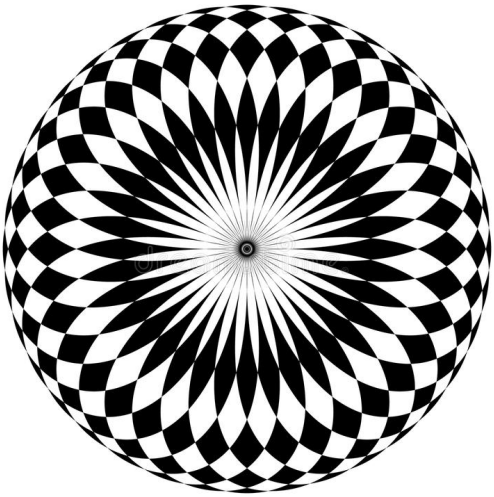
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
КАЗАНСКИЙ ФЕДЕРАЛЬНЫЙ (ПОВОЛЖСКИЙ) УНИВЕРСИТЕТ
РЕСПУБЛИКАНСКИЙ ОЛИМПИАДНЫЙ ЦЕНТР РТ

Олимпиады по информатике школьников Татарстана

2021-2022 и 2022-2023



Казань – 2024

УДК 372.800.4
ББК 74.263.2

Печатается по решению учебно-методической комиссии
Института математики и механики КФУ
им. Н.И. Лобачевского

Киндер М.И.

Олимпиады по информатике школьников Татарстана. 2021-2022 и 2022-2023: Учебно-методическое пособие / М.И. Киндер. — Казань: Казанский федеральный университет, 2024. — 134 с.

Брошюра предназначена для школьников, учителей, преподавателей и тренеров по олимпиадной информатике. В ней представлены задачи, предлагавшиеся в 2021-2022 и 2022-2023 учебных годах на муниципальном и региональном этапах Всероссийской олимпиады школьников Татарстана по информатике. Для каждой задачи приведены идея решения, система оценки и описание конкретной реализации на одном из языков, принятом на олимпиадах по спортивному программированию.

Введение

Муниципальный и региональный этапы Всероссийской олимпиады школьников по информатике ежегодно проводятся во всех субъектах России. Задачи для муниципального этапа разрабатываются предметно-методической комиссией, состоящей из преподавателей и студентов Казанского федерального университета. Возглавляет рабочую группу — председатель жюри муниципального и регионального этапов, доцент КФУ М. И. Киндер. В состав жюри входят также тренеры и учителя информатики лицеев г. Казани, которые оказывают огромную помощь в организации и проведении олимпиадных соревнований в Республике Татарстан.

Особая благодарность учителю информатики лицея-интерната № 2 г. Казани И. И. Сафиуллину, бессменному администратору проверяющей системы соревнования.

Представленные в книге задачи предназначены для учащихся 7-11 классов, а также для учителей и тренеров, работающих с талантливыми школьниками в области информатики.

2021-2022 учебный год

Муниципальный этап 34-й Всероссийской олимпиады по информатике среди школьников Республики Татарстан состоялся 18 ноября 2021 г. В олимпиаде приняли участие почти 400 школьников 7-8 классов и 800 школьников 9-11 классов. Школьники младших и старших классов решают соответственно четыре или пять задач. Как правило, первые две задачи опираются, в основном, на логику алгоритмического мышления и не требуют «профессиональных» олимпиадных навыков. Следующие две или три задачи в списке заданий — более сложные, они охватывают достаточно большой спектр различных стандартных и нестандартных алгоритмов.

Региональный этап олимпиады проходил с 15 по 17 января 2022 г. Кроме школьников 9-11 классов — традиционных участников регионального этапа — на олимпиаду были приглашены также ученики 7-8 классов, показавшие хорошие результаты на муниципальном этапе Всероссийской олимпиады. Для школьников 7-8 классов жюри подготовило несколько задач, сложность которых, по мнению жюри, соответствовала возрастной категории участников.

Материалы муниципального и регионального этапа Всероссийской олимпиады по информатике среди школьников Республики Татарстан (результаты участников, архив жюри с тестами, генераторами тестов и решениями жюри) доступны в интернете по адресу:

http://kpfu.ru/math/olimpiady-dlya-shkolnikov-i-studentov/olimpiady-shkolnikov-po-informatike

Ниже представлены условия и подробные решения задач всех перечисленных олимпиад. Для каждой задачи приведена идея решения, система оценки и описание конкретной реализации на языке `C++` или `Python`. Для большинства олимпиадных заданий указаны фамилии авторов идеи и фамилии тех, кто готовил эти решения.

