Задача A. Разгрузочная машина

Для решения этой задачи требуется вывести значение $T_a + N \cdot T_b$ (время на запуск машины, плюс число ящиков, умноженное на время, необходимое для выгрузки одного ящика).

Эту задачу также можно решать поэтапно, начиная с первой группы тестов. Чтобы решить эту задачу на частичный балл, можно воспользоваться дополнительным ограничением $N = 1$, и тогда достаточно выводить значение $T_a + T_b$.

Критерии оценивания:

№	Баллы	Ограничения			Необх. группы
		N	T_a	T_b	
1	56	$N = 1$	$T_a \leqslant 100$	$T_b \leqslant 100$	—
2	44	$N \leqslant 100$	$T_a \leqslant 100$	$T_b \leqslant 100$	1

Баллы выставляются автоматически проверяющей системой.

Задача B. Соки

В первой группе тестов на вечеринку не придет ни одной пары, поэтому нет дополнительных ограничений, как следует наливать соки гостям. Поэтому, с одной стороны, есть $A + B + C$ стаканов сока, а с другой, есть N гостей, каждый из которых хочет по стакану. Значит, максимальное число гостей, которые могут уйти довольными, равно $\min(A + B + C, N)$.

Во второй группе на вечеринку могут прийти пары. Отметим, что вместо пары можно сделать довольными двух одиноких гостей, но вместо двух одиноких не всегда можно сделать довольными пару. Поэтому сначала сделаем довольными все пришедшие пары.

С одной стороны, есть A стаканов сока, что позволит сделать довольными не больше $A//2$ пар, где $//$ — операция деления нацело. С другой стороны, пар ровно M. Поэтому можно сделать довольными не больше $2 \cdot \min(M, A//2)$ гостей, если считать только пары. После этого давайте вычтем полученный результат из суммарного числа стаканов, и используя оставшееся число стаканов, нужно сделать довольными как можно больше одиноких гостей. Так задача сведена к предыдущей группе.

Чтобы решить последнюю группу тестов, надо обобщить формулу для пар: $2 \cdot \min(M, A//2 + B//2 + C//2)$. К тому же, отметим, что одиночкам неважно, какой именно сок был потрачен на пары, общее число стаканов при любом распределении будет $A + B + C -$ «результат для пар», и результат, соответственно: $\min(A + B + C -$ «результат для пар», $N)$. Так, тоже в два шага, можно решить задачу на полный балл.

Критерии оценивания:

№	Баллы	Ограничения				Необх. группы
		A	B	C	M	
1	18	$A \leqslant 5 \cdot 10^8$	$B \leqslant 5 \cdot 10^8$	$C \leqslant 5 \cdot 10^8$	$M = 0$	—
2	33	$A \leqslant 5 \cdot 10^8$	$B = 0$	$C = 0$	$M \leqslant 5 \cdot 10^8$	—
3	49	$A \leqslant 5 \cdot 10^8$	$B \leqslant 5 \cdot 10^8$	$C \leqslant 5 \cdot 10^8$	$M \leqslant 5 \cdot 10^8$	1–2

Баллы выставляются автоматически проверяющей системой.

Задача C. Ручной счетчик

1 группа. В данной группе достаточно было вывести n. Действительно, Артем может после каждой записи сбрасывать счетчик, и тогда каждую минуту он будет записывать на листочек число 1. Сложность решения: $\mathcal{O}(1)$.

2 группа. Артем не может использовать кнопку сброса, поэтому он будет последовательно выписывать натуральные числа от 1 до n. Давайте тогда посчитаем сумму чисел $1 + 2 + \ldots + n$ при помощи цикла и выведем ее.

3 группа. Чтобы пройти ограничения по времени при решении этой группы, вспомним формулу суммы первых m чисел арифметической прогрессии с шагом 1 и началом 1, (собственно, ее Артем и пишет без учета возможных сбросов счетчика) — $S_m = \frac{m \cdot (m+1)}{2}$.

Тогда переберем при помощи цикла, после какого числа Артем сбросит счетчик. Для каждого такого варианта посчитаем по формулы две суммы первых элементов арифметической прогрессии: до сброса счетчика и после. Так, сумма этих двух значений — возможный результат. Ответом будет минимум из всех таких результатов. Сложность: $\mathcal{O}(n)$

4 группа. Эта группа предполагает полный перебор всех вариантов, перед какими днями Артем должен сбросить счетчик. Это можно сделать при помощи метода перебора битовых масок или при встроенных средств языка (например, в Python можно воспользоваться библиотекой `itertools`). Но универсальным решением является перебор при помощи рекурсии.

