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. Помогите Коле

1 группа. Понятно, что к одному числу бесполезно сначала прибавлять, а затем убавлять единицу, — две различные операции над одним и тем же числом можно не делать, и будет получено то же число в ответе за меньшее число операций.

Будем перебирать циклом, какому значению x должны быть равны все числа в массиве после операций. Можно перебирать x в диапазоне от $\min_i\{a_i\}$ до $\max_i\{a_i\}$ или от 1 до 750 (ограничения группы).

Чтобы посчитать, сколько операций понадобится сделать, чтобы все элементы стали равны x, добавим вложенный цикл, перебирающий элементы $a_1, a_2, \ldots, a_n$. Число операций, которое необходимо сделать для преобразования числа a_i в x, равно $|x-a_i|$ (модуль во многих языках программирования реализован в стандартной библиотеке функцией `abs()`). Сумма этих чисел — ответ при заданном x. Требуется вывести минимум среди всех таких вариантов. Сложность решения: $\mathcal{O}(nk)$

2 группа. Для решения данной подзадачи нужно воспользоваться следующим фактом: среди стратегий Коли рассмотрим те, в которых какие-то числа исходного массива $a_1, a_2, \ldots, a_n$ останутся без изменений. Среди таких стратегий точно есть оптимальная стратегия, то есть та, при которой будет совершено минимальное число операций.

Докажем этот факт конструктивно. Если минимальный ответ достижим при некотором алгоритме действий, где каждое число изменилось, покажем, как из него получить оптимальную стратегию, где какое-то число осталось тем же. Пусть в результате какого-то оптимального по числу операций алгоритма действий все числа будут равны x. Определим тогда массив *изменений*: $k_1, k_2, \ldots, k_n$, где $k_i = x - a_i$. Понятно, что если a_i изменится на k_i, то для изменения a_i потребуется совершить $|k_i|$ операций.

Пусть cnt_- — количество значений $k_i < 0$, cnt_+ — количество значений $k_i > 0$. Положим, $cnt_- \geqslant cnt_+$. Тогда можно улучшить ответ на значение $cnt_- - cnt_+$, если отменим 1 операцию уменьшения на единицу к числам, чье изменение отрицательно, но применим 1 операцию увеличения на единицу к числам, чье изменение положительно. Так получится набор действий, делающий все числа равными $x+1$ за количество операций не большее, чем оптимальный, а значит этот вариант должен быть оптимальным. При этом все k_i увеличатся на 1. Так можно продолжать «улучшать» стратегию, пока какое-то из отрицательных k_i не увеличится до 0 — тогда мы получили оптимальную стратегию, в которой какое-то из чисел не меняется. Аналогично, если $cnt_- \leqslant cnt_+$, можно уменьшить все k_i на 1, что улучшит ответ на $cnt_+ - cnt_-$, и делать так до тех пор, пока какое-то положительное k_i не уменьшится до 0.

Хоть этот факт и непростой, из него вытекает довольно простая оптимизация решения предыдущей группы: давайте перебирать x только по числам $a_1, a_2, \ldots, a_n$. Действительно, выше доказано, что среди таких вариантов x есть тот, для которого существует стратегия для Димы, делающая все числа равными за минимальное число шагов. В остальном решение оставим прежним. Сложность: $\mathcal{O}(n^2)$

Решение 4-й группы отличается от решения 3-й группы только использованием факта выше: перебираются все x или только $x = a_i$. Поэтому разберем сразу решение 4-й группы.

4 группа. Улучшим предыдущее решение. Будем перебирать a_i, но уже в порядке возрастания. Чтобы было проще, отсортируем массив $a_1, a_2, \ldots, a_n$. Чтобы уложиться в ограничения, нужно применить достаточно быструю сортировку, например, «быструю сортировку» или сортировку слиянием. Также большинство алгоритмов сортировки из стандартных библиотек языков программирования подходят.

Для перебираемого варианта x зафиксируем i как индекс числа, которое зафиксировано как неизменяемое ($x = a_i$). Тогда из-за того, что массив отсортирован, $a_1 \leqslant x$, $a_2 \leqslant x$, …, $a_{i-1} \leqslant x$ и $a_i \geqslant x$, $a_{i+1} \geqslant x$, …, $a_n \geqslant x$.

