000$).

В третьей строке находится целое число q — количество запросов ($1 \leqslant q \leqslant 300\,000$).

В следующих q строках находится по три целых числа a_i, b_i, d_i, разделённых пробелами — исходный тип детали, тип детали, которую надо произвести, и количество этих деталей ($0 \leqslant a_i < b_i \leqslant n$, $1 \leqslant d_i \leqslant 1\,000\,000$).

Формат выходных данных

Выведите q строк. В i-й строке должно быть записано одно целое число: наименьшее количество монет, которых хватит на d_i преобразований из детали типа a_i в деталь типа b_i.

Система оценки

В этой задаче ваше решение будет проверяться на нескольких группах тестов. Решение проверяется на тестах группы, если оно прошло все примеры и все тесты предыдущих групп. За группу начисляются баллы, если решение прошло все тесты этой группы.

В первой группе $1 \leqslant n, v_i, q, d_i \leqslant 64$. За эту группу можно получить 12 баллов.

Во второй группе $1 \leqslant n, v_i, q \leqslant 250$ и $1 \leqslant d_i \leqslant 500\,000$. За эту группу можно получить 18 баллов.

В третьей группе $1 \leqslant n, v_i, q \leqslant 5000$ и $1 \leqslant d_i \leqslant 500\,000$. За эту группу можно получить 28 баллов.

В четвёртой группе дополнительных к формату входных данных ограничений нет. За эту группу можно получить оставшиеся 42 балла.

Пример

стандартный ввод	***стандартный вывод***
5 1 5 2 4 3 8 0 5 1 0 5 30 0 1 7 1 2 7 2 3 7 3 4 7 4 5 7 3 5 9	5 69 7 2 4 2 3 6

Задача F. Сколько равных?

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мебибайт

Вам даны n отрезков целых чисел $[\ell_1, r_1]$, $[\ell_2, r_2]$, ..., $[\ell_n, r_n]$ (числа ℓ_i и r_i могут быть отрицательными). Назовём массив целых чисел $x_0, x_1, \ldots, x_n$ *хорошим*, если $x_0 = 0$ и разность $x_i - x_{i-1}$ попадает в отрезок $[\ell_i, r_i]$ для каждого i от 1 до n (иными словами, $\ell_i \leqslant x_i - x_{i-1} \leqslant r_i$). Какое наибольшее количество попарно равных элементов может быть в хорошем массиве? Гарантируется, что $\ell_i < r_i$ для каждого i от 1 до n.

Формат входных данных

В первой строке дано целое число n — количество отрезков ($1 \leqslant n \leqslant 10^6$). В i-й из следующих n строк даны два целых числа ℓ_i и r_i — границы i-го отрезка ($-10^9 \leqslant \ell_i < r_i \leqslant 10^9$).

Формат выходных данных

Выведите одно целое число — наибольшее возможное количество попарно равных элементов в хорошем массиве.

Система оценки

Тесты к этой задаче состоят из четырёх групп. Баллы за каждую из первых трёх групп ставятся только при прохождении примера, а также всех тестов, подходящих под ограничения этой группы. Баллы за четвёртую группу ставятся лишь при прохождении примера и первых трёх групп. При этом каждый тест четвёртой группы оценивается независимо.

Во всех тестах первой группы $1 \leqslant n \leqslant 20$. Также в этой группе для всех i от 1 до n выполнено условие $-100 \leqslant \ell_i < r_i \leqslant 100$. За прохождение всех тестов первой группы можно получить 8 баллов.

В тестах второй группы $1 \leqslant n \leqslant 10^4$. В этой и во всех следующих группах дополнительных ограничений на ℓ_i и r_i не накладывается, то есть $-10^9 \leqslant \ell_i < r_i \leqslant 10^9$. За вторую группу можно получить ещё 23 балла.

В тестах третьей группы $1 \leqslant n \leqslant 2 \cdot 10^5$. За третью группу можно получить ещё 37 баллов.

