Задача A. Три сына

Для решения этой задачи требуется считать целые числа A, B и C, а затем вывести $A + B + C$.

Однако в этой задаче есть две вспомогательные группы тестов, позволяющие решать ее поэтапно.

Для прохождения 1-й группы тестов можно не считывать вводимые числа вовсе. Достаточно написать вывод числа 8 — ответа на задачу при $A = 8, B = 0, C = 0$.

Во 2-й группе тестов $B = 0$ и $C = 0$, поэтому достаточно считать только первое число и вывести ответ, равный A.

Критерии оценивания:

№	Баллы	Ограничения			Необх. группы
		A	B	C	
1	12	$A = 8$	$B = 0$	$C = 0$	—
2	25	$A \leqslant 100$	$B = 0$	$C = 0$	1
3	63	$A \leqslant 100$	$B \leqslant 100$	$C \leqslant 100$	1, 2

Баллы выставляются автоматически проверяющей системой.

Задача B. Голландские вафли

Если $t = a + b$ и $k = p$, то Ваня может сделать не больше 1 вафли, а 1 вафлю можно сделать всегда. Поэтому чтобы решить 1-ю группу, достаточно вывести единицу.

Во 2-й группе также нельзя испечь больше 1 вафли, так как $k = p$. Поэтому задача состоит в том, чтобы проверить хватит ли нам времени, чтобы испечь 1 вафлю, то есть проверить, что $t \geqslant a + b$.

3-я группа решается аналогично 2-й, только здесь надо проверить, хватит ли теста, чтобы испечь 1 вафлю, то есть что $p \geqslant k$.

В 4 группе теста хватит максимум на 2 вафли. Давайте посмотрим, хватит ли Ване времени на приготовление одной ($t \geqslant a + b$) или двух ($t \geqslant 2 \cdot (a + b)$) вафель. Разобрав несколько случаев, получаем ответ.

Чтобы сдать эту задачу на полный балл, давайте посчитаем, на сколько вафлей хватает времени и на сколько вафлей хватает теста: $c_1 = t//(a + b)$, $c_2 = k//p$, где // обозначает целочисленное деление. Тогда ответом будет $\min(c_1, c_2)$. Итоговая сложность решения — $\mathcal{O}(1)$.

Критерии оценивания:

№	Баллы	Ограничения		Необх. группы
		t	k	
1	9	$t = a + b$	$k = p$	—
2	14	$t \leqslant 2 \cdot 10^9$	$k = p$	1
3	14	$t = a + b$	$k \leqslant 2 \cdot 10^9$	1
4	21	$t \leqslant 2 \cdot 10^9$	$k \leqslant 2p$	1, 2
5	42	$t \leqslant 2 \cdot 10^9$	$k \leqslant 2 \cdot 10^9$	1-4

Баллы выставляются автоматически проверяющей системой.

Задача C. Урок математики

В 1-й группе тестов достаточно было заметить, что если $e = 0$, то и a всегда должно быть равно 0. Тогда осталось посчитать, сколько существует b таких, что $\frac{0}{b} = \frac{0}{f}$. Понятно, что любое b в промежутке от 1 до M подходит, а значит, надо было просто вывести M.

Во 2-й группе можно было перебрать a и b, а затем проверить, что $\frac{a}{b} = \frac{e}{f}$, то есть $a \cdot f = e \cdot b$. Сложность такого решения будет $\mathcal{O}(M^2)$.

Чтобы решить 3 группу, давайте представим a в виде $k \cdot e$, где k — положительное число. Тогда $\frac{k \cdot e}{b} = \frac{e}{f}$, $k \cdot e \cdot f = e \cdot b$, а следовательно, $b = k \cdot f$.