Тогда число действий для заданного x равно $(x - a_1) + (x - a_2) + \ldots + (x - a_{i-1}) + (a_i - x) + (a_{i+1} - x) + \ldots + (a_n - x) = x \cdot (i-1) - sum_1 + sum_2 - x \cdot (n - i + 1)$, где $(i-1)$ и $(n-i+1)$ — это количество чисел в массиве до i-го и начиная с i-го соответственно, а sum_1 и sum_2 — это суммы первых $i-1$ чисел и последних $n-i+1$ чисел соответственно.

Для каждого i посчитать значение $x \cdot (i-1) - sum_1 + sum_2 - x \cdot (n-i+1)$ можно за $\mathcal{O}(1)$, если воспользоваться тем, что i перебирается циклом последовательно от 1 до n. Научимся поддерживать значения sum_1 и sum_2 в некоторых переменных актуальными для каждого шага цикла.

В первый шаг достаточно посчитать $sum_2 = a_1 + a_2 + \ldots + a_n$, а $sum_1 = 0$. После этого, когда закончена обработка i-го шага, вычтем из sum_2 число a_i, а к sum_1 это число прибавим. Так, в этих переменных всегда будут необходимые суммы. Сложность решения: $\mathcal{O}(n + \texttt{сложность сортировки})$

Критерии оценивания:

№	Баллы	Ограничения		Необх. группы
		n	a_i	
1	26	$n \leqslant 750$	$\max(a_1, a_2, \ldots, a_n) \leqslant 750$	—
2	25	$n \leqslant 750$	$\max(a_1, a_2, \ldots, a_n) \leqslant 3 \cdot 10^5$	1
3	33	$n \leqslant 10^5$	$\max(a_1, a_2, \ldots, a_n) \leqslant 3 \cdot 10^5$	1, 2
4	16	$n \leqslant 10^5$	$\max(a_1, a_2, \ldots, a_n) \leqslant 10^9$	1– 3

Баллы выставляются автоматически проверяющей системой.

Задача D. Лыжный лагерь

1 группа. В этой группе ребята будут передвигаться последовательно от s к станции $s+1$ по единственной трассе, затем от $s+1$ к $s+2$, и т. д. до станции n. Поэтому данные u_i и v_i хранить бессмысленно, достаточно только сохранить массив W, где $W[i] = w_i$.

Их игра тогда выглядит следующим образом: сначала Петя получает $W[s]$ очков, потом Вася $W[s+1]$ очков, и так далее. Для решения задачи далее достаточно только посчитать знакопеременную сумму на массиве $W[s], W[s+1], \ldots, W[n]$, она будет равна равную разнице между очками Пети и Васи. И далее остается только при помощи условного оператора посмотреть на знак получившейся суммы: если сумма равна нулю, то у мальчиков ничья, если положительна, то победил Петя с соответствующей разницей, а если отрицательна, то Вася с преимуществом, равным числу, противоположному этой сумме. Сложность решения: $\mathcal{O}(n)$.

2 группа. Дополнительное ограничение «$u_i \neq u_j$» значит, что из каждой станции выходит не более одной трассы. Поэтому эта группа похожа на предыдущую: ребята просто скатываются по единственному возможному маршруту, но здесь он не обязательно идет по последовательным вершинам. Тогда разумно свести задачу к предыдущей, то есть получить массив W, который состоит из длин трасс, которые ребята посетят во время спуска от станции s.

Для того, чтобы составить массив W, нужно научиться быстро определять, на какую станцию можно добраться из текущей. Для этого подойдет стандартный способ, называющийся «списком смежности»: создадим массив массивов g длины n, все элементы которого — пустые списки. Теперь для каждой трассы (u_i, v_i, w_i) добавим в конец u_i-го массива эту трассу в виде пары (v_i, w_i) (u_i можно восстановить из индекса массива, в котором эта пара находится, поэтому хранить это число не обязательно). Получившийся массив массивов g следует трактовать следующим образом: находясь на станции s можно переместиться по трассам, хранящимся в $g[s]$, каждая из которых описывается номером станции, куда она ведет, и длиной. Отметим, что в этой группе тестов у всех $g[i]$ размер равен 1.

Начнем формировать изначально пустой W, приписывая в его конец новые числа. Для этого будем поддерживать станцию st, изначально равную st, из которой ребята в очередной раз должны переместиться дальше. Тогда следующая трасса — это $(v_i, w_i) = g[st][0]$, если $g[st]$ не пуст. Добавим w_i в массив W, а st сменим на v_i — далее спуск продолжится с этой станции. Если же $g[st]$ на каком-то шаге пуст, то на этом спуск ребят заканчивается.