На тесты четвёртой группы не накладывается никаких дополнительных ограничений, то есть в ней $1 \leqslant n \leqslant 10^6$. В этой группе 16 тестов, которые оцениваются независимо и стоят по 2 балла каждый. При этом баллы за эти тесты начисляются, только если вы полностью прошли первые три группы.

Примеры

стандартный ввод	*стандартный вывод*
1 -7 11	2
1 4 13	1
2 1 2 -100 -1	2
2 4 10 -2 2	2

Пояснения к примерам

В первом примере подходит массив $x_0 = 0$, $x_1 = 0$. В нём два раза встречается число 0.

Во втором примере подходит массив $[0, 7]$. В нём по разу встречаются числа 0 и 7.

В третьем примере подходит массив $[0, 1, 0]$. В нём два раза встречается число 0.

Разбор задачи «Сколько нулей?»

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

Отсортируем числа на входе по возрастанию b_i и просуммируем, поддерживая сумму в таком же формате. Легко видеть, что если в какой-то момент времени b_i стало строго больше, чем количество нулей в конце суммы, то ответ на задачу уже не изменится, и мы можем сразу же его вывести, не суммируя оставшиеся числа. С другой стороны, также легко видеть, что если этого не случилось, то количество ненулевых цифр в сумме не бывает больше пятнадцати (так как всего чисел не более $2 \cdot 10^5$, и каждое из них не превосходит 10^9), а значит, складывать числа можно за константное время.

Разбор задачи «Бочка и песочные часы»

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

Для удобства будем мерить песок в количестве секунд, которое уйдёт на то, чтобы этот песок полностью пересыпался в другую половину часов. Например, фраза «в нижней половине хотя бы t песка» значит, что если часы перевернуть, песок будет сыпаться из верхней половины (которая до переворачивания была нижней) в нижнюю не менее t секунд.

Сначала допустим, что в бочке двое песочных часов, $t_1 \leqslant t_2$. Как отмерить $t_1 + t_2$ секунд? Для этого нужно перевернуть часы, прождать t_2 секунд (то есть дождаться второго писка динамика), ещё раз перевернуть часы, прождать t_1 секунд. Почему этот способ сработает? Потому что, если мы прождали t_2 секунд, то в первых часах гарантированно за это время весь песок высыплется в нижнюю половину.

Аналогично давайте отмерим любую линейную комбинацию с целыми неотрицательными коэффициентами. Пусть надо отмерить $\sum\limits_{i=1}^{n} k_i t_i$ секунд. Решение — сначала k_n раз отмерить t_n секунд, потом k_{n-1} раз отмерить t_{n-1} секунд, и так далее. Как это сделать, и почему это будет работать?

Пусть самый большой k_i, не равный нулю — $k_m \geqslant 1$ (то есть $k_i = 0$ при $m < i \leqslant n$). Будем мерить количество песка. Будем поддерживать следующий инвариант: для каждого i в нижней половине i-х песочных часов должно быть не менее $\min\{t_i, t_m\} = t_{\min\{i,m\}}$ песка. Другими словами, в первых m часах в нижней половине весь песок, а в остальных часах в нижней половине не менее t_m песка. Заметим, что если инвариант выполняется для m, то он выполняется и для всех меньших индексов.

Как этот инвариант поможет нам? Допустим, он выполнен в текущий момент C. Перевернём часы и дождёмся m-го писка динамика (который мог произойти одновременно с несколькими другими — в частности, возможно, мы в итоге услышим более m писков). Докажем, что прошло ровно t_m секунд.

- Докажем, что к этому моменту прошло не меньше t_m секунд. Действительно, поскольку мы услышали m писков, мы услышали писк хотя бы одних часов с номером $i \geqslant m$. Но из инварианта в момент C следует, что после переворачивания в их верхней половине было не менее t_m песка, а, значит, на то, чтобы весь песок высыпался, ушло не менее t_m секунд.