Муниципальный этап, 2021-2022

Задача 1. Розы (7-8 классы)

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`256 мегабайт`

Известный садовод Гладиолус Ромашкин выращивает розы у себя в теплице, а потом продаёт их в городе на цветочном рынке. Перед походом на рынок Ромашкин пересчитал лепестки у каждой розы и решил, что розы с k лепестками он будет продавать за r рублей. Если лепестков у розы меньше k, то отсутствие каждого из лепестков уменьшит цену на 1 рубль. Если же лепестков больше k, то такую розу Ромашкин будет продавать в 2 раза дороже, то есть за $2r$ рублей. К концу дня Гладиолус Ромашкин продал все n роз и хочет подсчитать свою выручку.

Ваша задача — помочь ему в этом.

Формат входных данных

Первая строка содержит три целых числа k, r, n ($1 \leqslant k \leqslant 100$; $k \leqslant r \leqslant 10^5$; $1 \leqslant n \leqslant 10^5$). Во второй строке записано n целых чисел a_i — количество лепестков каждой из роз ($1 \leqslant a_i \leqslant 1000$).

Формат выходных данных

Запишите одно число — выручку Гладиолуса Ромашкина.

Пример

стандартный ввод	стандартный вывод
10 20 3 10 5 15	75

Пояснение к примеру

В примере Ромашкин сможет продать первую розу за 20 рублей, вторую — за 15 рублей, а третью — за $2 \cdot 20 = 40$ рублей. Общая выручка составит $20 + 15 + 40 = 75$ рублей.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	20	$1 \leqslant n \leqslant 3$	
2	25	$1 \leqslant n \leqslant 10$	1
3	25	$1 \leqslant n \leqslant 100$	1, 2
4	30	$1 \leqslant n \leqslant 10^5$	1, 2, 3

Задача 2. До последней стружки (7-11 классы)

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`256 мегабайт`

В токарном цехе ремонтного завода вытачиваются втулки из бронзовых заготовок. На изготовление каждой бронзовой втулки требуется по одной заготовке. После изготовления втулок остаётся бронзовая стружка, которая идёт на переплавку и литьё новых заготовок. Для выплавки одной дополнительной заготовки хватит стружки, оставшейся после обработки m заготовок.

Вам необходимо составить программу для вычисления количества заготовок, необходимого для получения n бронзовых втулок.

Формат входных данных

В единственной строке записаны два целых числа n и m ($2 \leqslant$ $\leqslant n \leqslant 10^{18}$; $2 \leqslant m \leqslant 10^{18}$).

Формат выходных данных

Запишите одно число — количество заготовок, необходимое для изготовления n бронзовых втулок.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	20	$n, m \leqslant 10$	
2	40	$n, m \leqslant 5 \cdot 10^6$	1
3	20	$n, m \leqslant 10^9$	1, 2
4	20	$n, m \leqslant 10^{18}$	1, 2, 3

Примеры

стандартный ввод	стандартный вывод
7 3	5

Пояснение к примеру

Из 5 бронзовых заготовок можно получить 5 втулок, а из оставшихся стружек сделать ещё одну заготовку и 2 единицы стружки. Из этой заготовки получается ещё одна втулка, причём образуется ещё одна единица стружки, то есть всего будет $2 + 1 = 3$ единицы стружки. Из них можно изготовить ещё одну втулку. Таким образом, получится $5 + 1 + 1 = 7$ втулок.

Задача 3. Произведение цифр (7-11 классы)

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`256 мегабайт`

Петя задумал натуральное число x, в десятичной записи которого все цифры *разные*. Затем он перемножил цифры, и полученное произведение n сообщил Васе. (Если в числе x только одна цифра, произведение считается равным этой цифре.) Однако этого оказалось недостаточно для определения Петиного числа. Тогда Петя добавил, что число x — наибольшее среди всех чисел с заданным произведением цифр.

Вам необходимо найти число x.

Формат входных данных

В единственной строке записано число n – произведение цифр задуманного числа ($1 \leqslant n \leqslant 10^9$).

Формат выходных данных

Запишите задуманное Петей число x. Если такого числа нет, выведите `-1`.

Примеры

стандартный ввод	стандартный вывод
4	41
44	-1

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	10	$n \leqslant 20$	
2	15	$n \leqslant 100$	1
3	20	$n \leqslant 10^6$, n – нечётное	1
4	20	$n \leqslant 10^9$, n – нечётное	1, 3
5	35	$n \leqslant 10^9$	1, 2, 3, 4

Задача 4. Ковбой Джонни (7-11 классы)

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`256 мегабайт`

Далеко-далеко на лугу пасутся ко... Нет, не кони и не козы, а коровы. Юный ковбой Джонни обожает молоко и хочет пронумеровать все n своих коров. Поскольку он умеет считать только до числа m, каждая корова получает номер в виде целого числа от 1 до m, номера коров могут повторяться. Построив их в ряд, ковбой Джонни раздаёт каждой корове номер в соответствии с его придуманным правилом:

- для каждого i номер коровы, стоящей на i-м месте, должен быть меньше номеров коров, стоящих на местах $i+2$ и $i+3$. (Если нет коровы на месте $i+2$, то есть $i+2 > n$, нет и этого ограничения на номер коровы на i-м месте. То же самое и для коровы на месте $i+3$.)

