Задача A. Сумма дробей с округлением

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

Дано целое число n. Найдите значение суммы

$$\left\lfloor \frac{n-1}{1^2} \right\rfloor + \left\lfloor \frac{n-2}{2^2} \right\rfloor + \left\lfloor \frac{n-3}{3^2} \right\rfloor + \ldots + \left\lfloor \frac{2}{(n-2)^2} \right\rfloor + \left\lfloor \frac{1}{(n-1)^2} \right\rfloor.$$

Здесь $\lfloor x \rfloor$ — это округление x вниз: наибольшее целое число, не превосходящее x.

Формат входных данных

В первой строке записано целое число n ($2 \leqslant n \leqslant 10^{18}$).

Формат выходных данных

В первой строке выведите одно целое число — значение суммы.

Примеры

стандартный ввод	*стандартный вывод*
3	2
10	11

Пояснения к примерам

В первом примере $\left\lfloor \frac{2}{1^2} \right\rfloor + \left\lfloor \frac{1}{2^2} \right\rfloor = 2 + 0 = 2$.

Во втором примере

$$\left\lfloor \frac{9}{1^2} \right\rfloor + \left\lfloor \frac{8}{2^2} \right\rfloor + \left\lfloor \frac{7}{3^2} \right\rfloor + \left\lfloor \frac{6}{4^2} \right\rfloor + \left\lfloor \frac{5}{5^2} \right\rfloor + \left\lfloor \frac{4}{6^2} \right\rfloor + \left\lfloor \frac{3}{7^2} \right\rfloor + \left\lfloor \frac{2}{8^2} \right\rfloor + \left\lfloor \frac{1}{9^2} \right\rfloor = 9 + 2 + 0 + \ldots + 0 = 11.$$

Система оценки

В этой задаче два примера и 50 основных тестов. Каждый основной тест оценивается в **2** балла. Как и в других задачах, решение должно правильно работать на примерах — иначе проверка на основных тестах не производится.

Задача B. Половинки

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

Даша принесла своей младшей сестре Еве игрушку: детскую книжку «Половинки». В этой книжке две половины: верхняя и нижняя. Страницы в половинах можно листать независимо. Всего в книжке и сверху, и снизу по n разворотов. Развороты в каждой половине пронумерованы целыми числами от 1 до n в порядке, в котором обычно листают книгу.

Сверху на каждом развороте нарисована банка с краской. Все n цветов краски разные, и для удобства пронумерованы целыми числами от 1 до n в том же порядке, что и развороты.

Снизу на каждом развороте нарисован какой-то предмет. Каждый из n предметов имеет один из n цветов, и каждый цвет встречается ровно один раз.

Исходно книжка открыта на первом развороте сверху и первом развороте снизу. За одну секунду Ева может перевернуть одну страницу: либо сверху, либо снизу перейти либо к предыдущему развороту, либо к следующему (конечно, если он есть). Цель игры с книжкой — для каждого цвета k показать одновременно и разворот с краской цвета k сверху, и разворот с предметом цвета k снизу. Цвета можно показывать в любом порядке.

Даша задумалась: как действовать, чтобы достичь цели за минимально возможное время? Помогите ей придумать кратчайшую последовательность действий.

Формат входных данных

В первой строке задано целое число n — количество разворотов в книжке ($2 \leqslant n \leqslant 20$). Во второй строке заданы n целых чисел $a_1, a_2, a_3, \ldots, a_n$ — на каком развороте снизу нарисован предмет первого, второго, третьего, …, n-го цвета ($1 \leqslant a_i \leqslant n$, все числа a_i различны).

Формат выходных данных

Выведите строку, задающую кратчайшую последовательность действий, при которых для каждого из n цветов будет момент, когда сверху открыта банка с краской этого цвета, а снизу — предмет этого цвета. Если кратчайших ответов несколько, выведите любой из них.

Каждое действие записывается одной буквой:

- «`L`» — в верхней половине перевернуть страницу назад,
- «`l`» — в нижней половине перевернуть страницу назад,
- «`R`» — в верхней половине перевернуть страницу вперёд,
- «`r`» — в нижней половине перевернуть страницу вперёд.

Пример

стандартный ввод	*стандартный вывод*
5 1 3 5 2 4	RrrRRlRrrLLr

Пояснение к примеру

Для примера запишем в каждую секунду, начиная с нулевой, пару: на каком по счёту развороте открыты верхняя и нижняя половинки. Жирным шрифтом выделены те пары, которые обязательно нужно получить хотя бы один раз.

$$(\mathbf{1}, \mathbf{1})$$
$$\xrightarrow{\texttt{R}} (2,1) \xrightarrow{\texttt{r}} (2,2) \xrightarrow{\texttt{r}} (\mathbf{2}, \mathbf{3})$$
$$\xrightarrow{\texttt{R}} (3,3) \xrightarrow{\texttt{R}} (4,3) \xrightarrow{\texttt{l}} (\mathbf{4}, \mathbf{2})$$
$$\xrightarrow{\texttt{R}} (5,2) \xrightarrow{\texttt{r}} (5,3) \xrightarrow{\texttt{r}} (\mathbf{5}, \mathbf{4})$$
$$\xrightarrow{\texttt{L}} (4,4) \xrightarrow{\texttt{L}} (3,4) \xrightarrow{\texttt{r}} (\mathbf{3}, \mathbf{5})$$

Система оценки

В этой задаче один пример и 50 основных тестов. Каждый основной тест оценивается в **2** балла. Как и в других задачах, решение должно правильно работать на примере — иначе проверка на основных тестах не производится.

Задача C. Многомерные ферзи

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

Макс, завоевав рейтинг Эло 932 на популярном шахматном сайте, понял, что классические шахматы себя изжили, и принялся думать над их аналогами. Ему в голову пришёл вопрос: а почему доска для шахмат плоская? Почему бы не поиграть в какую-то более многомерную игру?

Размышляя, он придумал, что в новой версии шахмат доска будет иметь форму d-мерного клетчатого параллелепипеда размером $n_1 \times n_2 \times \ldots \times n_d$, где d натуральных чисел $n_1, \ldots, n_d$ Максу ещё предстоит выбрать. Он решил сделать их такими, чтобы силы различных фигур были как можно более сбалансированными. А какая самая сильная фигура? Конечно, ферзь!

Будем считать, что ферзь на клетке $(q_1, q_2, \ldots, q_d)$ бьёт клетку $(s_1, s_2, \ldots, s_d)$, если существует такое натуральное число r, что вдоль каждой оси $i \in \{1, 2, \ldots, d\}$ выполняется одна из двух возможностей:

- либо вдоль этой оси координаты совпадают, то есть $q_i = s_i$;
- либо вдоль этой оси координаты отличаются ровно на r, то есть $|q_i - s_i| = r$.

При этом вдоль хотя бы одной оси должна выполняться не первая возможность, а вторая, другими словами, клетки $(q_1, q_2, \ldots, q_d)$ и $(s_1, s_2, \ldots, s_d)$ должны быть различны. Это нужно для того, чтобы по правилам ферзь не бил клетку, в которой он и так стоит.

Для $d = 2$ выше описаны стандартные правила перемещения шахматного ферзя. Однако при больших d ферзь, по мнению Макса, становится чересчур мощным: количество клеток, которые может побить ферзь, растёт экспоненциально от d.

Помогите Максу понять, хороши ли будут те или иные размеры доски. Он вам скажет d и все n_i, а также скажет координаты клетки, на которую он поставил ферзя, а вы ответьте, сколько клеток бьёт этот ферзь. Поскольку это число может оказаться слишком большим, выведите его по модулю $998\,244\,353$.

Формат входных данных

В первой строке находится одно целое число d — размерность доски ($1 \leqslant d \leqslant 100\,000$).

Во второй строке находится d целых чисел $n_1, n_2, \ldots, n_d$ через пробел — размеры доски вдоль каждой оси ($1 \leqslant n_i \leqslant 100\,000$).

В третьей строке находится d целых чисел $q_1, q_2, \ldots, q_d$ через пробел — координаты ферзя ($1 \leqslant q_i \leqslant n_i$).

Формат выходных данных

Выведите одно целое число — количество полей на доске $n_1 \times n_2 \times \ldots \times n_d$, которые бьёт ферзь из клетки $(q_1, q_2, \ldots, q_d)$, по модулю $998\,244\,353$.

Система оценки

В этой задаче пять подгрупп тестов. В первой подгруппе $d = 1$, $n_i \leqslant 1000$, она стоит **4** балла. Чтобы их получить, нужно пройти первый тест из условия и все тесты этой подгруппы.

Во второй подгруппе $d = 2$, $n_1 \leqslant n_2 \leqslant 30$, она стоит **10** баллов. Чтобы их получить, нужно пройти первые два теста из условия и все тесты этой подгруппы.

В третьей подгруппе $d \leqslant 6$, $n_i \leqslant 19$, она стоит **10** баллов. Чтобы их получить, нужно пройти первые три теста из условия и все тесты этой подгруппы.