- Докажем, что к этому моменту прошло не больше t_m секунд. Действительно, если мы прождём ровно t_m секунд, то в первых m часах непременно высыплется вниз весь песок (так как в каждых из них не более t_m песка), они все пропищат, и дальше точно будет бессмысленно ждать.

Нетрудно понять, что теперь инвариант для числа m выполнен: в первых m часах весь песок пересыпался, а в остальных пересыпалось вниз хотя бы t_m песка. Значит, для нового m' (которое равно m, если $k_m > 1$, и меньше m, если $k_m = 1$) инвариант также верен. Таким образом, в момент $C + t_m$ мы снова можем действовать и снова выполнен инвариант. Продолжая так делать, мы отмерим все нужные промежутки t_i в порядке убывания.

Разбор задачи «Половина информации»

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

Эта задача про кодирование и декодирование отличается от других тем, что возможно не любое сопоставление исходной и закодированной строки: символы, которые не меняются на знаки вопроса, должны остаться теми же.

Решение 1: Давайте заменять на вопросики ту цифру, которая встречается чаще.

Например, в строке «01001111» три нуля и пять единиц, поэтому закодируем её как «0?00????». При декодировании вопросики — это та цифра, которой не осталось: если мы видим «0?00????», то понимаем, что, раз остались нули, вопросики стоят на месте единиц.

Единственная проблема в таком решении — то, что строка из всех нулей и строка из всех единиц заменяются на одну и ту же строку. Действительно, увидев код «?????», мы не можем понять, какой была исходная строка: «00000» или «11111».

Для нечётных n можно, например, сделать так: «00000» заменять на «?????», а «11111» на «111??»: $\lceil n/2 \rceil$ единиц и $\lfloor n/2 \rfloor$ вопросиков. Действительно, при кодировании всех других строк вопросиков будет строго больше половины, значит, такая строка не получится.

Для чётных n будем решать так же, но сначала определим, что делать при равном количестве нулей и единиц: например, будем заменять на вопросики единицы, а нули — оставлять. Тогда из «1100» получится «??00», а не «11??». Теперь наше правило с превращением «1111» в «11??» сработает и для чётного n, так как «11??» не могло получиться заменой нулей.

Решение 2: Давайте разделим задачу на маленькие части.

Сначала придумаем какое-нибудь решение при $n = 2$: например, 00 ↔ 0?, 01 ↔ ??, 10 ↔ 1?, 11 ↔ ?1.

Теперь для чётных n просто разобьём строку на пары символов, и в каждой паре решим подзадачу с $n = 2$. Например, «01 00 11 11» превратится в «?? 0? ?1 ?1». Получится, что вопросиков не меньше половины, так как их не меньше половины в каждой паре.

Для нечётных n сделаем то же самое, а последний символ оставим без изменений. Поскольку требуемое количество вопросиков округляется вниз, вопросиков в парах будет достаточно.

Разбор задачи «Красивые множества»

Автор задачи:	Никита Гаевой
Подготовка тестов и решений:	Никита Гаевой
Автор разбора:	Никита Гаевой

Заметим, что сумма чисел в условии равна $(n + 1)!$. Этот факт легко доказывается по индукции, а также это можно было получить, вычислив несколько значений суммы при маленьких значениях n и угадав формулу. Наивного решения, вычисляющего факториал по модулю, было достаточно, чтобы пройти все группы тестов, кроме последних двух. Для того, чтобы пройти предпоследнюю группу, достаточно было выполнить предподсчёт факториалов с шагом в 10^7. Для того, чтобы пройти и последнюю группу, нужно было дополнительно применить теорему Вильсона, из которой следует, что $998\,244\,352!$ сравним с -1 по модулю $998\,244\,353$. Теорема Вильсона позволяет легко учесть все множители, не кратные $998\,244\,353$.

К счастью, для того, чтобы учесть все остальные, достаточно решить такую же задачу вычисления количества конечных нулей и последней ненулевой цифры числа $\left\lfloor \frac{n+1}{998\,244\,353} \right\rfloor !$, что можно сделать рекурсивно.