Определим рекурсивную функцию $f(n, l, k)$ как минимальный ответ, если Артем еще хочет записать n чисел, следующее число, которое покажет счетчик, равно l, а также он может еще k раз нажать на кнопку сброса. Тогда ответ должен вернуть вызов функции $f(n, 1, k)$.

Эту функцию $f(n, l, k)$ для произвольных n, l, k можно определить следующим образом. Если $n = 0$, то Артем больше не запишет на листочек ни одного числа, поэтому сумма чисел на листочке будет равна 0, немедленно вернем это значение. Иначе Артем точно напишет число l, а затем либо сбросит счетчик, если это возможно, либо продолжит без сброса. В первом случае он напишет на листочке числа, сумма которых не больше $l + f(n - 1, 1, k - 1)$, а во втором $l + f(n - 1, l + 1, k)$. Функция должна вернуть минимум из этих двух чисел.

Сложность решения зависит от реализации, но ее точно можно оценить сверху как $\mathcal{O}(2^n \cdot n)$.

5-6 группы. Нажатиями на кнопку сброса Артем делит все время, пока он пишет числа на листочек, на $k + 1$ последовательных блоков, в каждом из них содержится несколько элементов арифметической прогрессии с началом в 1 и шагом 1. Пусть $l_1, l_2, \ldots, l_{k+1}$ — длины получившихся блоков.

Заметим, что в оптимальном ответе $\max(l_i) - \min(l_i) \leqslant 1$. Докажем этот факт от противного: пусть в оптимальном ответе $\max(l_i) - \min(l_i) > 1$. Тогда без ограничения общности, l_1 — самый маленький блок, l_2 — самый длинный. Давайте уменьшим второй блок на 1, а первый увеличим на 1. Тогда ответ изменится на $l_2 - l_1 - 1$. А это значение больше 0, так как $l_2 - l_1 > 1$. Противоречие: вариант оказался не оптимальным.

Тогда достаточно разделить все n минут на $k+1$ блок, где $\max(l_i) - \min(l_i) \leqslant 1$. Такое разбиение существует, и при том только одно, с точностью до перестановки. Его можно сгенерировать при помощи цикла, если исходить из следующих соображений: длины блоков должны отличаться не больше, чем на единицу, а значит, отличаться от $L = \lfloor \frac{n}{k+1} \rfloor$ не более, чем на единицу. Затем можно сгенерировать $k+1$ блоков длины L и добавить оставшиеся $(n \bmod (k+1))$ минут в какие-то $(n \bmod (k+1))$ блоков. Напомним, что $\lfloor x \rfloor$ обозначает целую часть числа x, а $a \bmod b$ — остаток от деления числа a на b.

На полный балл вместо генерации тем или иным образом, сразу выведем формулу: $(k+1-n \bmod (k+1)) \cdot \frac{L \cdot (L+1)}{2} + (n \bmod (k+1)) \cdot \frac{(L+1) \cdot (L+2)}{2}$. Сложность решения: $\mathcal{O}(1)$.

Критерии оценивания:

№	Баллы	Ограничения		Необх. группы
		n	k	
1	5	$n \leqslant 10^9$	$k = n - 1$	—
2	9	$n \leqslant 10^5$	$k = 0$	—
3	19	$n \leqslant 10^5$	$k \leqslant 1$	2
4	26	$n \leqslant 10$	$k \leqslant 10$	—
5	25	$n \leqslant 10^5$	$k \leqslant 10^5$	2–4
6	16	$n \leqslant 10^9$	$k \leqslant 10^9$	1–5

Баллы выставляются автоматически проверяющей системой.

Задача D. Табличные данные

В 1 группе есть всего два возможных варианта входных данных: $n = 3$, $m = 3$, $d = 1$ и $n = 3$, $m = 3$, $d = 2$. Эти два варианта можно разобрать на листочке, получить ответ и написать следующее решение с использованием только условного оператора и ввода-вывода данных: если вводится $d = 1$, то вывести ответ на первый вариант, а иначе — на второй.

Во 2-й группе итоговая таблица имеет определенный вид: у клеток в углах три клетки на расстоянии, не большем 1, у остальных клеток на границе их 4, а у оставшихся 5. Исключение составляют только случаи, когда $n = 1$ или $m = 1$: тогда в крайних клетках стоят 2, а в остальных 3.