В четвёртой подгруппе $d \leqslant 12$, $n_i \leqslant 1000$, она стоит **38** баллов. Чтобы их получить, нужно пройти все тесты из условия, все тесты предыдущих подгрупп и все тесты этой подгруппы.

В последней подгруппе нет никаких ограничений, не обозначенных в условии и в формате входных данных, она стоит **38** баллов. Чтобы их получить, нужно пройти вообще все тесты: все тесты из условия, все тесты предыдущих подгрупп и все тесты этой подгруппы.

Примеры

стандартный ввод	***стандартный вывод***
1 9 6	8
2 8 8 1 7	21
4 1 2 3 4 1 1 1 1	11
10 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2	59048

Задача D. Собрать многоугольник

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

У вас есть n палок целой длины. Требуется выбрать из них наименьшее возможное количество палок так, чтобы из выбранных палок можно было составить строго выпуклый многоугольник, либо определить, что из имеющихся палок составить строго выпуклый многоугольник невозможно.

Формат входных данных

В первой строке дано целое число n — количество палок ($3 \leqslant n \leqslant 10^5$). Во второй строке даны n целых чисел $a_1, a_2, \ldots, a_n$, разделённые пробелами — длины палок ($1 \leqslant a_i \leqslant 10^9$).

Формат выходных данных

Выведите наименьшее количество сторон в строго выпуклом многоугольнике, который можно составить из имеющихся палок. Если составить строго выпуклый многоугольник невозможно, выведите «`-1`» (без кавычек).

Система оценки

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

Во всех тестах первой группы $3 \leqslant n \leqslant 4$. За прохождение первой группы можно получить **12** баллов.

В тестах второй группы $3 \leqslant n \leqslant 15$. За вторую группу можно получить ещё **36** баллов.

На тесты третьей группы не накладывается никаких дополнительных ограничений, то есть в ней $3 \leqslant n \leqslant 10^5$. За неё можно получить оставшиеся **52** балла.

Примеры

стандартный ввод	*стандартный вывод*
4 1 2 4 6	4
4 7 4 1 2	-1
4 4 2 5 1	3
3 1 2 1	-1
4 1 1 1 100	3

Пояснения к примерам

В первом примере строго выпуклый многоугольник можно составить, только если использовать все имеющиеся палки.

Во втором примере строго выпуклый многоугольник составить невозможно.

В третьем примере можно составить строго выпуклый многоугольник из всех четырёх палок, но он не наименьший, так как из палок с длинами 4, 2 и 5 можно составить невырожденный треугольник. Поэтому ответ для третьего примера равен 3.

В четвёртом примере всего три палки, и они не образуют невырожденный треугольник, поэтому ответ равен `-1`.

В пятом примере нельзя построить строго выпуклый многоугольник из всех четырёх палок, но можно из трёх палок длины 1, поэтому ответ равен 3.

Задача E. Минимизация паросочетания

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	4 секунды
Ограничение по памяти:	1024 мебибайта

Ваша задача — по трём целым числам n, Δ и δ построить граф на n вершинах, в котором наибольшая и наименьшая степени вершин равны Δ и δ, соответственно, чтобы размер максимального паросочетания в нём был как можно меньше. Это краткое описание задачи, ниже приведено полное формальное условие.

Пусть G — граф без петель и кратных рёбер. Обозначим через $\delta(G)$ наименьшую степень вершины в нём, $\Delta(G)$ — наибольшую степень вершины в нём, $v(G)$ — количество вершин, $\alpha'(G)$ — размер максимального паросочетания в G, то есть максимальное количество рёбер G, никакие два из которых не имеют общий конец.

Пусть $n > \Delta \geqslant \delta \geqslant 0$ — три целых числа. Обозначим через $\mathcal{G}(n, \Delta, \delta)$ множество всех графов, в которых $v(G) = n$, $\Delta(G) = \Delta$, $\delta(G) = \delta$. Назовём такую тройку целых чисел *хорошей*, если $\mathcal{G}(n, \Delta, \delta) \neq \varnothing$. Обратите внимание, что некоторые тройки не являются хорошими, например, $(2, 1, 0)$ или $(3, 1, 1)$.

Вам дана хорошая тройка (n, Δ, δ), найдите граф $G \in \mathcal{G}(n, \Delta, \delta)$, в котором $\alpha'(G)$ как можно меньше.

Формат входных данных

В единственной строке находится три целых числа n, Δ, δ, образующих хорошую тройку ($0 \leqslant \delta \leqslant \Delta < n \leqslant 400$).

Формат выходных данных

Выведите целое число m — число рёбер графа. В следующих m строках выведите по два различных целых числа e_i, f_i через пробел — концы очередного ребра ($1 \leqslant e_i, f_i \leqslant n$). Никакая пара чисел не должна повторяться дважды, даже в обратном порядке.

Система оценки

В этой задаче каждый тест, кроме двух примеров, оценивается независимо вещественным числом от 0 до t_i баллов, где t_i — стоимость этого теста. Оценка вашего решения равняется сумме оценок за каждый тест. Примеры оцениваются в 0 баллов, но на каждый из них должен быть выдан граф с правильным числом вершин, максимальной и минимальной степенью, чтобы решение было принято на проверку. Тесты избраны жюри так, чтобы покрывать как можно больше разнообразных вариантов параметров и соответствующих графов. Всего подготовлено 102 теста. Гарантируется, что тесты 1–2 — это примеры, тесты 3–14 имеют $2 \leqslant n \leqslant 8$, тесты 15–27 имеют $9 \leqslant n \leqslant 16$, тесты 28–40 имеют $17 \leqslant n \leqslant 50$. Стоимость каждого из тестов 3–40 равна $\frac{25}{56} \approx 0.44643$, а стоимость каждого из тестов 41–102 равна $\frac{75}{56} \approx 1.33929$. Эти числа подобраны так, чтобы каждый из тестов 3–40 стоил втрое меньше, чем каждый из тестов 41–102, а суммарная стоимость всех тестов была равна 100 баллам.

Если ваш вывод не описывает граф из $\mathcal{G}(n, \Delta, \delta)$, вы получите вердикт «`Wrong Answer`» или «`Presentation Error`» и 0 баллов за этот тест. Иначе пусть α' — размер максимального паросочетания в вашем ответе, A' — наименьший возможный размер максимального паросочетания. За прохождение теста i вы получите $\frac{1}{\log_2(\alpha' - A' + 1) + 1} \cdot t_i$ баллов, где t_i — максимальное количество баллов за этот тест, а $\log_2$ — двоичный логарифм.

Примеры

стандартный ввод	*стандартный вывод*
1 0 0	0
4 3 1	3 1 2 1 3 1 4

Пояснения к примерам

В первом примере существует только один граф из $\mathcal{G}(1,0,0)$ — граф без рёбер. Если бы тест стоил один балл, то за этот граф вы бы и получили один балл.

Во втором примере существует два неизоморфных графа в $\mathcal{G}(4,3,1)$. Один из них — полный двудольный граф $K_{1,3}$, который и приведён в выводе. В нём размер максимального паросочетания равен единице, и это и есть оптимальный ответ. При единичной стоимости теста вы бы получили один балл за данный граф. Второй граф в $\mathcal{G}(4,3,1)$ — это треугольник, из которого торчит ребро. В этом графе есть паросочетание размера 2, поэтому из одного балла за тест этот граф дал бы $\frac{1}{\log_2(2-1+1)+1} = \frac{1}{2}$ балла.

Но на самом деле оба примера дают 0 баллов вне зависимости от того, какой граф из $\mathcal{G}(n,\Delta,\delta)$ выведет ваша программа.

Задача F. Здесь был Вася

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

Вася — уличный художник. Ночью он гуляет по городу с рюкзаком, набитым баллончиками с краской. В укромных местах, увидев однотонную или просто некрасиво раскрашенную стену, Вася останавливается, достаёт краски и рисует на стене что придётся. К большому сожалению Васи, не все одобряют такое творчество, поэтому действовать приходится быстро.

Сейчас Вася нашёл очередную такую стену. На улице горит только один фонарь, поэтому видна только часть стены. Наш герой задумался: нет ли на стене следов его предыдущего рисунка? Если есть, рисовать на ней во второй раз не стоит — но если нет, нужно обязательно оставить ещё один уникальный след на стенах города!

Протокол взаимодействия

В этой задаче ваше решение будет запущено на каждом тесте два раза.

При первом запуске решение получит схематичное изображение видимой части стены. Стена условно разбита на клетки: пять клеток в высоту и 40 клеток в ширину. При этом фонарь освещает только часть стены шириной в 30 клеток. Каждая клетка покрашена в какой-то один цвет. Цвета обозначаются большими английскими буквами, разные буквы обозначают разные цвета.

Итак, решение получит пять строк, по 30 больших английских букв в каждой. Она уже раскрашена в какие-то цвета, в разных тестах по-разному. Вася ещё не касался этой стены. Нужно раскрасить её — как угодно, но чтобы потом можно было узнать своё творчество. У Васи есть три баллончика с краской: красной, зелёной и синей (обозначаются буквами «`R`», «`G`», «`B`»). Краски каждого цвета хватит, чтобы закрасить этим цветом не более 20 клеток на видимой части стены — независимо от того, какого цвета они были до этого.