Разбор задачи «n станков»

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

Чтобы решить первые две группы, достаточно было исполнять каждый запрос за линейное время. Чтобы сделать d_i преобразований из деталей a_i в детали b_i, нужно для каждого j от $a_i + 1$ до b_i совершить не менее d_i преобразований из детали $j-1$ в деталь j. Хотелось бы сказать, что для этого надо j-й станок арендовать на $\frac{d_i}{v_j}$ дней, но в общем случае d_i не делится на v_j, поэтому на самом деле надо арендовать станок на $\left\lceil \frac{d_i}{v_j} \right\rceil$ дней. Итак, надо просто посчитать

$$\sum_{j=a_i+1}^{b_i} \left\lceil \frac{d_i}{v_j} \right\rceil.$$

Чтобы получить полный балл, предстояло это ускорить. Рассмотрим каждое слагаемое $\left\lceil \frac{d_i}{v_j} \right\rceil$ как функцию от d_i. Как она себя ведёт при замене $d_i - 1$ на d_i? Нетрудно понять, что следующим образом: если d_i сравнимо с единицей по модулю v_j, то возрастает на единицу, а если не сравнимо, то не изменяется. При $d_i = 0$ функция нулевая. Таким образом, можно рассмотреть строку, в которой на позициях, сравнимых с единицей по модулю v_j, стоит единица, а на остальных стоит ноль, и требуемое число — количество единичек на позициях с нулевой по d_i-ю. Назовём эту строку *аддитивным массивом*.

Воспользуемся корневой декомпозицией. Обозначим максимальное возможное d_i буквой D. Зафиксируем некоторое n_0 и рассмотрим два вида v_j — не превосходящие n_0 и превосходящие. Станки первого вида назовём *частыми*, второго — *редкими*. Сначала разберёмся с частыми станками. Для каждого $i \leqslant n_0$ построим структуру данных, позволяющую за $\mathcal{O}(1)$ на отрезке массива $\{v_j\}_{i \in \{1,\ldots,n\}}$ находить количество элементов, равных i (префиксные суммы). Теперь в каждом запросе те станки, у которых $v_j \leqslant n_0$, мы можем обработать за $\mathcal{O}(n_0)$: переберём i, найдём от $a+1$ до b число s_i станков с $v_j = i$ и добавим к ответу $\left\lceil \frac{d}{i} \right\rceil$. Суммарно эти операции займут $\mathcal{O}(qn_0)$ времени.

Теперь разберёмся с редкими станками. Они хороши тем, что в у каждого из них в аддитивном массиве не более $\left\lceil \frac{D}{n_0} \right\rceil$ единичек. Будем постепенно увеличивать b и для каждого его значения строить дерево Фенвика, поддерживающее массив, равный поэлементной сумме аддитивных массивов всех редких станков с первого по b-й, и позволяющее находить префиксную сумму этого массива. Чтобы перейти от b к $b+1$, надо либо ничего не сделать (если станок частый), либо сделать $\left\lceil \frac{D}{n_0} \right\rceil$ запросов к дереву Фенвика, в каждом из которых добавляется единица в некоторый элемент. Такой проход займёт $\mathcal{O}(\frac{nD \log D}{n_0})$ времени (логарифм на операции с деревом Фенвика).

Как тогда отвечать на запросы? Воспользуемся тем, что все запросы известны заранее (то есть поступают в offline). Тогда для ответа на запрос (a_i, b_i, d_i) мы должны в момент, когда мы храним дерево Фенвика в состоянии a_i, вычесть из ответа ans_i d_i-ю префиксную сумму, а когда мы храним дерево Фенвика в состоянии b_i, добавить d_i-ю префиксную сумму. Тогда мы в итоге $2q$ раз обратимся к дереву Фенвика и потратим на это $\mathcal{O}(q \log D)$ времени.