Так, задача сводится к предыдущей группе, но вообще говоря, массив W можно опустить, и считать знакопеременную сумму «на лету», по ходу продвижения ребят по трассам. Сложность решения: $\mathcal{O}(n)$.

В следующих группах нельзя однозначно определить, как выстроен маршрут, поэтому нужно перебирать различные варианты. Чтобы пройти 3 группу, можно взять реализацию 2-й группы с подсчетом суммы по ходу продвижения, и поместить сам алгоритм продвижения в рекурсивную функцию $f(st, sum, sgn)$, где st — текущая вершина, sum — текущая сумма, а sgn — то, с каким знаком нужно взять следующую длину в сумму. Пусть эта функция возвращает итоговую сумму, то есть $f(0, 0, 1)$ вернет число, преобразующееся в ответ на задачу, как в решении первой группы.

Раз действия те же, что и во второй группе, то эта функция вызывает сама себя от следующей вершины по ходу движения. Адаптируем ее под новую группу тестов: если $g[st]$ содержит больше одного элемента, то для каждого из них она запускает себя же от разных вершин и сохраняет локально результаты вызова функций. Теперь когда ребята приезжают на некоторую станцию, набор этих значений — это выбор, куда можно поехать, и какие при этом результаты получить. Выбор

определяется однозначно: в вершине st Петя ($sgn = 1$) поедет туда, где сумма будет наибольшей, а Вася ($sgn = -1$) — где наименьшей. Этот результат функция и должна вернуть.

Но где здесь используется факт, что на каждую станцию можно попасть единственным способом? Он нужен для оценки сложности решения. Благодаря этому ограничению, все вызовы функции f будут с различными значениями st, и поэтому сложность программы: $\mathcal{O}(n)$.

К сожалению, если этого ограничения нет, решение работает слишком долго. Обычно рекурсии оптимизируют при помощи меморизации — техники, где функция запоминает значение, которое она вернула при некоторых аргументах, например, в словаре, и при повторном вызове функции оно проверяет наличие некогда подсчитанного результата по этим аргументам в словаре, она возвращает то значение. Так, можно гарантировать, что функция будет вызвана рекурсивно только один раз от каждого набора аргументов. И эту технику следует применить, но аргумент функции sum может принимать слишком много значений в этой задаче, поэтому следует модифицировать функцию, чтобы она вызывалась от меньшего числа аргументов, например, только от st.

Для этого изменим то, что функция возвращает. f использовала то, что путь ребят начинается на некоторой станции s, и считала конец маршрута, зная начало. Но можно подсчет начала маршрута (до момента, когда ребята придут в st) перенести. Тогда попробуем написать функцию $f(st)$, которая считает наибольшую возможную знакопеременную сумму (ответ), если ребята начнут весь свой маршрут в вершине st.

Тогда меняется способ выбора станции из возможных вариантов (локального массива внутри функции f). Всегда делается выбор для Пети, но функция вернет результат так, как будто бы первым ходил Петя, а в действительности на следующем шагу будет ходить Вася. Тогда в действительности итоговая знакопеременная сумма, если пойти по трассе (v_i, w_i) — это $w_i - f(v_i)$, где минус связан со сменой хода и, как следствие, с чередованием знаков при подсчете суммы длин. Но идея остается той же: по всем таким значением игрок в вершине st, ходящий первым, старается максимизировать итоговый результат, поэтому f должна вернуть максимум по этим значениям. Сложность решения: $\mathcal{O}(n + m)$.

Этот алгоритм — вариация на стандартный алгоритм для решения подобных задач, который называется «обход в глубину».

Также в этой задаче есть 4 группа тестов, в которой $w_i = 1$. Ее решение почти такое же, как решение 5-й группы, но в ней проще определить, что должна возвращать функция f и как это значение посчитать, если опираться на термины из теории игр. Заметим, что если все $w_i = 1$, то возможно всего два исхода: выиграет Петя с преимуществом 1 или будет ничья. Разумно назвать станции, старт в которых приведет к победе Пети, выигрышными, а остальные — проигрышными. Тогда $f(st)$ может возвращать значение истина/ложь, является ли станция st выигрышной. Определение выигрышной позиции в таком случае совпадает со стандартным для игровых задач: если с текущей станции можно переместиться на проигрышную, то она выигрышная, а иначе — проигрышная.