После этого могут случиться по порядку три события:

- Придёт Петя — другой уличный художник — и нарисует на этой же стене свою картину — возможно, поверх Васиной. У Пети три баллончика с красками тех же трёх цветов, что и у Васи, но они меньше: каждого хватит не более чем на 10 клеток стены.
- Придёт дворник и закрасит какую-то часть стены шириной в 10 клеток белым цветом (обозначается буквой «`W`»).
- Придёт электрик и перевесит фонарь, в результате чего ночью будет видна другая часть стены шириной в 30 клеток (напомним, что общая ширина стены — 40 клеток).

В каждом тесте все эти действия, как и сам факт их наличия, зафиксированы заранее и не зависят от действий Васи. Петя и дворник могут работать с любой частью стены — даже с той, которая не видна ночью.

После всего этого решение участника будет запущено второй раз. Решение опять получит пять строк, по 30 больших английских букв в каждой — состояние стены после всех действий. В этом запуске нужно понять, что Вася уже рисовал на этой стене, и больше ничего не делать.

Задача состоит в том, чтобы по заданному изображению распознать, первый это запуск на тесте или второй.

Формат входных данных

При каждом запуске решение получает изображение видимой части стены — пять строк, каждая из которых состоит ровно из 30 заглавных английских букв.

Формат выходных данных

Если решение считает, что Вася ещё не рисовал на этой стене — то есть это первый запуск на тесте, — выведите в первой строке «No». В следующих пяти строках выведите по 30 символов, задающих рисунок Васи на видимой части стены. Каждый символ может быть точкой («.») или одной из трёх букв «R», «G», «B». Точки означают, что клетка останется нетронутой, а буквы говорят о том, что нужно покрасить клетку в соответствующий цвет: красный, зелёный или синий. Каждая буква должна встречаться в рисунке не более 20 раз.

Если же решение считает, что Вася уже рисовал на этой стене — то есть это второй запуск на тесте, — просто выведите в первой строке «Yes».

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***
WWWWWWFBRHMVEGNOIJTMKWWWWWWWWW WWWWPLOAALVBZLAMDNCBXKWWWWWWWWW WWWWWQZASXDCFGVKBGJNLMXWWWWWWWW WWWWWWIWUERPTVEIQLWWWWWWWWWWWW WWWWWWWWWWWWWWWWWWWWWWWWWWWW	No ....RRR.......GG......BBB..... ....R..R.....G..G.....B..B.... ....RRR......G........BBB..... ....R.R......G.GG.....B..B.... ....R..R......GGG.....BBB.....
WWWWRRRBRHMVEGGGIJTMKWBBBWWWWW WWWWRLORALVBZGAMGNCBXKBWWBWWWW WWWWRRRASXDCFGVKBGJNLMBBBWWWWW WWWWRWRWUERPTGEGGLWWWWBWWBWWWW WWWWRWWRWWWWWWGGGWWWWWBBBWWWWW	Yes

Система оценки

Тесты состоят из примера и восьми групп, каждая из которых даёт 12.5 баллов.

В группах 2, 4, 6 и 8 на стене рисует Петя.

В группах 3, 4, 7 и 8 приходит дворник.

В группах 5, 6, 7 и 8 приходит электрик.

Чтобы получить баллы за группу, в которой происходит множество действий S, нужно пройти все тесты, где множество действий является подмножеством S. Например, баллы за группу 7 можно получить, только пройдя все тесты из групп 1, 3, 5 и 7, а также пример из условия. Итоговое количество баллов округляется к ближайшему целому числу (к чётному, если дробная часть равна 0.5).

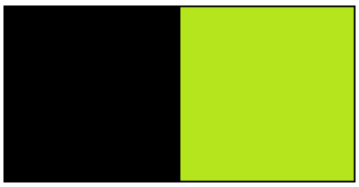
Задача G. Связная фигура

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

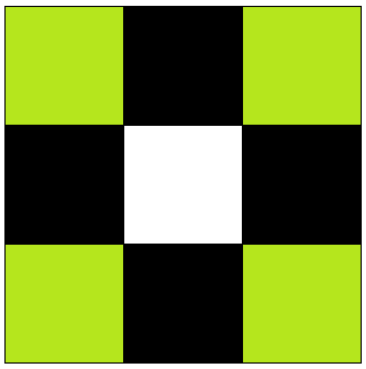
Бесконечный лист клетчатой бумаги раскрашен в шахматном порядке. Какое наименьшее количество белых клеток может быть в связной по стороне клетчатой фигуре, в которой ровно n чёрных клеток?

Формально, клетчатая фигура — это произвольное конечное множество клеток листа. Фигура называется *связной*, если от любой клетки фигуры можно дойти до любой другой, перемещаясь только в соседние по стороне клетки фигуры (выходить за пределы фигуры нельзя).

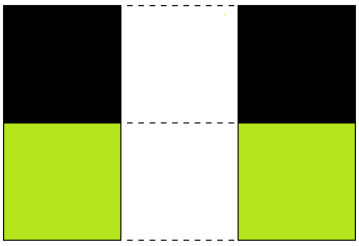
Примеры ниже иллюстрируют, какие фигуры — связные, а какие — нет. В каждом из них для наглядности картинки закрашены только клетки, принадлежащие фигуре, причём белые клетки фигуры закрашены салатовым цветом, чтобы отличаться от белого фона.



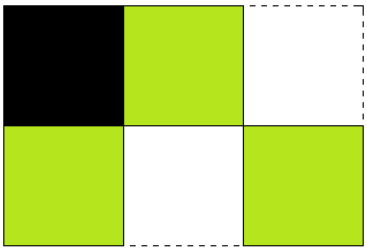
Связная фигура из одной чёрной и одной белой клетки.



Связная фигура с «дыркой». Она состоит из 4 белых и 4 чёрных клеток. Центральная клетка — тоже белая, но не является частью фигуры.



Несвязная фигура — от её левой половины нельзя дойти до правой, оставаясь внутри фигуры.



Всё ещё несвязная фигура, так как ходить можно только между соседними по стороне клетками (касания по углу недостаточно).

Формат входных данных

В единственной строке дано целое число n — количество чёрных клеток в фигуре ($1 \leqslant n \leqslant 10^{18}$).

Формат выходных данных

Выведите одно число — наименьшее возможное количество белых клеток в связной клетчатой фигуре с n чёрными клетками.

Система оценки

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

Во всех тестах первой группы $1 \leqslant n \leqslant 5$. За прохождение всех тестов первой группы можно получить 21 балл.

В тестах второй группы $1 \leqslant n \leqslant 10^5$. За вторую группу можно получить ещё 42 балла.

На тесты третьей группы не накладывается никаких дополнительных ограничений, то есть в ней $1 \leqslant n \leqslant 10^{18}$. За неё можно получить оставшиеся 37 баллов.

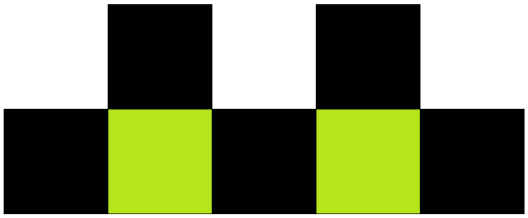
Примеры

стандартный ввод	***стандартный вывод***
1	0
5	2

Пояснения к примерам

В первом примере можно взять фигуру, состоящую из одной чёрной клетки, поэтому ответ на него равен 0.

Можно доказать, что в связной клетчатой фигуре с 5 чёрными клетками всегда будет хотя бы 2 белых клетки. Пример с 2 белыми клетками можно увидеть ниже.



Возможный пример для $n = 5$.

Задача H. Тропический минимум

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

Миша получил загадочную посылку из далёкой страны. Открыв её, он обнаружил внутри тропический калькулятор, а также массив целых чисел. При ближайшем рассмотрении Миша понял, что калькулятор умеет вычислять результат довольно длинных выражений с операциями сложения и умножения, однако не умеет обрабатывать скобки. При ещё более детальном рассмотрении он обнаружил, что в качестве результата сложения двух чисел калькулятор возвращает минимальное из чисел вместо их суммы, а в качестве результата умножения — сумму множителей вместо произведения. Мише сразу же стало интересно, какое **минимальное** число может получиться, если он расставит между числами в присланном ему массиве операции сложения и умножения, после чего вычислит на калькуляторе результат полученного выражения. Обратите внимание, что числа в массиве переставлять нельзя, а приоритет умножения у калькулятора выше, чем у сложения, несмотря на то, что сами операции необычны.

Формат входных данных

В первой строке задано одно число T — количество наборов входных данных, для которых нужно решить задачу ($1 \leqslant T \leqslant 10^5$).

В первой строке каждого набора задано число n — размер массива ($1 \leqslant n \leqslant 10^6$). В следующей строке заданы через пробел n целых чисел $a_1, a_2, \ldots, a_n$ — элементы массива ($-2^{31} \leqslant a_i < 2^{31}$).

Гарантируется, что сумма n по всем наборам входных данных не превосходит 10^6.