Таким образом, получилась асимптотика $\mathcal{O}\left(qn_0 + q \log D + \frac{nD \log D}{n_0}\right)$. Оптимально подобрать такое n_0, чтобы первое и третье слагаемое были равны с точностью до константы. Для этого возьмём $n_0 = \Theta\left(\sqrt{\frac{nD \log D}{q}}\right)$ и получим асимптотику $\mathcal{O}\left(q \log D + \sqrt{nqD \log D}\right)$.

Разбор задачи «Сколько равных?»

Автор задачи:	Владислав Макаров
Подготовка тестов и решений:	Владислав Макаров, Никита Гаевой
Автор разбора:	Владислав Макаров

В разборе будет описано решение жюри на полный балл (возможно, есть какие-то существенно другие решения). Для того, чтобы прийти к этому решению, нужно было сделать несколько последовательных наблюдений, каждое из которых приводит к решению на большее количество баллов.

1 Решение на 31 балл

Каждый хороший массив $x = [x_0, x_1, \ldots, x_n]$ задаётся своим массивом разностей $d = [d_1, d_2, \ldots, d_n]$, где $d_i = x_i - x_{i-1} \in [\ell_i, r_i]$ для каждого i от 1 до n. Более того, для $0 \leqslant i < j \leqslant n$, условие $x_i = x_j$ эквивалентно тому, что $0 = x_j - x_i = (x_j - x_{j-1}) + (x_{j-1} - x_{j-2}) + \ldots + (x_{i+1} - x_i) = d_j + d_{j-1} + \ldots + d_{i+1}$.

Заметим, что это условие зависит только от d-шек с номерами от $i+1$ до j. Пусть есть хороший массив, в котором элементы на позициях $0 \leqslant i_1 < i_2 < \ldots < i_k \leqslant n$ равны (то есть $x_{i_1} = x_{i_2} = \ldots = x_{i_k}$). То, равны ли x_{i_1} и x_{i_2} или нет, зависит только от того, как мы выбрали d-шки с номерами от $i_1 + 1$ до i_2. Аналогично, равны ли x_{i_2} и x_{i_3} или нет, зависит только от d-шек с номерами от $i_2 + 1$ до i_3. И так далее. Важно, что все эти отрезки d-шек попарно не пересекаются.

Получается, что условия $x_{i_1} = x_{i_2}$, $x_{i_2} = x_{i_3}$, $\ldots$, $x_{i_{k-1}} = x_{i_k}$ в каком-то смысле «независимы»: если можно выполнить каждое из них по отдельности, то можно выполнить и все вместе.

Давайте построим такой граф: вершинами будут числа от 0 до n, а ориентированное ребро из вершины i в вершину j будет тогда и только тогда, когда существует хороший массив, в котором $x_i = x_j$. Несложно видеть, что нас интересует длина самого длинного пути в таком графе.

Когда может выполниться условие $x_i = x_j$? В точности когда $d_{i+1} + d_{i+2} + \ldots + d_j$ может равняться 0. С другой стороны, $d_{i+1} + d_{i+2} + \ldots + d_j$ — произвольное целое число из отрезка $[\ell_{i+1} + \ell_{i+2} + \ldots + \ell_j, r_{i+1} + r_{i+2} + \ldots + r_j]$.

Следовательно, нужно проверить, содержит ли этот отрезок число 0. Если зафиксировать j и рассматривать все $i < j$ в порядке убывания, то границы этого отрезка поддерживать легко. Пусть dp_i — длина наибольшего пути, заканчивающегося в вершине i. Она пересчитывается вот так: dp_j — это максимум $\mathrm{dp}_i + 1$ по всем таким $i < j$, что в графе есть ребро $i \to j$; если же таких i вообще нет, то 1. Получили решение за $O(n^2)$ времени и $O(n)$ памяти.

2 Решение на 68 баллов

Давайте немного упростим предыдущее решение. А именно, пусть $L_i := \ell_1 + \ell_2 + \ldots + \ell_i$, $R_i := r_1 + r_2 + \ldots + r_i$ для всех $0 \leqslant i \leqslant n$. Тогда наличие ребра $i \to j$ в графе эквивалентно тому, что отрезок $[\ell_{i+1} + \ell_{i+2} + \ldots + \ell_j, r_{i+1} + r_{i+2} + \ldots + r_j] = [L_j - L_i, R_j - R_i]$ содержит число 0. Теперь нам уже не важен порядок, в котором мы перебираем i: всё равно условие проверяется за $O(1)$.