Давайте докажем, что k в этой группе тестов должно быть целым числом. Конечно, k — рациональное (т. е. представимое в виде дроби), так как иначе число $a = e \cdot k$ было бы иррациональным,

а нам нужно, чтобы a было рациональным, и более того — целым. Тогда докажем от противного: пусть k такое *нецелое* число, что $a = k \cdot e$ и $b = k \cdot f$. Так как k — рациональное, то представим k в виде несократимой дроби: $k = \frac{p}{q}$. Значит, $\frac{p \cdot e}{q}$ и $\frac{p \cdot f}{q}$ равны a и b соответственно. Но тогда e и f делятся на q, потому что p и q взаимно просты, и $q \neq 1$, так как мы взяли такое k, что оно нецелое. Следовательно, дробь $\frac{e}{f}$ можно сократить на q, но $\frac{e}{f}$ — несократимая дробь по условию группы. Противоречие. Получается, что k — целое положительное число.

Тогда если мы будем перебирать k, то a и b будут определяться однозначно. Будем увеличивать k, пока $e \cdot k$ и $f \cdot k$ меньше или равно M, и прибавлять единицу к ответу на каждом шаге перебора.

В 4-й группе может быть дана сократимая дробь. Тогда давайте просто сократим её и запустим решение для 3 группы. Для этого найдем набольший общий делитель e и f, а потом поделим e и f на него. Наибольший общий делитель можно найти с помощью алгоритма Евклида или при помощи встроенных в стандартную библиотеку языка функций. С его оптимальной реализацией временная сложность решения будет $\mathcal{O}(M + \log \min(e, f))$. Это решение проходит первые 4 группы.

Чтобы пройти последние 2 группы, заметим, что представленный алгоритм перебора проходит ровно $M / \max(e, f)$ шагов, если $\frac{e}{f}$ — несократимая дробь. Поэтому можно сократить введенную дробь $\frac{e}{f}$ и посчитать ответ при помощи формулы. Следовательно, итоговая сложность алгоритма — $\mathcal{O}(\log \min(e, f))$ из-за алгоритма Евклида.

Критерии оценивания:

№	Баллы	Ограничения			Необх. группы
		e	M	дополнительно	
1	13	$e = 0$	$M \leqslant 1000$	—	—
2	18	$e \leqslant 10^5$	$M \leqslant 1000$	—	1
3	26	$e \leqslant 10^5$	$M \leqslant 2 \cdot 10^5$	$\frac{e}{f}$ — несократимая дробь	1
4	14	$e \leqslant 10^5$	$M \leqslant 2 \cdot 10^5$	—	1–3
5	13	$e \leqslant 10^5$	$M \leqslant 2 \cdot 10^9$	$\frac{e}{f}$ — несократимая дробь	1, 3
6	16	$e \leqslant 10^5$	$M \leqslant 2 \cdot 10^9$	—	1–5

Баллы выставляются автоматически проверяющей системой.

Задача D. Вася и цветочки

В 1-й группе при заданных ограничениях Вася может только проходить каждое поле друг за другом раз в день. Значит, в i-й день он будет получать единицу недовольства, если посетит поле, которое ему нравится — такой уж странный Вася. Так, за каждую единицу в строке s Вася получит одну единицу недовольства, поэтому, чтобы посчитать ответ, посчитаем при помощи цикла или встроенной в стандартную библиотеку языка функции число символов «1» в строке s.

Во 2-й группе выгоднее всего в первый же день пройти весь маршрут. Тогда недовольство Васи будет равно 1, если в строке было нечетное число единиц, и 0 иначе. Данное решение отличается от решения первой группы тестов только выводом.

Поймем, почему это оптимальная стратегия. Если Вася получит нулевое недовольство, то меньшее недовольство точно получить нельзя. Но если эта стратегия даст нам единицу недовольства, почему нет маршрута, который не принесет ни единицы недовольства?

Если этот алгоритм дает единицу в качестве ответа, то всего в строке s нечетное число единиц. Чтобы получить в результате ноль единиц недовольства, Васе нужно организовать маршрут так, чтобы каждый день проходить четное число полей, которые ему нравятся. Но придерживаясь такой стратегии он суммарно пройдет тоже четное число полей, а должен пройти нечетное! Поэтому предложенная стратегия оптимальна.