Формат выходных данных

Для каждого набора входных данных выведите в отдельной строке одно число — ответ на задачу.

Система оценки

Задача содержит восемь подзадач. Для того, чтобы решение было принято на проверку, необходимо, чтобы оно проходило тест из условия. Баллы за каждую подзадачу будут начислены, если пройдены все тесты этой и всех предыдущих подзадач. Исключения составляют третья подзадача, для прохождения которой не требуется прохождение всех тестов второй подзадачи (но нужно пройти все тесты первой подзадачи), а также последняя подзадача, состоящая из 40 тестов, каждый из которых оценивается индивидуально в 1 балл.

Подзадача	Баллы	Ограничения		
		T	n	a_i
1	10	$T = 1$	$n \leqslant 10$	$\lvert a_i \rvert \leqslant 100$
2	5	$T \leqslant 10^5$	$n \leqslant 10$	$\lvert a_i \rvert \leqslant 100$
3	5	$T \leqslant 10$	$n \leqslant 20$	$\lvert a_i \rvert \leqslant 100$
4	5	$T \leqslant 10$	$n \leqslant 100$	$\lvert a_i \rvert \leqslant 100$
5	5	$T \leqslant 10$	$n \leqslant 100$	$-2^{31} \leqslant a_i < 2^{31}$
6	10	$T \leqslant 10$	$n \leqslant 1000$	$-2^{31} \leqslant a_i < 2^{31}$
7	20	$T = 1$	$n \leqslant 10^4$	$-2^{31} \leqslant a_i < 2^{31}$
8	40	$T \leqslant 10^5$	$n \leqslant 10^6$	$-2^{31} \leqslant a_i < 2^{31}$

Пример

стандартный ввод	***стандартный вывод***
4 7 -1 -1 -1 2 -1 -1 -1 3 7 3 6 5 10 -50 20 -100 30 3 -100 100 100	-4 3 -130 -100

Задача I. Тропический максимум

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

Миша получил загадочную посылку из далёкой страны. Открыв её, он обнаружил внутри тропический калькулятор, а также массив целых чисел. При ближайшем рассмотрении Миша понял, что калькулятор умеет вычислять результат довольно длинных выражений с операциями сложения и умножения, однако не умеет обрабатывать скобки. При ещё более детальном рассмотрении он обнаружил, что в качестве результата сложения двух чисел калькулятор возвращает минимальное из чисел вместо их суммы, а в качестве результата умножения — сумму множителей вместо произведения. Мише сразу же стало интересно, какое **максимальное** число может получиться, если он расставит между числами в присланном ему массиве операции сложения и умножения, после чего вычислит на калькуляторе результат полученного выражения. Обратите внимание, что числа в массиве переставлять нельзя, а приоритет умножения у калькулятора выше, чем у сложения, несмотря на то, что сами операции необычны.

Формат входных данных

В первой строке задано одно число T — количество наборов входных данных, для которых нужно решить задачу ($1 \leqslant T \leqslant 10^5$).

В первой строке каждого набора задано число n — размер массива ($1 \leqslant n \leqslant 10^6$). В следующей строке заданы через пробел n целых чисел $a_1, a_2, \ldots, a_n$ — элементы массива ($-2^{31} \leqslant a_i < 2^{31}$).

Гарантируется, что сумма n по всем наборам входных данных не превосходит 10^6.

Формат выходных данных

Для каждого набора входных данных выведите в отдельной строке одно число — ответ на задачу.

Система оценки

Задача содержит восемь подзадач. Для того, чтобы решение было принято на проверку, необходимо, чтобы оно проходило тест из условия. Баллы за каждую подзадачу будут начислены, если пройдены все тесты этой и всех предыдущих подзадач. Исключения составляют третья подзадача, для прохождения которой не требуется прохождение всех тестов второй подзадачи (но нужно пройти все тесты первой подзадачи), а также последняя подзадача, состоящая из 40 тестов, каждый из которых оценивается индивидуально в 1 балл.

Подзадача	Баллы	Ограничения		
		T	n	a_i
1	10	$T = 1$	$n \leqslant 10$	$\lvert a_i \rvert \leqslant 100$
2	5	$T \leqslant 10^5$	$n \leqslant 10$	$\lvert a_i \rvert \leqslant 100$
3	5	$T \leqslant 10$	$n \leqslant 20$	$\lvert a_i \rvert \leqslant 100$
4	5	$T \leqslant 10$	$n \leqslant 100$	$\lvert a_i \rvert \leqslant 100$
5	5	$T \leqslant 10$	$n \leqslant 100$	$-2^{31} \leqslant a_i < 2^{31}$
6	10	$T \leqslant 10$	$n \leqslant 1000$	$-2^{31} \leqslant a_i < 2^{31}$
7	20	$T = 1$	$n \leqslant 10^4$	$-2^{31} \leqslant a_i < 2^{31}$
8	40	$T \leqslant 10^5$	$n \leqslant 10^6$	$-2^{31} \leqslant a_i < 2^{31}$

Пример

стандартный ввод	***стандартный вывод***
4 7 -1 -1 -1 2 -1 -1 -1 3 7 3 6 5 10 -50 20 -100 30 3 -100 100 100	-1 16 -50 100

Задача J. Раздача подарков

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

Скоро Новый Год! Зайчата помогли Деду Морозу упаковать все подарки для детей. Теперь настало время им самим получить подарки!

Всего Деду Морозу помогали n зайчат, пронумерованных целыми числами от 1 до n. Дед Мороз собирается разложить вдоль волшебного коридора n подарков, также пронумерованных по порядку целыми числами от 1 до n. Чтобы получить подарки, зайчата будут по очереди заходить в коридор. Про каждого зайчонка известно, какие из n подарков ему нравятся, а какие нет. Зайчонок прыгает вперёд до **первого** из оставшихся подарков, который ему нравится, забирает его себе и идёт отдыхать. После этого в коридор заходит следующий зайчонок.

Например, пусть $n = 3$, первому зайчонку нравятся все три подарка, второму — только подарок номер 3, а третьему — подарки с номерами 1 и 2. Сначала первый зайчонок возьмёт себе подарок номер 1. Затем второй зайчонок пропустит подарок номер 2 и возьмёт подарок номер 3. Наконец, третий зайчонок возьмёт подарок номер 2 — ему бы понравился и подарок номер 1, но этого подарка в коридоре уже нет.

Увы, может так случиться, что очередной зайчонок зайдёт в коридор, но ни один из оставшихся подарков ему не понравится. Но Дед Мороз на то и волшебник, чтобы этого не допустить!

В этом году Дед Мороз увлёкся алгеброй. Теперь, стоит лишь ему подумать про какую-то квадратную матрицу размера $n \times n$ из нулей и единиц — и подарки окажутся такими, что:

- Если в i-й строке и j-м столбце стоит 1, то i-му зайчонку нравится j-й подарок.
- Если в i-й строке и j-м столбце стоит 0, то i-му зайчонку не нравится j-й подарок.

Квадратная матрица размера $n \times n$ из нулей и единиц называется *счастливой*, если в случае, когда Дед Мороз подумает про эту матрицу, все зайчата получат подарки. Сколько существует различных счастливых матриц? Поскольку их количество может быть очень большим, выведите его остаток от деления на $10^9 + 7$.

Формат входных данных

В единственной строке дано целое число n ($1 \leqslant n \leqslant 10^5$).

Формат выходных данных

Выведите одно число — количество счастливых матриц по модулю $10^9 + 7$.

Система оценки

В этой задаче 22 теста: два примера и 20 скрытых тестов. Если ваше решение проходит оба примера, то за каждый из пройденных скрытых тестов вы получите по 5 баллов.

Гарантируется, что первые 10 скрытых тестов — это все целые числа от 3 до 12.

Примеры

стандартный ввод	*стандартный вывод*
1	1
2	6

Пояснения к примерам

В первом примере подходит только матрица (1).

Во втором примере 6 счастливых матриц: $\begin{pmatrix}1&0\\0&1\end{pmatrix}$, $\begin{pmatrix}1&0\\1&1\end{pmatrix}$, $\begin{pmatrix}1&1\\0&1\end{pmatrix}$, $\begin{pmatrix}1&1\\1&1\end{pmatrix}$, $\begin{pmatrix}0&1\\1&0\end{pmatrix}$ и $\begin{pmatrix}0&1\\1&1\end{pmatrix}$.

Матрица $\begin{pmatrix}1&1\\1&0\end{pmatrix}$ **не** является счастливой: сначала первый зайчонок возьмёт себе первый подарок, затем в коридор зайдёт второй зайчонок — но оставшийся подарок ему не понравится. Также не является счастливой, например, матрица, целиком состоящая из нулей.

Разбор задачи «Сумма дробей с округлением»

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

Наивное решение — вычислять и складывать дроби до тех пор, пока очередная из них не будет иметь целую часть 0. Такое решение реализуется легко, работает за $O(\sqrt{n})$, и на компилируемых языках уже получает 90 с чем-то баллов! Возникает закономерный вопрос: стоят ли несколько оставшихся баллов дополнительных усилий?