Но давайте пойдём ещё чуть дальше. Что означает, что отрезок $[L_j - L_i, R_j - R_i]$ содержит 0? Это означает, что $L_j - L_i \leqslant 0 \leqslant R_j - R_i$, то есть $L_j \leqslant L_i$ и $R_i \leqslant R_j$. Иными словами, отрезок $[L_i, R_i]$ содержится в отрезке $[L_j, R_j]$!

Таким образом, dp_j — это максимум $\mathrm{dp}_i + 1$ по таким i, что отрезок $[L_i, R_i]$ содержится в $[L_j, R_j]$. Отсюда получается решение за $O(n \log^2 n)$: идём по j в порядке возрастания и храним двумерную структуру данных на максимум, делаем запрос максимума на прямоугольнике

$[L_j, +\infty) \times (-\infty, R_j]$, с помощью него понимаем значение dp_j, после чего обновляем структуру данных в точке (L_j, R_j) значением dp_j.

Поскольку все отрезки $[L_j, R_j]$ известны заранее, то можно заранее сжать все координаты (это не самая простая, но стандартная техника; её объяснение выходит за рамки данного разбора). В зависимости от конкретных используемых структур и аккуратности реализации такое решение должно получать от 68 до 100 баллов.

3 Решение на 100 баллов

Предыдущее решение использует двумерные структуры данных и работает за $O(n \log^2 n)$. Авторское решение не использует никаких структур данных, работает за $O(n \log n)$ и реализуется очень просто. Чтобы к нему прийти, нужно сделать ещё одно наблюдение.

На самом деле мы хотим найти самую длинную последовательность отрезков $[L_{i_k}, R_{i_k}]$, в которой каждый отрезок вложен в следующий, а индексы i_k возрастают. Оказывается, что от второго условия можно избавиться. Действительно, длины отрезков $[L_i, R_i]$ возрастают с увеличением i, при этом даже строго (из-за условия $\ell_i < r_i$). Поэтому, если один отрезок вложен в другой, то у него автоматически меньший индекс.

Давайте поэтому отсортируем отрезки по возрастанию R (а при равенстве R — по убыванию L). Тогда, если один отрезок вложен в другой, то бо́льший отрезок идёт в этом порядке позже. На набор вложенных отрезков должно выполняться два условия: нестрогое убывание L-ок и нестрогое возрастание R-ок. Второе условие выполняется автоматически (мы так отсортировали). Поэтому, в таком порядке сортировки, наборы вложенных отрезков соответствуют в точности нестрого убывающим последовательностям L-ок.

Итак, мы свели нашу к задачу к известной задаче о наибольшей возрастающей подпоследовательности (с точностью до знака и строгости/нестрогости сравнений). Она решается либо с помощью сжатия координат и одномерного дерева отрезков, либо с помощью двоичного поиска без каких-либо структур данных вообще. Второе решение особенно приятно в реализации и заходит в ограничения по времени с почти шестикратным запасом.

Интуитивно можно сказать, что произошло следующее: у нас была «трёхмерная» задача про отрезки $[L_i, R_i]$ (измерениями были левые границы L, правые границы R и сами индексы i). В таких задачах обычно можно «избавиться» от одного из измерений с помощью сканирующей прямой. В предыдущем решении мы так избавлялись от i. Однако, оказалось, что это третье измерение не независимо от двух других, а очень строго с ними связано. Поэтому от него можно избавиться «бесплатно», а ещё от одного измерения (в данном случае, R-ок) — с помощью сканирующей прямой. Здесь использовалась специфика задачи (длины отрезков возрастают с увеличением i); для произвольных отрезков $[L_i, R_i]$, скорее всего, нет никакого «одномерного» решения.