В 3-й группе можно применить следующий алгоритм: если два следующих поля нравятся Васе, то в этот день посетим 2 поля, иначе только 1.

Обоснуем эту стратегию. Чтобы минимизировать суммарное недовольство, нужно как можно меньше дней проходить нечетное число полей, которые нравятся Васе. В этой группе тестов это значит, что если Вася встретит поле, которое ему нравится, то единственный случай, когда он не получит единицу недовольства — если за ним снова поле, которое ему нравится, и он пройдет за

один день оба поля. Отметим также, что в остальных случаях проходить два поля за один день бессмысленно: ответ это улучшить не может, и Васе безразлично, за сколько дней он прошел весь маршрут. Зато это заметно упрощает реализацию.

Чтобы проэмулировать эту стратегию, понадобится переменная, в которой будем считать общее недовольство Васи, и цикл, на i-м шаге которого будем обрабатывать ситуацию: «Вася прошел все поля до i-го, то есть следующее поле, которое он должен посетить — i-е». Посмотрим на текущее поле и на следующее, и если хотя бы один символ $s[i]$ или $s[i+1]$ равен `0`, то следует посетить только i-е поле, то есть перейти к следующему шагу цикла (например, при помощи оператора `continue`), перед этим, если нужно, добавив единицу недовольства к общему недовольству. Если же $s[i] = s[i+1] = $ `1`, то нужно пропустить следующий шаг цикла, и сразу продолжить с шага $i + 2$.

Чтобы такое реализовать, заведем вспомогательную переменную `skip`, которая должна показывать, сколько шагов цикла следует пропустить. Будем начинать цикл с того, что проверим, не нужно ли текущий шаг пропустить: если `skip` > 0, уменьшим `skip` на единицу и начнем следующий шаг (`continue`) — этот шаг цикла пропущен. Если же `skip` $= 0$, то шаг пропускать не нужно. Тогда при встрече двух единиц подряд, чтобы пропустить следующий шаг цикла, достаточно присвоить `skip` значение 1.

Отметим также, что проверку *следующего* поля нужно делать аккуратно: если текущий шаг цикла последний, то i — последнее поле, за ним не стоит следующего. Если этот случай не учесть и всегда обращаться к $s[i+1]$, то программа может аварийно завершиться.

Немного улучшив решение, можно придумать оптимальную стратегию и решить задачу на полный балл. Если Вася способен ходить на k полей за один день, то единственный случай, когда это однозначно лучше, чем ходить по одному полю за день — это ситуация, когда следующее поле (которое Вася еще не посетил) ему нравится, и нужно посетить четное число полей, чтобы не увеличивать Васино недовольство. Тогда нет смысла проходить четное число полей, большее 2, ведь один день, за который Вася проходит 4, 6, 8, и т д. полей можно разбить на 2, 3, 4, и т д. дня, в которые он проходит по два поля. Это упрощает реализацию.

Тогда стратегия Васи в следующем: он посещает по одному полю в день до тех пор, пока не встретит поле, которое ему нравится. Когда же Вася встречает поле, которое ему нравится, проверим, сможет ли он посетить следующее за ним поле, которое ему тоже нравится. Если да, посетим за текущий день все поля между этими двумя, а если нет — посетим только одно поле.

Возможность пропускать любое число шагов уже реализовано при помощи переменной `skip`. А для поиска следующей `1` в строке s можно запустить вложенный цикл от позиции $i+1$ до $i+k$. Это правильный подход, но неэффективный: сложность такого решения составит $\mathcal{O}(nk)$.

Рабочий способ ускорить поиск следующей единицы — заранее подсчитать в массиве h позицию следующей единицы для каждого индекса i. Чтобы это сделать, запустим цикл от конца строки к началу и будем поддерживать во вспомогательной переменной `last` последнюю позицию, где встретилась единица. Тогда на каждом шаге цикла i сначала присвоим $h[i]$ значение `last`, а потом, если $s[i] = $ `1`, запишем в `last` значение i.