Предположим, что более простые способы получить баллы кончились, и ответ на указанный вопрос положительный. Посмотрим, как ведут себя слагаемые на каком-нибудь примере побольше, например, при $n = 1000$:

$$\left\lfloor \frac{999}{1} \right\rfloor + \left\lfloor \frac{998}{4} \right\rfloor + \left\lfloor \frac{997}{9} \right\rfloor + \left\lfloor \frac{996}{16} \right\rfloor + \left\lfloor \frac{995}{25} \right\rfloor + \left\lfloor \frac{994}{36} \right\rfloor + \left\lfloor \frac{993}{49} \right\rfloor + \left\lfloor \frac{992}{64} \right\rfloor + \left\lfloor \frac{991}{81} \right\rfloor + \left\lfloor \frac{990}{100} \right\rfloor + \left\lfloor \frac{989}{121} \right\rfloor + \left\lfloor \frac{988}{144} \right\rfloor + \dots$$

$$= 999 + 249 + 110 + 62 + 39 + 27 + 20 + 15 + 12 + 9 + 8 + 6 + \dots$$

Видно, что числа вначале очень быстро убывают. С другой стороны, все следующие числа будут не больше шести — но ещё встретится почти два десятка ненулевых слагаемых.

Это может натолкнуть на следующую идею: давайте отдельно решим задачу для больших слагаемых и отдельно — для маленьких. Пока очередное слагаемое не меньше какого-то предела k, будем прибавлять его к ответу и вычислять следующее, как в наивном решении. А слагаемые, которые меньше k, посчитаем так: для каждого $t < k$ найдём количество слагаемых, равных t — пусть это $c(t)$ — и прибавим к ответу величину $t \cdot c(t)$.

Как понять, чему равно $c(t)$? Сначала напишем, когда слагаемое равно t: это означает, что $t = \left\lfloor \frac{n-x}{x^2} \right\rfloor$, то есть $\frac{n-x}{x^2} \leqslant t < \frac{n-x}{x^2} + 1$. Можно честно порешать двойное неравенство, а можно для каждого t найти границы x двоичным поиском. В вычислениях следует избежать 64-битных вещественных чисел (`double`), так как с ними, скорее всего, для вычислений не хватит точности. Можно считать в целых числах или в 80-битных вещественных.

Как выбрать предел k? Первая часть решения работает за $O(n/k^2)$, а вторая — за $O(k)$ или $O(k \log n)$. Поэтому для решения с формулой подходит граница $k = \sqrt[3]{n}$. Учитывая ограничения на n, можно использовать константу $k = 10^6$. Для решения с двоичным поиском границу k можно взять чуть меньше.

Разбор задачи «Половинки»

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

Решение этой задачи — динамическое программирование по подмножествам.

Сначала построим граф для удобного решения задачи. В графе будет n вершин, i-я — состояние книги, в котором открыты развороты с i-м цветом краски и i-м предметом. Рёбра — переходы между этими состояниями. Длина каждого ребра — количество перелистываний, суммарно сверху и снизу.

Теперь переформулируем нашу задачу. Дан граф, нужно на нём найти кратчайший путь, который посещает каждую вершину ровно один раз. Это классическая задача, известная как «задача коммивояжёра».

Решение, работающее за $O(2^n \cdot n^2)$ времени и $O(2^n \cdot n)$ памяти, вкратце устроено так. Рассмотрим все возможные состояния (S, u) вида «посещено подмножество S, а последняя посещённая вершина u». База: в начальные состояния вида $(\{u\}, u)$ мы можем попасть за время, требуемое, чтобы перелистать книгу от первых разворотов до разворотов с u-м цветом краски и u-м предметом. Переходы: в состояние (S, u) мы могли попасть изо всех состояний вида $(S \backslash \{u\}, v)$, где вершина $v \in S \backslash \{u\}$, добавив время прохода по ребру из вершины v в

вершину u. Изо всех этих способов нужно выбрать тот, в котором суммарное время как можно меньше. Ответ: минимальное из значений в состояниях вида (T, u), где множество T содержит все n вершин нашего графа.

Как обычно в динамике по подмножествам, будем кодировать все возможные множества S двоичными числами от 0 до $2^n - 1$. Чтобы восстановить сами перелистывания, кроме минимального времени в каждом состоянии будем хранить, из какой вершины v мы совершили оптимальный переход.

Разбор задачи «Многомерные ферзи»

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

Назовём *расстоянием Чебышёва* между двумя клетками $Q = (q_1, \ldots, q_d)$ и $S = (s_1, \ldots, s_d)$ величину $\ell_\infty(Q, S) = \max_{i \in \{1,\ldots,d\}} |q_i - s_i|$, то есть число ходов многомерного короля, необходимое для того, чтобы добраться от одной клетки до другой. Здесь и далее через d мы обозначаем размерность шахматной доски.

Пусть ферзь стоит в клетке Q. Посчитаем число A_ℓ — это сколько клеток S, находящихся на расстоянии Чебышёва $\ell > 0$ от Q, бьёт этот ферзь. По определению вдоль каждой оси i координата s_i либо совпадает с q_i, либо отличается ровно на ℓ, то есть $s_i \in \{q_i - \ell, q_i, q_i + \ell\}$. Некоторые из этих вариантов *допустимые*, то есть находятся в пределах от 1 до n_i (где n_i — ширина доски вдоль i-й оси). Так как $q_i - \ell$ может оказаться не больше нуля, а $q_i + n_i$ может оказаться больше n_i, допустимых значений координаты — от одной до трёх. Обозначим через a_i их количество.

Важно заметить, что, находя одну из подходящих клеток S, можно выбирать каждую координату из допустимого множества независимо от выбора вдоль других осей; единственное накладываемое на нас ограничение заключается в том, что нельзя вдоль всех осей взять $s_i = q_i$, хотя бы раз необходимо выбрать другое число. Это комбинаторное рассуждение приводит нас к формуле $A_\ell = \prod_{i=1}^{d} a_i - 1$. Для удобства дальнейшей работы с этим выражением обозначим $P_\ell = \prod_{i=1}^{d} a_i$, тогда $A_\ell = P_\ell - 1$.

Пока что вышесказанное приводит нас к алгоритму, работающему за $\mathcal{O}(Nd)$, где N — максимальная ширина доски вдоль одной из осей. А именно, переберём ℓ от 1 до N, посчитаем все a_i для каждого такого ℓ и за $\mathcal{O}(d)$ найдём A_ℓ. В конце все эти числа сложим.

Теперь попробуем этот алгоритм соптимизировать. Заметим, что, когда мы перебираем в порядке возрастания все возможные ℓ, каждое a_i с каждым шагом либо не изменяется, либо уменьшается. Более того, моменты, когда оно уменьшается, легко посчитать: наименьшее ℓ, при котором a_i становится двойкой, равно $\min\{q_i, n_i - q_i + 1\}$, а наименьшее ℓ, при котором a_i становится единицей, равно $\max\{q_i, n_i - q_i + 1\}$ (если эти два числа равны, то a_i сразу станет единицей без промежуточной фазы, в которой оно равно двум). Давайте сделаем массив длины N из списков, в ℓ-м списке будем хранить номера осей, вдоль которых число a_i уменьшается именно при данном ℓ (и будем хранить дважды, если a_i в этот момент падает с тройки сразу до единицы). Этот список легко заполнить за $\mathcal{O}(d)$. Затем будем перебирать все возможные ℓ и для них считать P_ℓ, в каждый момент храня текущие a_i. Для удобства будем считать, что $P_0 = 3^d$, то есть все $a_i = 3$, но A_0 не будем учитывать в ответе. Для каждого ℓ надо просто вычислить P_ℓ, зная $P_{\ell-1}$ и список тех осей, вдоль которых произошли изменения при данном ℓ. Возьмём в качестве предварительного значения P_ℓ уже посчитанное $P_{\ell-1}$ и переберём номера в списке номеров осей, постепенно меняя с этим P_ℓ. Если там есть i-я ось и $a_i = 3$, то запишем новое значение $a_i = 2$ и заменим P на $\frac{2}{3}P$. Здесь важно знать, что в кольце остатков по модулю натурального числа M можно не только складывать, вычитать и умножать, но также и делить на число, взаимно простое с M. В частности, по модулю $998\,244\,353$ вместо домножения на $\frac{2}{3}$ можно домножать на $\frac{2+2\cdot 998\,244\,353}{3} = 665\,496\,236$. Аналогично, если в списке нашлась ось i, для которой $a_i = 2$, то надо заменить a_i на единицу и домножить P_ℓ на $\frac{1}{2}$, или же на $\frac{1+998\,244\,353}{2} = 499\,122\,177$. (В частности, если i хранится в одном и том же списке дважды, то и домножим мы P_ℓ последовательно на $\frac{2}{3}$ и на $\frac{1}{2}$; при

желании можно было хранить в отдельном массиве списков те оси, для которых при данном ℓ значение a_i падает с трёх до одного, и за каждую из таких осей домножать P_ℓ сразу на $\frac{2}{3} \cdot \frac{1}{2} = \frac{1}{3} \equiv \frac{1+998\,244\,353}{3} = 332\,748\,118 \pmod{998\,244\,353}$.) Наконец, посчитав P_ℓ, легко найдём A_ℓ, которое на единицу меньше, и прибавим его к ответу. Также есть альтернатива: находить сумму всех P_ℓ, а в конце вычесть N — по единичке за каждое ℓ. Асимптотика этого способа $\mathcal{O}(N + d)$: N за перебор всех ℓ, а d за перебор всех осей и сохранение в списках данных об этих осях.