Такую таблицу несложно сгенерировать, причем даже с учетом крайних случаев. Для этого заведем двумерный массив $a[i][j]$ (часто для этого используется массив массивов) размера $n \times m$, и запишем во все ячейки число 5. Теперь запустим цикл, перебирающий i от 1 до n, при помощи него вычтем единицу из всех клеток $a[i][1]$ и $a[i][m]$. И аналогично переберем i от 1 до m, вычтем единицу из всех клеток $a[1][i]$ и $a[n][i]$. Так, во всех клетках, которые находятся на границе, будет записана 4, а в углах — 3, так как углы будут перебраны в двух циклах. Можно убедиться, что алгоритм работает и на крайних случаях ($n = 1$ или $m = 1$): крайние клетки встретятся в циклах трижды, поэтому в них в итоге будет записана 2, а остальные — дважды, поэтому в них будет записана 3. Сложность такого решения: $\mathcal{O}(NM)$.

Подобную идею можно реализовать и для решения 3-й группы тестов, однако алгоритм будет гораздо сложнее.

В 4-й группе ограничения достаточно небольшие. Переберем числа i, j — номер клетки в таблице, и для каждой клетки переберем все остальные клетки в пару к этой, то есть числа r, s. Так, при помощи вложенных циклов посчитаем явно для клетки (i, j), сколько есть клеток (r, s) таких, что $|i - r| + |j - s| \leqslant d$. Сложность такого решения: $\mathcal{O}(N^2M^2)$.

Чтобы решить 3-ю и 5-ю группы, улучшим решение 4-й группы. Чтобы считать ответ для клетки (i, j), переберем только клетки квадрата $2d \times 2d$ с центром в клетке (i, j). Действительно, все клетки на необходимом расстоянии могут находиться только внутри этого квадрата. Чтобы это сделать, достаточно ограничить перебираемые значения циклов по переменным r и s числами $i - d$, $i + d$ и $j - d$, $j + d$, соответственно.

Решение можно еще ускорить, если дополнительно ограничить перебор s в зависимости от значения r. Так можно добиться того, что алгоритм для каждой клетки (i, j) посетит только клетки, которые точно находятся на расстоянии не больше d. Но в любом случае, сложность решения: $\mathcal{O}(NMd^2)$.

Чтобы решить задачу на полный балл, создадим шаблон-прямоугольник достаточно большого размера (например, 4000×4000). В каждой клетке этого прямоугольника, которая находится на расстоянии меньше или равном d от центральной клетки $(2000, 2000)$, поставим 1, а в остальных 0. Идея заключается в следующем: для каждой клетки (i, j) будем накладывать на шаблон-прямоугольник поле $n \times m$ так, чтобы центральная клетка шаблона находилась там же, что и клетка (i, j) в поле. Так, если посчитать число единиц на подпрямоугольнике шаблона, получим ответ для этой клетки. То есть реализация этого «наложения» — это поиск координат подпрямоугольника, на котором нужно найти сумму, а для ее быстрого подсчета можно посчитать на прямоугольнике-шаблоне двумерные префиксные суммы — вспомогательный массив, при помощи которого сумму на любом подпрямоугольнике можно посчитать при помощи нескольких операций сложения/вычитания. Сложность решения: $\mathcal{O}(NM)$.

Критерии оценивания:

№	Баллы	Ограничения			Необх. группы
		n	m	d	
1	5	$n = 3$	$m = 3$	$d \leqslant 2$	—
2	14	$n \leqslant 150$	$m \leqslant 150$	$d = 1$	—
3	11	$n \leqslant 150$	$m \leqslant 150$	$d \leqslant 2$	1, 2
4	16	$n \leqslant 10$	$m \leqslant 10$	$d \leqslant 10$	1
5	23	$n \leqslant 150$	$m \leqslant 150$	$d \leqslant 10$	1–4
6	31	$n \leqslant 1000$	$m \leqslant 1000$	$d \leqslant 2000$	1–5

Баллы выставляются автоматически проверяющей системой.

Задача E. Биржа

В первой группе тестов стоимость акций неубывает, а у Димы ограничения на баланс акций достаточно большие, поэтому оптимально первую половину рабочего дня покупать акции, а вторую — продавать их. Прибыль Димы тогда можно посчитать при помощи цикла.