Теперь Джонни захотелось узнать, сколько способов такой нумерации можно придумать.

Формат входных данных

В первой строке записано одно целое положительное число T – количество вопросов Джонни ($1 \leqslant T \leqslant 10$). В каждой из следующих T строк записано по два целых числа n и m, где n — число коров ($3 \leqslant n \leqslant 10^3$), а m — максимально возможный номер каждой коровы ($1 \leqslant m \leqslant 10^3$).

Формат выходных данных

Для каждого вопроса запишите количество способов нумерации в виде остатка от деления на $10^6 + 7$, по одному на строке.

Пример

стандартный ввод	стандартный вывод
2 4 2 4 3	1 8

Пояснение к примеру

В примере для $n = 4$ и $m = 2$ есть только один способ нумерации — 1122.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. В подзадаче 4 символ $\lceil x \rceil$ означает округление числа x до ближайшего целого в бóльшую сторону, то есть наименьшее целое, бóльшее или равное x.

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	10	$3 \leqslant n \leqslant 10, \quad 1 \leqslant m \leqslant 10$	
2	20	$3 \leqslant n \leqslant 50, \quad 1 \leqslant m \leqslant 50$	1
3	30	$3 \leqslant n \leqslant 200, \ 1 \leqslant m \leqslant 200$	1, 2
4	10	$3 \leqslant n \leqslant 1000, \ \ m = \lceil n/2 \rceil$	
5	30	$3 \leqslant n \leqslant 1000, 1 \leqslant m \leqslant 1000$	1, 2, 3, 4

Задача 5. Леон и Ронни (9-11 классы)

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`256 мегабайт`

На n карточках написаны различные натуральные числа a_1, $a_2, \ldots, a_n$ (на каждой по одному). За один ход игрок забирает себе ровно одну из карточек. Леон и Ронни играют очень рассеянно и часто забывают, чья очередь хода. Поэтому неудивительно, что кто-то из них в некоторые моменты игры мог сделать несколько ходов подряд.

После каждого хода вычисляется сумма всех чисел на карточках каждого из игроков. Если сумма чисел на карточках у Леона оказывается больше, чем у Ронни, записывается символ `L`, иначе — `R`. Например, если в начале игры Леон забрал карточку с числом 2, затем Ронни — карточку с числом 1, и, наконец, снова Ронни забрал карточку с числом 4, то для этой последовательности ходов записывают слово `LLR`.

Вам необходимо составить программу, которая по заданному набору из n чисел и слову длины n восстанавливает ход игры, то есть определяет, кто из игроков делал очередной ход и какие карточки брали игроки.

Формат входных данных

В первой строке записано целое число n ($1 \leqslant n \leqslant 10^5$) — количество карточек. Вторая строка содержит n различных целых чисел a_1, a_2, …, a_n ($1 \leqslant a_i \leqslant 10^9$, $1 \leqslant i \leqslant n$) — числа на карточках. В третьей строке записано слово длины n — последовательность (без пробелов) из n символов `L` и `R`.

Формат выходных данных

Выведите последовательность из n строк. В каждой строке запишите число на очередной карточке, затем через пробел символ `L` или `R`, обозначающий, кто из игроков взял указанную карточку. Если решений несколько, выведите любое из них. Если требуемое слово получить невозможно, запишите `-1`.

Примеры

стандартный ввод	стандартный вывод
2 4 3 LR	3 L 4 R
3 7 1 4 LRR	4 L 7 R 1 R

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	10	$1 \leqslant n \leqslant 2$	
2	20	$1 \leqslant n \leqslant 10$	1
3	30	$1 \leqslant n \leqslant 10^3$	1, 2
4	40	$1 \leqslant n \leqslant 10^5$	1, 2, 3

Региональный этап, 2021-2022

Региональный этап Всероссийской олимпиады школьников состоялся 15-17 января 2022 года. Соревнование проходило в два тура с перерывом (16 января) между ними на день отдыха. На каждый тур олимпиады школьникам предлагалось 4 задачи.

Ниже представлены тексты и разбор решения этих задач, которые подготовили Денис Акилов, Николай Будин, Савелий Григорьев, Михаил Первеев, Андрей Станкевич, Григорий Хлытин.

Общая информация о проверке

Во всех задачах баллы за подзадачу начисляются только, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Информация о тестировании приведена в таблице.

Тип информации	Пояснение
Полная	Для каждого теста подзадачи сообщается результат работы программы участника на этом тесте.
Первая ошибка	Для подзадачи сообщается одно из двух: • если все тесты пройдены, то сообщаются баллы за подзадачу; • если хотя бы один тест не пройден, сообщается номер первого не прошедшего теста и результат работы программы участника на этом тесте.
Баллы	Для подзадачи сообщаются баллы за эту подзадачу.