В терминах вычислительной сложности наше решение всё ещё несовершенно, так как оно не работает за полиномиальное время от входных данных. В самом деле, в асимптотике участвует слагаемое N, но числа во входных данных имеют длину не N, а $\log N$. Например, если бы размеры доски были не до 10^5, а до 10^{100}, то входные данные были бы всё ещё сравнительно небольшие, может быть, меньше мегабайта, в то время как решение не отработало за время жизни вселенной. Поэтому покажем, как избавиться от слагаемого N за счёт небольшого увеличения слагаемого d. А именно, давайте привяжем к перебору всех возможных ℓ течение времени: в t-й момент времени $\ell = t$. То, что a_i изменилось при некотором значении ℓ, будет событием, произошедшим в момент ℓ. Для каждого i у нас есть два события: значение стало двойкой и значение стало единицей (либо одно, если реализация предпочитает склеивать два одновременных события вдоль одной оси). Запишем все эти события в массив и отсортируем его по ℓ. Далее будем перебирать эти события одно за другим и обрабатывать, как и прежде — изменять a_i и домножать текущее значение P_ℓ на $\frac{2}{3}$ или $\frac{1}{2}$. Если у очередного события момент ℓ' больше, чем момент ℓ предыдущего события, значит, мы можем прибавить к ответу текущее A_ℓ. Более того, если $\ell' \neq \ell + 1$, то тогда $A_\ell = A_{\ell+1} = \ldots = A_{\ell'-1}$, и можно разом прибавить к ответу их сумму $A_\ell \cdot (\ell' - \ell)$. В таком случае эта фаза алгоритма отработает за $\mathcal{O}(d)$, и самой «тяжеловесной» частью будет начальная сортировка событий, работающая за время $\mathcal{O}(d \log d)$. Это и есть время работы всего алгоритма.

Примечание. В асимптотике $\mathcal{O}(d \log d)$ мы не учитываем работу с длинными числами. На самом деле, конечно, времени $\mathcal{O}(d \log d)$ не хватит даже на то, чтобы считать входные данные. Если считать модуль $M = 998\,244\,353$ константным, то перед запуском нормального решения потребуется ещё $\mathcal{O}(d \log N)$, или, точнее, $\mathcal{O}(\text{len}(\text{Input}))$ времени на то, чтобы свести решение к работе с числами константной длины, и тогда решение будет с асимптотикой $\mathcal{O}(d \log d + \text{len}(\text{Input})) = \mathcal{O}(d \log Nd)$. Но в теории сложности вычислений принято меньше внимания обращать на сложности арифметических операций и больше — на суть алгоритма.

Разбор задачи «Собрать многоугольник»

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

Мы будем пользоваться следующим геометрическим фактом: если у вас есть k палок (где $k \geqslant 3$) положительных длин $\ell_1 \leqslant \ell_2 \leqslant \ldots \leqslant \ell_k$, то из них можно составить строго выпуклый многоугольник тогда и только тогда, когда $\ell_k < \ell_1 + \ell_2 + \ldots + \ell_{k-1}$ (то есть когда самая длинная палка короче суммы всех оставшихся).

Из этого факта уже следует решение за время $O(n^2)$. Не умаляя общности, будем считать, что $a_1 \leqslant a_2 \leqslant \ldots \leqslant a_n$, то есть палки пронумерованы в порядке возрастания их длин. Переберём в порядке возрастания k — число сторон в искомом выпуклом многоугольнике.

Далее переберём i — самый большой номер взятой в многоугольник палки. Очевидно, $i \geqslant k$, иначе взять k палок с номерами не больше i не получится. Более того, по вышеупомянутому факту, если мы можем составить строго выпуклый многоугольник из i-й палки и ещё *каких-то* $k - 1$ палок с номерами, меньшими i, то составить строго выпуклый многоугольник из $i - k + 1$-й, $i - k + 2$-й, …, i-й палок получится и подавно: мы не поменяли длину самой длинной палки, но не уменьшили длины всех остальных палок.

Таким образом, для каждого $i \geqslant k$ нужно проверить, правда ли, что верно $a_{i-k+1} + a_{i-k+2} + \ldots + a_{i-1} > a_i$. Если это так, то можно составить строго выпуклый многоугольник из k палок, для которого i — самый большой номер взятой палки. Иначе — нельзя. Проверить это неравенство можно за $O(1)$, если заранее посчитать префиксные суммы для массива a. Итоговая асимптотика — $O(n^2)$, так как нужно перебрать k и i.

Но оказывается, что это решение на самом деле работает быстрее! А именно, оно работает за $O(n \cdot \mathrm{ans})$, где ans — ответ на задачу, если строго выпуклый многоугольник составить можно, и n, если нельзя. Докажем, что $\mathrm{ans} = O(\log C)$, где C — верхнее ограничение на длины палок (в задаче $C = 10^9$).

Начнём с чуть более простого утверждения: если ответа нет, то $n = O(\log C)$. Действительно, пусть ни из какого подмножества палок с длинами $a_1 \leqslant a_2 \leqslant \ldots \leqslant a_n$ нельзя составить строго выпуклый многоугольник. Тогда $1 \leqslant a_1 \leqslant a_2$ (так как все длины целые и положительные), $a_3 \geqslant a_1 + a_2 \geqslant 1 + 1 = 2$ (иначе из первой, второй и третьей палок можно составить строго выпуклый многоугольник), $a_4 \geqslant a_3 + a_2 + a_1 \geqslant 2 + 1 + 1 = 4$, $a_5 \geqslant a_4 + a_3 + a_2 + a_1 = 4 + 2 + 1 + 1 = 8$ и так далее. По индукции получается, что $a_d \geqslant 2^{d-2}$. Таким образом, если $n > \log_2 C + 2$, то $a_n > C$, что невозможно.

Следовательно, для любых $\lceil \log_2 C \rceil + 2$ палок *какой-то* ответ существует. Следовательно, если палок *больше*, чем $\lceil \log_2 C \rceil + 2$, то точно существует какой-то ответ, состоящий из не более чем $\lceil \log_2 C \rceil + 2$ палок: возьмём только первые $\lceil \log_2 C \rceil + 2$ палок и найдём какой-то ответ среди них. Таким образом, $\mathrm{ans} \leqslant \lceil \log_2 C \rceil + 2$, а наше решение работает за время $O(n \log C + n \log n)$, где $O(n \log n)$ нужно на сортировку длин.

Разбор задачи «Минимизация паросочетания»

Авторы задачи:	Дмитрий Карпов Михаил Иванов
Подготовка тестов и решений:	Михаил Иванов
Автор разбора:	Михаил Иванов

В системе для подготовки задач эта задача разрабатывалась под кодовым названием «`appalling-matchings`» — *ужасающие паросочетания*. Это неспроста!

Чтобы набрать частичные баллы, участники в основном писали решения, основанные на жадности или на других эвристиках, и эта задача была своеобразным «полем битвы» эвристик — чья сможет «раскусить» больше разных типов параметров. Жюри пыталось сделать так, чтобы оценки от нуля до ста распределялись примерно равномерно между людьми, пытавшимися решить эту задачу, но на деле получилось по-другому: практически любой, кто писал приличный жадный алгоритм, получал от 40 до 60 баллов, и лишь один участник набрал больше — он набрал 86 баллов. Здесь же мы расскажем, как получить полный балл за задачу.

Сложность задачи в том, что в пространстве параметров n, Δ, δ есть довольно много существенно разных областей, и в этой куче случаев очень непросто разобраться. С этой кропотливой работой в серии статей справились П. Эрдёш, Л. Поса, Б. Боллобаш, С. Эллдридж в 1962–1976 годах. Доктор физико-математических наук Дмитрий Карпов, составляя свой курс по паросочетаниям и факторам графов, нашёл все эти статьи и составил из их результатов общую картину, благодаря чему эта задача и появилась на свет. Чтобы не превращать этот разбор в лекцию на несколько часов, мы опустим все доказательства и лишь приведём формулы и соответствующие графы в разных случаях.