Во второй группе можно написать полный перебор вариантов при помощи рекурсии. Пусть $f(i, b, p)$ — функция, которая должна перебирать все возможные стратегии Димы, начиная с i-го дня при имеющемся балансе акций b и прибыли p. Если $i > n$, то рабочий день закончен, и если $b = 0$, то p — один из возможных ответов. Максимальный ответ можно хранить в некоторой глобальной переменной, и тогда ее следует обновить. Если же $i \leqslant n$, выберем в i-й день, покупает Дима акцию, продает акцию или ничего не делает — это три рекурсивных запуска функции: $f(i+1, b+1, p-a_i)$, $f(i+1, b-1, p-a_i)$ и $f(i+1, b, p)$. Осталось только учесть ограничения l и r, то есть не вызывать те функции, в которых баланс выходит из установленных ограничений. Сложность такого решения: $\mathcal{O}(3^n)$.

Остальные группы тестов предполагают решение при помощи метода динамического программирования. Напомним, что основная идея этого метода — придумать, как ответ на сложную задачу («посчитать максимальную прибыль Димы») можно посчитать как некоторую функцию (обычно сумма или максимум) от ответов на задачу в такой же формулировке, но сколько-нибудь упрощенную. Естественная идея при таком подходе: считать задачу «на префиксах», то есть попробовать научиться решать задачу «посчитать максимальную прибыль Димы к i-й минуте» как функция от таких же задач при меньших i.

В третьей группе тестов баланс может быть только 1 или 0. Поэтому можно применить метод динамического программирования, где за $dp[i]$ (при реализации dp — это обычно массив) обозначим максимальную прибыль Димы к концу минуты i, если его баланс равен нулю. Вырожденный случай: $dp[0] = 0$ (0-й момент времени — это начало рабочего дня).

Чтобы посчитать $dp[i]$, поймем, как у Димы может оказаться нулевой баланс к концу i-й минуты. Это может быть в двух случаях: баланс, равный нулю, у Димы остался с прошлой минуты и в минуту i он продал акцию, когда-то до этого купленную. Так как требуется найти максимум, $dp[i]$ можно посчитать следующим образом:

$$dp[i] = \max(dp[i-1], dp[0] - a_1 + a_i, dp[1] - a_2 + a_i, \ldots, dp[i-2] - a_{i-1} + a_i)$$

Когда будет посчитано $dp[n]$, выведем это число. Сложность решения: $\mathcal{O}(n^2)$.

Для решения задачи на полный балл предлагается добавить еще один параметр — текущий баланс. Так, заведем двумерный массив $dp[i][j]$, в котором поддерживать максимальную возможную прибыль к концу i-й минуты при балансе j.

Инициализация массива: $dp[0][0] = 0$, $dp[0][j \neq 0] = -\infty$ (нам нужно, чтобы никогда не было выгодно обновляться от $dp[0][j \neq 0]$, потому что единственное возможное начало в этой задаче — 0-й момент времени с нулевым балансом). Теперь можно пересчитывать значения, разобрав все возможные варианты действий Димы. После минуты i он может иметь баланс j, если он с прошлой минуты продаст акцию, купит акцию или ничего не сделает. Тогда значение $dp[i][j]$ можно посчитать следующим образом:

$$dp[i][j] = \max(dp[i-1][j], dp[i-1][j-1] - a_i, dp[i-1][j+1] + a_i)$$

Такую «динамику» нужно посчитать по всем i и j, поэтому сложность решения: $\mathcal{O}(n(r-l))$. Отметим также, что здесь можно выйти в отрицательные индексы, так как баланс может быть отрицательным. Решение, которое не учитывает это, проходит только четвертую группу тестов. Учесть отрицательный баланс можно, если сдвинуть индексы, то есть хранить максимальную возможную прибыль к концу i-й минуты при балансе j в $dp[i][j + 2000]$.

Критерии оценивания:

№	Баллы	Ограничения			Необх. группы
		N	l, r	Дополнительно	
1	11	$n \leqslant 1500$	$\|l\| = 0, \|r\| = n$	$a_1 \leqslant a_2 \leqslant \ldots \leqslant a_n$	—
2	21	$n \leqslant 8$	$\|l\| \leqslant n, \|r\| \leqslant n$		—
3	28	$n \leqslant 1500$	$\|l\| = 0, \|r\| = 1$		—
4	34	$n \leqslant 1500$	$\|l\| = 0, \|r\| = n$		1
5	6	$n \leqslant 1500$	$\|l\| \leqslant n, \|r\| \leqslant n$		1–4

Баллы выставляются автоматически проверяющей системой.