Задача 1. Чемпионат по устному счету

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`512 мегабайт`

Председатель жюри чемпионата по устному счету Николай Константинович Минусов придумал новое задание для участников чемпионата. Исходно на доске выписывается n целых чисел $a_1, a_2, \ldots, a_n$. После этого участник должен выполнять команды двух типов:

1. Стереть i-е число с доски и записать вместо него число x. Другими словами, если на доске были записаны числа $a_1, a_2, \ldots, a_n$, то после выполнения этой команды числа будут равны: $a_1, \ldots, a_{i-1}, x, a_{i+1}, \ldots, a_n$.
2. Циклически сдвинуть последовательность чисел на k вправо. Другими словами, если на доске были записаны числа $a_1, a_2, \ldots, a_n$, то после выполнения этой команды числа будут равны: $a_{n-k+1}, a_{n-k+2}, \ldots, a_n, a_1, a_2, \ldots, a_{n-k}$.

После выполнения каждой команды участник должен вычислить сумму всех чисел, записанных на доске, и сообщить ее жюри. Чтобы подготовиться проверять ответы участников, членам жюри необходимо самим вычислить требуемые суммы.

Формат входных данных

В первой строке записано целое число n — количество чисел, изначально записанных на доске ($2 \leqslant n \leqslant 10^5$).

Во второй строке через пробел записаны n целых чисел a_1, a_2, $\ldots$, a_n – числа, изначально выписанные на доске ($-10^9 \leqslant a_i \leqslant 10^9$).

В третьей строке записано целое число q — количество команд, которые необходимо выполнить ($1 \leqslant q \leqslant 10^5$).

В каждой из следующих q строк записана очередная команда в следующем формате:

- $1\ i\ x$ – это означает, что участник должен заменить i-е число последовательности на число x ($1 \leqslant i \leqslant n$; $-10^9 \leqslant x \leqslant 10^9$).
- $2\ k$ – это означает, что участник должен циклически сдвинуть последовательность чисел на k вправо ($1 \leqslant k < n$).

Формат выходных данных

В качестве ответа выведите q строк, в каждой из которых записано одно целое число.

В i-й строке должна быть записана сумма чисел на доске после выполнения первых i команд.

Обратите внимание, что ответ может быть достаточно большим и для его хранения потребуется 64-битный тип данных — `int64` в паскале, `long long` в `C++`, `long` в `Java`.

Примеры

стандартный ввод	стандартный вывод
6 4 1 2 1 5 3 5 2 3 1 3 10 1 4 4 2 1 1 1 -10	16 23 23 23 11
3 1000000000 1000000000 1000000000 3 1 2 999999999 2 2 1 2 999999999	2999999999 2999999999 2999999998

Пояснение к примеру

Рассмотрим первый пример из условия. Изначально последовательность записанных на доске чисел равна: $4, 1, 2, 1, 5, 3$.

После первой команды последовательность циклически сдвигается на 3 элемента вправо. Новая последовательность: 1, 5, 3, 4, 1, 2. Сумма чисел равна: $1 + 5 + 3 + 4 + 1 + 2 = 16$.

После второй команды необходимо заменить третий элемент последовательности на число 10. Новая последовательность: 1, 5, 10, 4, 1, 2. Сумма чисел равна: $1 + 5 + 10 + 4 + 1 + 2 = 23$.

После третьей команды заменить четвертый элемент на число

4. Так как четвертый элемент уже равен 4, последовательность не изменяется. Сумма чисел также равна 23.

После четвертой команды последовательность циклически сдвигается на 1: $2, 1, 5, 10, 4, 1$. Сумма чисел не изменилась.

Наконец, после пятой команды последовательность становится равна: $-10, 1, 5, 10, 4, 1$. Сумма чисел в итоговой последовательности равна $-10 + 1 + 5 + 10 + 4 + 1 = 11$.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Для подзадач 1–3 сообщается «Полная» информация, для подзадачи 4 – «Первая ошибка».

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	22	$2 \leqslant n \leqslant 1000$, только команды 1-го типа	
2	17	$2 \leqslant n \leqslant 1000$, во всех командах 2-го типа $k = 1$	
3	23	$2 \leqslant n \leqslant 1000$	1, 2
4	38		1, 2, 3

Задача 2. Прыгающий робот

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`512 мегабайт`

Компания «Flatland Dynamics» разрабатывает прыгающего робота. Для испытания робота используется полигон, на котором организован круговой маршрут из n специальных платформ, пронумерованных от 1 до n. Расстояние между i-й и $(i+1)$-й платформой равно d_i, аналогично расстояние между n-й и первой платформой равно d_n.