Предварительно введём и напомним полезные обозначения. Через $\delta(G)$ обозначается наименьшая степень вершины в графе G, через $\Delta(G)$ — наибольшая степень вершины в нём, $v(G)$ — количество вершин, $\alpha'(G)$ — размер максимального паросочетания в G, то есть максимальное количество рёбер G, никакие два из которых не имеют общий конец. $\mathcal{G}(n, \Delta, \delta)$ — множество всех графов, в которых $v(G) = n$, $\Delta(G) = \Delta$, $\delta(G) = \delta$. Через K_n обозначается полный граф на n вершинах, то есть такой граф, в котором проведены все возможные $\frac{n(n-1)}{2}$ рёбер между его вершинами. $K_{m,n}$ — полный двудольный граф на $m + n$ вершинах с долями

размерами m и n (внутри долей рёбер не проводится, а между долями проводятся все mn возможных рёбер). $\lfloor\alpha\rfloor$ — это нижняя целая часть вещественного числа α, то есть наибольшее целое число, не превосходящее α. $\lceil\alpha\rceil$ — это верхняя целая часть вещественного числа α, то есть наименьшее целое число, не меньшее α. Также обозначим через R_n^d d-регулярный граф на n вершинах (то есть такой граф на n вершинах, в котором степени всех вершин равны d). Такой граф почти всегда не единственный, но нас устроит любой из них. Такой граф существует, если хотя бы одно из чисел n и d чётно и $0 \leqslant d < n$. Например, можно расставить n вершин по кругу в вершины правильного n-угольника и каждую соединить с $\left\lfloor\frac{d}{2}\right\rfloor$ соседями с одной стороны и с $\left\lfloor\frac{d}{2}\right\rfloor$ соседями с другой стороны. Так мы получим $2\left\lfloor\frac{d}{2}\right\rfloor$-регулярный граф, и при чётных d он нам подходит. При нечётных d этот граф является $(d-1)$-регулярным, и, чтобы получить d-регулярный, надо дополнить этот граф совершенным паросочетанием; и это легко сделать, ведь, раз d нечётно, n чётно, и можно в качестве совершенного паросочетания взять все $\frac{n}{2}$ главных диагоналей нашего n-угольника. Самый маленький d-регулярный граф — это K_{d+1}, и, разумеется, это единственный d-регулярный граф на $d+1$ вершинах.

Перейдём к задаче. Самая очевидная оценка на максимальное паросочетание в любом графе $0 \leqslant \alpha'(G) \leqslant \left\lfloor\frac{n}{2}\right\rfloor$, последнее неравенство следует из того, что большему числу рёбер в паросочетании не хватило бы всех вершин графа. Более хитрой является оценка Эрдёша–Посы: $\alpha'(G) \geqslant \min\left\{\left\lfloor\frac{n}{2}\right\rfloor, \delta\right\}$. Таким образом, проще всего случай, когда $\delta \geqslant \left\lfloor\frac{n}{2}\right\rfloor$, ведь в этом случае у любого графа из $\mathcal{G}(n, \Delta, \delta)$ максимальное паросочетание будет размера ровно $\left\lfloor\frac{n}{2}\right\rfloor$. Чтобы сделать какой-то граф из $\mathcal{G}(n, \Delta, \delta)$, можно сделать δ-регулярный или $(\delta-1)$-регулярный граф на n вершинах, после чего если он $\delta-1$-регулярный, добавить в него паросочетание, не покрывающее максимум одну вершину. Теперь надо выбрать эту вершину (а если граф уже δ-регулярный, то произвольную вершину) и добавить исходящих из неё рёбер, чтобы её степень стала Δ.

Здесь и дальше мы будем считать, что $\delta < \left\lfloor\frac{n}{2}\right\rfloor$. В таком случае $\min\left\{\left\lfloor\frac{n}{2}\right\rfloor, \delta\right\} = \delta$, и оценка Эрдёша-Посы тоже бывает точна — а именно, при $n < \Delta + \delta$. В таком случае минимальный возможный размер $\alpha'(G)$ равен δ, и он достигается на графе $K_{\delta, n-\delta}$, в который к вершине из доли размера δ добавили рёбра в её собственную долю, чтобы она стала степени Δ (нетрудно посчитать, что потребуется $\Delta+\delta-n$ рёбер, и именно здесь мы пользуемся тем, что $n < \Delta+\delta$).

За остальную и существенно более сложную часть ответственны Боллобаш и Элдридж, ведь там очевидная оценка Эрдёша-Посы недостаточно точна, она может быть улучшена. Здесь и далее мы считаем, что в придачу к $\delta < \left\lfloor\frac{n}{2}\right\rfloor$ верно ещё и $n \geqslant \Delta + \delta$. Начнём со случая, когда Δ и δ сильно различаются. Если $\Delta - \delta \geqslant 2$, то ответ на задачу равен $\left\lceil\frac{n\delta}{\Delta+\delta}\right\rceil$. Обозначим это число через s, тогда пример можно получить так: возьмём две доли X размером s и Y размером $n-s$ и соединим каждую вершину доли Y с δ вершинами доли X, чтобы в доле X степени всех вершин отличались не больше чем на единицу (этого можно добиться, например, расставив вершины доли X по кругу и, перебирая вершины из доли Y, каждой из них отсчитывать δ вершин X, идя по этому кругу, причём не останавливаясь на первом проходе по кругу, а идя дальше, и останавливаясь, лишь когда все вершины из Y получат по δ рёбер), а потом из одной из вершин $x \in X$ добавим ещё рёбер в вершины Y, пока она не станет степени Δ.

Дальше идут три похожих случая.

1. δ чётно и равно Δ. Тогда ответ $\left\lceil\frac{n\delta}{2(\delta+1)}\right\rceil$. Пусть $n = b(\delta+1) + c$, где b и c натуральные, $\delta+1 \leqslant c < 2(\delta+1)$ (такое представление легко получить с помощью деления с остатком). Тогда экстремальный граф получается объединением b копий графа $K_{\delta+1}$ и одной копии R_c^δ (напомним, что это δ-регулярный граф на δ вершинах, который существует, так как $c > \delta$ и δ чётно).

2. δ нечётно и равно $\Delta - 1$. Тогда ответ $\left\lceil\frac{n\Delta}{2(\Delta+1)}\right\rceil$. Представим $n = b(\Delta+1) + c$, где b и c натуральные, $\Delta+1 \leqslant c < 2(\Delta+1)$, тогда экстремальный граф получается объединением b копий $K_{\Delta+1}$ и одного графа R_c^Δ, из которого удалили ребро (как раз за счёт удалённого ребра и образуется две вершины степени δ, которых бы иначе не было).

3. δ чётно и равно $\Delta - 1$. Тогда ответ $\left\lceil \frac{n\delta+1}{2(\delta+1)} \right\rceil$. Представим $n = b(\Delta+1) + c$, где b и c натуральные, $\delta+1 < c \leqslant 2(\delta+1)$, тогда экстремальный граф получается объединением b копий $K_{\delta+1}$ и одного графа R_c^δ, к которому добавили ребро (как раз за счёт добавленного ребра и образуется две вершины степени Δ, которых бы иначе не было).

Внимательный читатель заметит, какой случай мы до сих пор не рассмотрели — нечётное $\delta = \Delta < \left\lfloor \frac{n}{2} \right\rfloor$. Другими словами, это нечётно-регулярный граф при чётном количестве вершин n. При $\delta = 1$ надо просто взять совершенное паросочетание, и никуда не деться — ответ $\frac{n}{2}$. Но при нечётном $\delta \geqslant 3$ эта задача оказывается *шокирующе* сложной, гораздо более навороченной, чем всё, что мы рассмотрели ранее.

Начнём с ответа. Представим n в виде $u(\delta+1)^2 + (2k+1)(\delta+2) + r$, где u, k, r — целые неотрицательные числа, $2k < \delta$, $r \leqslant 2\delta + 3$. Тогда ответом является $\frac{n-u(\delta-1)}{2} - k$. Несмотря на то, что тройка (u, k, r) определена не однозначно, для всех корректных троек величина $\frac{n-u(\delta-1)}{2} - k$ равна одному и тому же!

Попробуем построить нужный граф. Для начала определим граф $L_{p,d}^\delta$ на $p+d$ вершинах, в котором d вершин степени $\delta-1$ и $p-d$ вершин степени δ (здесь p, d — нечётные натуральные числа, $p > d$). Этот граф можно получить из нашей конструкции $R_p^{\delta-1}$, если дополнительно провести $\frac{p-d}{2}$ «почти главных» диагоналей, которые соединяют вершину i с вершиной $i + \frac{p-1}{2}$.

Граф M состоит из δ копий $L_{\delta+2,1}^\delta$ и одной вершины x. В каждой из копий ровно по одной вершине степени $\delta-1$. Соединим их всех с x, получится δ-регулярный граф — это и есть M.

Ещё страшнее граф N — он состоит из $L_{\delta+1+r,\delta-2k}^\delta$ (нетрудно доказать, что r нечётно), ещё $2k$ копий графа $L_{\delta+2,1}^\delta$ и, наконец, вершины y. Как и в прошлом абзаце, здесь все вершины имеют степень δ, кроме изолированной вершины y и δ вершин степени $\delta-1$ в графах $L_{\dots}^\delta$. Если соединим y с такими вершинами, получится δ-регулярный граф N.

Наконец, наш граф G на задачу состоит из графа N и u копий графа M. Понятное дело, он δ-регулярен, так как M и N δ-регулярны. Немного сложнее понять, что в нём n вершин, и ещё сложнее понять, что $\alpha'(G) = \frac{n-u(\delta-1)}{2} - k$. Но и это возможно сделать! Предоставляем это заинтересованным читателям.