Теперь исходный алгоритм состоит только в том, чтобы на i-м шаге, если $s[i] = $ `1`, посмотреть на значение $h[i]$ и убедиться, что расстояние от позиции i до позиции $h[i]$ не больше, чем k — тогда Вася может за текущий день посетить два поля, которые ему нравятся, что ему и следует сделать. Сложность такого решения: $\mathcal{O}(n)$.

Критерии оценивания:

№	Баллы	Ограничения		Необх. группы
		n	k	
1	7	$n \leqslant 1000$	$k = 1$	—
2	9	$n \leqslant 5 \cdot 10^5$	$k = n$	—
3	28	$n \leqslant 5 \cdot 10^5$	$k \leqslant 2$	1
4	34	$n \leqslant 1000$	$k \leqslant 1000$	1
5	22	$n \leqslant 5 \cdot 10^5$	$k \leqslant 5 \cdot 10^5$	1–4

Баллы выставляются автоматически проверяющей системой.

Задача E. Сделай равными

1 группа: Пусть $cnt(x)$ — количество чисел в массиве, равных x. Данную функцию можно посчитать при помощи линейного прохода по массиву или при помощи встроенных в стандартную библиотеку языка функций. В данной группе достаточно выбрать лучший вариант из двух: либо превратить все двойки в единицы, либо все единицы в двойки. Поэтому ответ равен $\min(n - cnt(1), n - cnt(2)) = \min(cnt(2), cnt(1))$. Сложность решения: $\mathcal{O}(n)$.

2 группа: Давайте перебирать, какому числу y из промежутка $1 \dots k$ будут равны введенные числа в итоге. Когда число y выбрано, посчитаем необходимое число операций, чтобы сделать все числа равными y, и ответом будет минимум из таких чисел по всем y.

Если y зафиксировано, то чтобы все числа сделать равными y, надо изменить все числа a_i на y, кроме чисел, которые изначально равны y, то есть всего $n - cnt(y)$ чисел, где $cnt(y)$ можно посчитать, как в решении для прошлой группы тестов. Сложность алгоритма: $\mathcal{O}(nk)$

3 группа: Нам выгодно оставить какие-то числа без изменений. Действительно, чтобы сделать все числа равными y, надо сделать $n - cnt(y)$ операций, поэтому нам выгодно, чтобы $cnt(y)$ было как можно больше, в частности, никогда не было равно 0.

Улучшим решение предыдущей группы просто: перебирать y будем не из всех чисел, а только из $a_1, a_2, \dots, a_n$. Считать для числа y необходимое число операций $n - cnt(y)$ будем аналогично. Сложность алгоритма: $\mathcal{O}(n^2)$

4 группа: Улучшим решение 3 группы. Долго работает линейный подсчет по массиву (то есть $cnt(y)$). Для решения этой проблемы заведем дополнительный массив h, где $h[i]$ должно быть равно $cnt(i)$. Чтобы этот массив завести за линейное время, можно запустить цикл-проход по массиву: для каждого i добавить единицу к $h[a_i]$. После этого $cnt(x)$ равно $h[x]$, и вместо подсчета $cnt(y)$ (циклом или встроенной функцией) будем обращаться к предподсчитанному ответу: $h[y]$. Сложность решения: $\mathcal{O}(n)$

Критерии оценивания:

№	Баллы	Ограничения		Необх. группы
		n	k	
1	14	$n \leqslant 1000$	$k = 2$	—
2	27	$n \leqslant 1000$	$k \leqslant 1000$	1
3	26	$n \leqslant 1000$	$k \leqslant 10^6$	1, 2
4	33	$n \leqslant 10^5$	$k \leqslant 10^6$	1–3

Баллы выставляются автоматически проверяющей системой.