Робот оснащен искусственным интеллектом и в процессе испытания учится прыгать всё дальше. В любой момент времени робот характеризуется своей *ловкостью* — целым числом a. Робот может перепрыгнуть с платформы i на платформу $i+1$, если $a \geqslant d_i$. Аналогично, прыжок с n-й платформы на первую возможен, если $a \geqslant d_n$. При этом после каждого прыжка ловкость робота увеличивается на 1.

Разработчики робота выбирают одну из платформ в качестве стартовой. Они считают эксперимент удачным, если робот может, совершив n прыжков от текущей платформы к следующей, завершить полный круг и вернуться на ту же платформу. Разработчикам необходимо выяснить, для какого минимального значения начальной ловкости робота им удастся провести эксперимент и с какой платформы роботу следует начать прыжки.

Формат входных данных

На первой строке ввода находится число n ($3 \leqslant n \leqslant 10^7$).

Вторая строка содержит одно целое число f, которое описывает формат, в котором задан массив расстояний между платформами.

Если $f = 1$, то на третьей строке находятся n целых чисел $d_1, d_2, \ldots, d_n$ ($1 \leqslant d_i \leqslant 10^9$).

Если $f = 2$, то на третьей строке находится число m ($2 \leqslant \leqslant m \leqslant \min(n, 10^5)$) и три целых числа x, y и z ($0 \leqslant x, y, z \leqslant 10^9$). На четвёртой строке находятся m целых чисел $c_1, c_2, \ldots, c_m$ ($1 \leqslant \leqslant c_i \leqslant 10^9$). Значения d_i вычисляются по следующим формулам.

Если $1 \leqslant i \leqslant m$, то $d_i = c_i$.

Если $m+1 \leqslant i \leqslant n$, то $d_i = ((x \cdot d_{i-2} + y \cdot d_{i-1} + z) \bmod 10^9) + 1$.

Здесь mod означает остаток от целочисленного деления, в языках `C++`, `Java` и `Python` он обозначается символом «`%`».

Формат выходных данных

Требуется вывести два целых числа: минимальную допустимую начальную ловкость a и номер стартовой платформы, на которую можно разместить робота, чтобы успешно провести эксперимент.

Если возможных стартовых платформ для минимальной начальной ловкости несколько, можно вывести любую из них.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Для всех подзадач сообщается информация «Первая ошибка».

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	15	$n \leqslant 300$, $f = 1$, $d_i \leqslant 300$	
2	17	$n \leqslant 500$, $f = 1$	1
3	10	$n \leqslant 100\,000$, $f = 1$; начать с 1-й платформы	
4	20	$n \leqslant 10^5$, $f = 1$	1, 2, 3
5	5	$f = 2$; начать с 1-й платформы	4
6	33	$f = 2$	1, 2, 3, 4, 5

Пример

стандартный ввод	стандартный вывод
5 1 3 7 4 2 5	4 3
10 2 5 1 2 3 1 2 3 4 5	653 1

Пояснение к примеру

Во втором примере массив расстояний между платформами равен $[1, 2, 3, 4, 5, 18, 45, 112, 273, 662]$. Значения от d_6 до d_{10} вычисляются по формулам:

$$
\begin{aligned}
d_6 &= ((1 \cdot d_4 + 2 \cdot d_5 + 3) \bmod 10^9) + 1 = \\
&= ((1 \cdot 4 + 2 \cdot 5 + 3) \bmod 10^9) + 1 = 18; \\
d_7 &= ((1 \cdot d_5 + 2 \cdot d_6 + 3) \bmod 10^9) + 1 = \\
&= ((1 \cdot 5 + 2 \cdot 18 + 3) \bmod 10^9) + 1 = 45; \\
d_8 &= ((1 \cdot d_6 + 2 \cdot d_7 + 3) \bmod 10^9) + 1 = \\
&= ((1 \cdot 18 + 2 \cdot 45 + 3) \bmod 10^9) + 1 = 112; \\
d_9 &= ((1 \cdot d_7 + 2 \cdot d_8 + 3) \bmod 10^9) + 1 = \\
&= ((1 \cdot 45 + 2 \cdot 112 + 3) \bmod 10^9) + 1 = 273; \\
d_{10} &= ((1 \cdot d_8 + 2 \cdot d_9 + 3) \bmod 10^9) + 1 = \\
&= ((1 \cdot 112 + 2 \cdot 273 + 3) \bmod 10^9) + 1 = 662.
\end{aligned}
$$

Задача 3. Треугольная головоломка

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`512 мегабайт`

Головоломка состоит из n треугольников. Чтобы решить головоломку, необходимо выбрать из них четыре треугольника и собрать из них большой треугольник по следующей схеме (рис. 1).