Критерии оценивания:

№	Баллы	Ограничения			Необх. группы
		m	w_i	дополнительно	
1	11	$m = n - 1$	$w_i \leqslant 10^9$	$u_i = i$, $v_i = i + 1$	—
2	23	$m \leqslant 10^5$	$w_i \leqslant 10^9$	$u_i \neq u_j$, для всех $i \neq j$	1
3	24	$m \leqslant 10^5$	$w_i \leqslant 10^9$	$v_i \neq v_j$, для всех $i \neq j$	1
4	20	$m \leqslant 10^5$	$w_i = 1$		—
5	22	$m \leqslant 10^5$	$w_i \leqslant 10^9$		1–4

Баллы выставляются автоматически проверяющей системой.

Задача E. Хорошие раскраски 3

Чтобы сдать 1-ю группу, давайте просто рассмотрим все квадраты $d \times d$. В них может быть максимум 4 элемента (если $d = 2$). Давайте посчитаем их сумму и возьмем из них максимум при помощи формулы и стандартных функций языка программирования. Теперь среди всех квадратов

суммой, большей s, возьмем квадрат с наименьшим максимумом, а среди них ближайший к клетке $(1, 1)$.

Чтобы решить 2-ю группу, надо просто проверить, существует ли квадрат с суммой большей или равной s, потому что если $s > 0$, то в этом квадрате гарантированно есть единица — максимально возможное число в этой группе, а значит, он один из самых интересных. Только если $s = 0$, надо сначала проверить, есть ли квадрат с суммой больше 0, так как он будет интереснее квадратов с нулевой суммой из-за единицы в нем.

Чтобы проверить есть ли квадрат с суммой, не меньшей s, давайте при помощи вложенных циклов перебирать его угол, а потом считать сумму на нем. Сложность этого решения: $\mathcal{O}(nmd^2)$.

Для решения 3-й группы давайте оптимизируем решение 2-й группы, использовав двумерные префиксные суммы, чтобы быстро искать сумму на подквадрате прямоугольника. Сложность оптимизированного решения: $\mathcal{O}(nm)$.

В 4-й группе давайте посчитаем в каждом квадрате сумму и максимум. Это можно сделать перебором угла квадрата, а потом перебором всех клеток этого квадрата. Теперь давайте рассмотрим квадраты с суммой больше или равной s и возьмем среди них квадрат с минимальным максимумом, а среди них ближайший к $(1, 1)$. Сложность этого решения — $\mathcal{O}(nmd^2)$.

Чтобы решить 5-ю группу, давайте перебирать максимум в самом интересном фрагменте. Пусть перебираемое значение равно k. Тогда давайте все значения большие k заменим на $-\infty$ — таким образом, при переборе всех квадратов $d \times d$ квадраты с максимумами, большими k, точно не наберут сумму s. Теперь на каждом шаге перебора останется только проверить, есть ли в полученной табличке квадрат с суммой, большей или равной s. Это можно сделать с помощью двумерных префиксных сумм. Полученная сложность: $\mathcal{O}(nm \max(a_{i,j}))$.

Теперь давайте воспользуемся бинарным поиском по ответу. Будем искать при помощи бинарного поиска k, а затем, как в решении на 5-ю группу, менять табличку и проверять, есть ли квадрат с суммой, большей или равной s. Если он существует, то двигаем правую границу поиска, иначе левую. Бинарный поиск будет работать, так как если существует квадрат с суммой, большей или равной s, и при этом максимумом меньше или равном k, то будет существовать квадрат с суммой, большей или равной s, а максимумом не большем $k + 1$.

Критерии оценивания:

№	Баллы	Ограничения			Необх. группы
		n, m	$\max(a_{i,j})$	d	
1	11	$n, m \leqslant 50$	$\max(a_{i,j}) \leqslant 10^9$	$d \leqslant 2$	—
2	12	$n, m \leqslant 50$	$\max(a_{i,j}) \leqslant 1$	$d \leqslant 50$	—
3	18	$n, m \leqslant 500$	$\max(a_{i,j}) \leqslant 1$	$d \leqslant 500$	2
4	7	$n, m \leqslant 50$	$\max(a_{i,j}) \leqslant 10^9$	$d \leqslant 50$	1, 2
5	26	$n, m \leqslant 500$	$\max(a_{i,j}) \leqslant 10$	$d \leqslant 500$	2, 3
6	26	$n, m \leqslant 500$	$\max(a_{i,j}) \leqslant 10^9$	$d \leqslant 500$	1 – 5

Баллы выставляются автоматически проверяющей системой.