Разбор задачи «Здесь был Вася»

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

Эта задача содержит некоторый простор как для творчества, так и для частичных решений. Расскажем схематично одно из решений, которое должно набирать полный балл.

Прежде всего нужно рисовать широко: ведь любую полосу из 10 соседних столбцов может закрасить дворник, и если весь наш рисунок был на ней, тогда во втором запуске мы вообще ничего не увидим. Далее — нужно уметь использовать хотя бы два цвета: если мы будем использовать только один цвет, это никак нам не поможет оставить свой след на стене такого же цвета. Наконец, нужно использовать свои краски по максимуму: если мы поменяем всего несколько клеток, то при некотором невезении может так получиться, что Вася просто закрасит их все.

Самое простое, что можно сделать — завести массив строковых констант, который будет содержать наш знак, всегда один и тот же. В знаке следует использовать максимальное количество клеток каждого из трёх доступных цветов. Чтобы большая часть нашего знака не совпала с тем, что было на стене до нас, раскидаем закрашиваемые клетки более-менее случайно.

Как проверять, что наш знак присутствует на стене? Попробуем наложить его на стену всеми способами — со всеми возможными сдвигами из-за перевешивания лампы. Посчитаем количество совпадений (какие клетки содержат то же, что и наш знак) и количество несовпадений (какие клетки в нашем знаке есть, а на стене имеют другие цвета). Отдельно посмотрим, где в несовпадениях белые клетки, и оценим, могут ли они все быть результатом

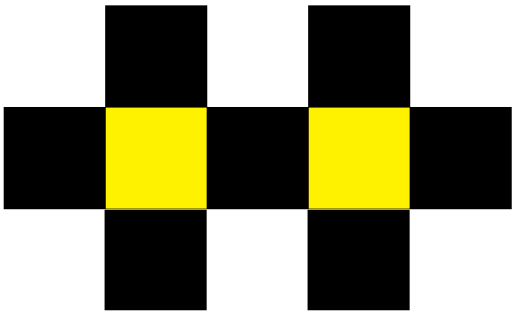
действий дворника (то есть образуют ли они непрерывную полосу, которая имеет ширину ровно в 10 клеток — или меньше, если она расположена у границы). Если белые клетки мог сделать дворник, совпадений много, а несовпадений (из-за Васи) мало — согласимся, что на этой стене мы уже рисовали. Если нет — нарисуем свой знак в точности так, как он задан в нашем массиве.

Разбор задачи «Связная фигура»

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

Давайте пока попытаемся решить немного другую задачу: какое *наибольшее* количество *чёрных* клеток может быть в связной фигуре с k *белыми* клетками?

Ответ — $3k + 1$. Пример придумать несложно — для $k = 0$ нужно просто взять одну чёрную клетку, а для больших k нужно взять длинную полоску с чёрными «шипами». Пример для $k = 2$ показан ниже на рисунке (для наглядности картинки закрашены только клетки, принадлежащие фигуре, причём белые клетки фигуры закрашены жёлтым цветом, чтобы отличаться от белого фона):



Возможный пример для $k = 2$

Чтобы понять, что больше $3k + 1$ чёрных клеток быть не может, достаточно просто порисовать картинки. Неформально, только у первой белой клетки может быть четыре «уникальных» чёрных соседа, все остальные же белые клетки «приклеиваются» к уже имеющейся картинке с помощью одной уже учтённой чёрной клетки.

Теперь вспомним, что нам нужно решить другую задачу. Пусть k — ответ на задачу. Тогда, как мы поняли выше, $n \leqslant 3k+1$ (иначе белых клеток слишком много). С другой стороны, наименьшее такое k, что $3k + 1 \geqslant n$, точно подойдёт: нужно просто взять картинку с k белыми и $3k+1$ чёрными клетками и «общипать» у неё часть шипов. Таким образом, ответ на задачу — наименьшее такое k, что $3k + 1 \geqslant n$. Другими словами, это $\left\lceil \frac{n-1}{3} \right\rceil = \left\lfloor \frac{(n-1)+2}{3} \right\rfloor = \left\lfloor \frac{n+1}{3} \right\rfloor$. Ответ следует считать с помощью целочисленного деления; использовать вещественное округление не рекомендуется, так как в последней подгруппе стандартного 64-битного вещественного типа не хватает, чтобы точно сохранить значения ответа.

Разбор задачи «Тропический минимум»

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

Рассмотрим подотрезок массива с минимальной суммой. Покажем, что его сумма и является ответом на задачу. Действительно, результат любой расстановки арифметических операций — сумма некоторого отрезка, следовательно, любая расстановка знаков дает результат не меньше наименьшей суммы подотрезка, с другой стороны, такую сумму легко получить, тропически перемножив элементы на отрезке и сложив их со всем остальным.

Как найти отрезок с минимальной суммой? Посчитаем префикс-суммы для нашего массива. Сумма любого отрезка представляется как разность двух префикс-сумм, соответствующих концу и началу отрезка. Переберем конец отрезка и для каждого конца найдем начало с максимальным значением префикс-суммы. Это можно сделать за константное время, если предварительно вычислить префикс-максимумы для префикс-сумм.

Итоговое время работы $O(n)$.

Разбор задачи «Тропический максимум»

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

Сделаем двоичный поиск по величине ответа. Теперь наша задача сводится к проверке того, можно ли расставить арифметические операции так, чтобы ответ был не меньше некоторого числа t. Эта задача эквивалентна задаче разбиения массива на подотрезки с суммой не больше t. Теперь решим эту задачу.

Вычислим префикс-суммы для нашего массива. Будем решать задачу методом динамического программирования. Пусть $\mathrm{dp}[i]$ — ответ на задачу для префикса длины i. Пусть мы вычислили $\mathrm{dp}[i]$ для всех $i < r$, тогда $\mathrm{dp}[r] = \texttt{true}$ тогда и только тогда, когда существует такое $l < r$, что $\mathrm{dp}[l] = \texttt{true}$ и сумма чисел на отрезке с l по r не превосходит t. Заметим, что среди всех возможных l нас интересуют только те, для которых $\mathrm{dp}[l] = \texttt{true}$, а среди таких нас интересует l с наибольшей префикс-суммой. Такое l и будем поддерживать. Таким образом, вычислить $\mathrm{dp}[r]$ можно за $O(1)$ при помощи сравнения разности префикс-сумм для r и l и числа t. Следовательно, решить задачу для заданного t можно за $O(n)$.

Итоговое время работы $O(n \log n)$.

Разбор задачи «Раздача подарков»

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

Пусть c_n — количество счастливых $n \times n$ матриц. Для $n = 1$ счастливой является только матрица $\begin{pmatrix}1\end{pmatrix}$, поэтому $c_1 = 1$. Давайте научимся выражать c_n через c_{n-1} для $n \geqslant 2$.

Если кратко, то идея такая: есть $2^n - 1$ способов выбрать первую строку, 2^{n-1} дозаполнить столбец, соответствующий взятому первым зайчонком предмету, а после удаления первой строчки и вышеупомянутого столбца матрица должна всё ещё остаться счастливой, но уже $(n-1) \times (n-1)$. Следовательно, $c_n = (2^n - 1) \cdot 2^{n-1} \cdot c_{n-1}$.

Давайте объясним предыдущий абзац более подробно.

Какой может быть первая строка счастливой матрицы? Почти любой — главное, чтобы она не состояла полностью из нулей. Следовательно, есть $2^n - 1$ способов выбрать первую строку счастливой матрицы.

Далее, пусть первый зайчонок забрал k-й подарок (то есть первый, второй, ..., $(k-1)$-й подарки ему не нравятся, а k-й — нравится). Тогда k-й подарок не участвует в дальнейшем ходе алгоритма, и действия второго, третьего, ..., n-го зайчат не зависят от того, нравится ли им k-й подарок или нет. Следовательно, есть ещё 2^{n-1} способ выбрать то, как второй, третий, ..., n-й зайчата относятся к k-му подарку.

Что же происходит дальше? Второй третий, ..., n-й зайчата рассматривают все подарки, кроме k-го, в порядке возрастания номеров. Каждый из них забирает первый, который ему понравился и не был взят раньше. Так как исходная матрица — счастливая, то они все должны получить по нравящемуся им подарку. Другими словами, матрица, полученная из исходной удалением первой строки и k-го столбца, тоже должна быть счастливой! Следовательно, есть ровно c_{n-1} способов выбрать все оставшиеся элементы матрицы.

Несложно заметить, что все рассуждения выше проходят и в обратную сторону. То есть для любого способа выбрать первую строку, k-й столбец и счастливую $(n-1)\times(n-1)$ матрицу соответствующая им $n \times n$ матрица тоже будет счастливой.

Следовательно, $c_n = (2^n - 1) \cdot 2^{n-1} \cdot c_{n-1}$. Из этой формулы уже получается решение, работающее за время $O(n)$, если заранее посчитать значения степеней двойки по модулю $10^9 + 7$.