Треугольники не должны пересекаться, в объединении они должны давать треугольник. Ровно по одному из выбранных треугольников должны находиться в углах, а один треугольник должен располагаться в центре. Треугольники лежат на столе, их можно свободно вращать и двигать, но нельзя зеркально отражать.

Требуется найти все различные наборы из четырех треугольников, из которых можно собрать большой треугольник по указанной схеме. Два набора считаются разными, если существует треугольник, входящий в один, но не входящий в другой.

Формат входных данных

В первой строке дано одно целое число t — номер теста.

В второй строке дано одно целое число n — количество треугольников в головоломке ($4 \leqslant n \leqslant 30$).

В следующих n строках дано описание треугольников. Один треугольник описывается координатами трех своих углов, данных в порядке обхода треугольника против часовой стрелки. Все координаты целые и по модулю не превышают 10^5. Гарантируется, что треугольники не являются вырожденными. В исходном расположении треугольными могут пересекаться.

Формат выходных данных

В первой строке выведите одно целое число — количество наборов из четырех треугольников, из которых можно собрать большой треугольник по указанной схеме. В следующих строках выведите наборы. Каждый набор задается номерами треугольников, которые в него входят. Треугольники внутри набора можно выводить в любом порядке. Наборы можно выводить в любом порядке.

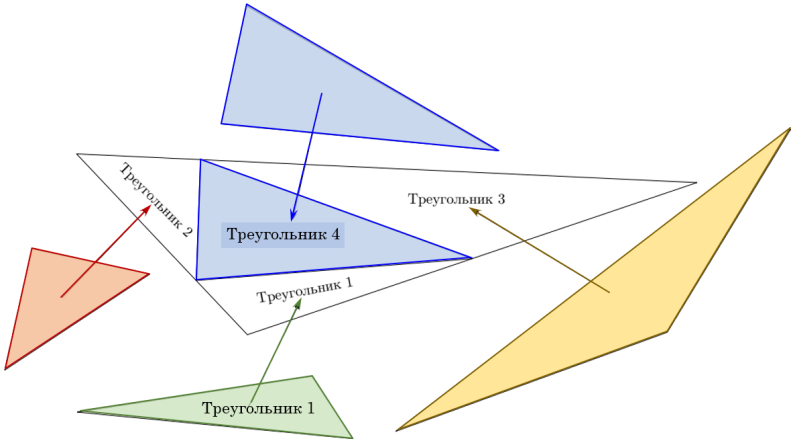


Рис. 1

Система оценивания

В этой задаче потестовая оценка. Каждый тест оценивается независимо и стоит 5 баллов. В качестве результатов проверки во время тура показывается результат на каждом тесте. Тесты удовлетворяют следующим ограничениям:

Тест	Описание теста
1	*тест из примера, не оценивается*
2	*тест из примера, не оценивается*
3	Все треугольники равны с точностью до поворота, $n \leqslant 30$
4	У каждого треугольника есть горизонтальная и вертикальная стороны, все треугольники равнобедренные, $n \leqslant 10$
5	У каждого треугольника есть горизонтальная и вертикальная стороны, все треугольники равнобедренные, $n \leqslant 30$
6	У каждого треугольника есть горизонтальная и вертикальная стороны, $n \leqslant 10$
7	У каждого треугольника есть горизонтальная и вертикальная стороны, $n \leqslant 30$
8	Все треугольники прямоугольные, $n \leqslant 10$
9	Все треугольники прямоугольные, $n \leqslant 30$
10	Для каждой четвёрки треугольников, из которой можно собрать треугольник, гарантируется, что треугольник можно собрать не вращая треугольники, $n \leqslant 10$
11	Для каждой четвёрки треугольников, из которой можно собрать треугольник, гарантируется, что треугольник можно собрать не вращая треугольники, $n \leqslant 20$
12	Для каждой четвёрки треугольников, из которой можно собрать треугольник, гарантируется, что треугольник можно собрать не вращая треугольники, $n \leqslant 30$
13	$n = 10$
14	$n = 10$
15	$n = 10$
16	$n = 20$
17	$n = 20$
18	$n = 20$
19	$n = 30$
20	$n = 30$
21	$n = 30$
22	$n = 30$

Примеры

стандартный ввод	стандартный вывод
1 4 0 0 6 2 1 2 0 0 5 0 6 3 0 0 3 1 1 3 0 0 6 3 3 6	1 1 2 3 4
2 6 0 0 1 0 1 1 0 1 0 0 1 0 -1 0 0 0 0 1 1 1 0 1 1 0 -1 0 0 -1 0 0 0 0 1 1 0 1	15 1 2 3 4 1 2 3 5 1 2 3 6 1 2 4 5 1 2 4 6 1 2 5 6 1 3 4 5 1 3 4 6 1 3 5 6 1 4 5 6 2 3 4 5 2 3 4 6 2 3 5 6 2 4 5 6 3 4 5 6

Пояснение к примерам

В первом примере из данных четырех треугольников можно собрать один. При этом треугольники не требуется вращать. Во втором примере все треугольники имеют одинаковую форму прямоугольного треугольника с длинами катетов равными 1. Из любых четырех треугольников можно собрать один.

Задача 4. Массивы-палиндромы

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`512 мегабайт`

Кай работает в лаборатории изучения массивов, где он экспериментирует с двумя массивами натуральных чисел:

$A = [a_1, a_2, \ldots, a_n]$ длины n и $B = [b_1, b_2, \ldots, b_m]$ длины m.

Эксперимент, который проводит Кай, устроен следующим образом. У каждого из массивов отбрасывается произвольный, возможно пустой, префикс, а также произвольный, возможно пустой, суффикс, таким образом, чтобы оставшиеся части массивов имели равную длину. Обозначим получившиеся массивы как A' и B', а их длину как k. Затем Кай суммирует поэлементно получившиеся массивы, итоговый массив Кай обозначает через $C = [c_1, c_2, \ldots, c_k]$.

Пусть, например, $n = 5$ и $A = [4, 3, 3, 2, 1]$, и пусть $m = 6$ и $B = [4, 1, 5, 1, 3, 2]$, от массива A отбрасывается первый и последний элемент, от массива B три первых. После этого массивы имеют вид $A' = [3, 3, 2]$, $B' = [1, 3, 2]$, результат их поэлементного суммирования $C = [4, 6, 4]$.

Задача Кая заключается в том, чтобы получать такие C, которые являются *массивами-палиндромами*, то есть массивами, у которых совпадают числа на первой и последней позиции, у которых совпадают числа на второй и предпоследней позиции, и так далее, то есть у которых для всех i совпадают числа на позициях i и $k - i + 1$.

Помогите Каю понять, какой максимальный по длине массив-палиндром он может получить в результате эксперимента.

Формат входных данных

В первой строке ввода даны два целых числа n и m — количество элементов в первом и во втором массиве, соответственно $(1 \leqslant n, m \leqslant 100\,000)$.

Во второй строке ввода даны n целых чисел a_i — массив A $(1 \leqslant a_i \leqslant 100)$.

В третьей строке ввода даны m целых чисел b_j — массив B $(1 \leqslant b_j \leqslant 100)$.

Формат выходных данных

Выведите единственное целое число — максимальное значение k, которое означает, что Кай в результате эксперимента может получить массив-палиндром длины k.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Для каждой подзадачи сообщается информация «Первая ошибка».

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	13	$n, m \leqslant 300$	
2	33	все элементы массива B одинаковые	
3	16	$n \leqslant 500,\ m \leqslant 10^5$	1
4	38	нет	1, 2, 3

Пример

стандартный ввод	стандартный вывод
5 6 4 3 3 2 1 4 1 5 1 3 2	3

Задача 5. Новый год в детском саду

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`512 мегабайт`

В детском саду готовятся к новому году, и воспитательница Яна Владимировна Курантова решила организовать детей, чтобы они подготовили украшения и отправили их Санте Клаусу для украшения своих оленей.

Дети с интересом восприняли идею и вырезали из бумаги a звёздочек и b снежинок. Теперь они планируют отправить их Санте Клаусу по почте. Им так понравились вырезанные ими украшения, что они, возможно, решат оставить себе часть. Таким образом, дети могут отправить Санте x звёздочек и y снежинок, где $0 \leqslant x \leqslant a$ и $0 \leqslant y \leqslant b$. Чтобы Санта не расстроился, дети должны отправить ему хотя бы одно украшение, то есть должно выполняться также условие $x + y > 0$.

Чтобы все олени выглядели красиво, на каждом должно оказаться одинаковое количество украшений. Известно, что у Санты n оленей, поэтому если будут отправлены x звёздочек и y снежинок, величина $x + y$ должна делиться на n.

Воспитательница заинтересовалась, сколько всего различных способов составить посылку Санте Клаусу. Два способа считаются различными, если в них отличается количество звёздочек или количество снежинок.

Формат входных данных

В одном наборе входных данных содержатся несколько тестов. Каждый тест следует решить независимо.

Первая строка входных данных содержит целое число t — количество тестов ($1 \leqslant t \leqslant 10^5$).

Следующие строки описывают тесты, по одному на строке. Описание теста состоит из трёх целых чисел n, a и b — количество оленей у Санты, количество звёздочек и количество снежинок, вырезанных детьми ($4 \leqslant n \leqslant 10^9$; $0 \leqslant a, b \leqslant 10^9$).

Формат выходных данных

Выведите t чисел. Для каждого теста выведите одно число: количество способов составить посылку для Санты Клауса.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Для каждой подзадачи сообщается информация «Первая ошибка».

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	10	$t = 1;\ \ a, b \leqslant 1000$	
2	10	$t \leqslant 1000;\ a = 0$	
3	15	$t \leqslant 1000;\ a, b < n \leqslant 1000$	
4	10	$t \leqslant 1000;\ \ a, b \leqslant 1000$	1, 3
5	15	$t = 1;\ \ n \leqslant 1000$	
6	10	$t \leqslant 1000;\ \ n \leqslant 1000$	3, 5
7	30	нет	1–6

Пример

стандартный ввод	стандартный вывод
4 4 2 2 4 4 4 6 5 5 8 13 17	1 6 5 30

Пояснение к примеру

В первом тесте у Санты 4 оленя, а дети вырезали 2 звёздочки и 2 снежинки. Здесь подходит только один набор — нужно отправить все вырезанные украшения. Во втором тесте у Санты также 4 оленя, но дети вырезали 4 звёздочки и 4 снежинки. Здесь подходит 6 наборов: 0 звёздочек и 4 снежинки, 1 звёздочка и 3 снежинки, 2 звёздочки и 2 снежинки, 3 звёздочки и 1 снежинка, 4 звёздочки и 0 снежинок, а также 4 звёздочки и 4 снежинки.

Задача 6. Сортировка дробей

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`512 мегабайт`

На доске выписано две последовательности из n различных целых чисел: $A = [a_1, a_2, \ldots, a_n]$ и $B = [b_1, b_2, \ldots, b_n]$.

Составим из них n^2 дробей вида a_i / b_j, сократим каждую дробь и отсортируем их по неубыванию.

Задано число q и q целых чисел $c_1, c_2, \ldots, c_q$. Для каждого j следует выдать c_j-ю в неубывающем порядке дробь из получившихся.

Формат входных данных

На первой строке ввода находятся числа n и q ($1 \leqslant n \leqslant 10^5$, $1 \leqslant q \leqslant 10^5$, $q \leqslant n^2$).

Дополнительно выполняется неравенство $n \cdot q \leqslant 10^5$.

На второй строке ввода находятся n различных целых чисел $a_1, a_2, \ldots, a_n$ ($1 \leqslant a_i \leqslant 10^6$).

На третьей строке ввода находятся n различных целых чисел $b_1, b_2, \ldots, b_n$ ($1 \leqslant b_i \leqslant 10^6$).

На четвертой строке ввода находятся q различных целых чисел $c_1, c_2, \ldots, c_q$ ($1 \leqslant c_i \leqslant n^2$).

Формат выходных данных

Выведите q строк. На j-й строке выведите c_j-ю по неубыванию дробь среди получившихся. Дробь p / q следует выводить в формате «`p q`», дробь должна быть несократимой.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Для каждой подзадачи сообщается информация «Первая ошибка».

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	14	$n \leqslant 50$	
2	13	$n \leqslant 500$	1
3	15	$q \leqslant 100,\ c_i \leqslant 100$	
4	21	$c_i \leqslant 10^5$	3
5	37		1–4

Примеры

стандартный ввод	стандартный вывод
4 8 3 4 1 2 2 3 4 5 1 16 2 4 5 6 10 15	1 5 2 1 1 4 2 5 1 2 1 2 4 5 3 2

Пояснение к примерам

В примере дроби исходно равны:

$$\left[\frac{3}{2}, \frac{3}{3}, \frac{3}{4}, \frac{3}{5}, \frac{4}{2}, \frac{4}{3}, \frac{4}{4}, \frac{4}{5}, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \frac{2}{2}, \frac{2}{3}, \frac{2}{4}, \frac{2}{5}\right],$$

после сокращения

$$\left[\frac{3}{2}, \frac{1}{1}, \frac{3}{4}, \frac{3}{5}, \frac{2}{1}, \frac{4}{3}, \frac{1}{1}, \frac{4}{5}, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \frac{1}{1}, \frac{2}{3}, \frac{1}{2}, \frac{2}{5}\right],$$

после сортировки

$$\left[\frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{2}{5}, \frac{1}{2}, \frac{1}{2}, \frac{3}{5}, \frac{2}{3}, \frac{3}{4}, \frac{4}{5}, \frac{1}{1}, \frac{1}{1}, \frac{1}{1}, \frac{4}{3}, \frac{3}{2}, \frac{2}{1}\right].$$

Задача 7. Оптические каналы связи

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`512 мегабайт`

Всего во Флатландии n городов, пронумерованных от 1 до n, столица Флатландии имеет номер 1. Компьютерная сеть Флатландии устроена следующим образом: в каждом городе есть один центр подключения, который может быть связан с некоторыми другими центрами с помощью проводных каналов связи. При этом между любыми двумя городами есть ровно один маршрут по каналам связи, иначе говоря, сеть представляет собой дерево. Для города i, где $i > 1$, обозначим первый город на маршруте от города i до столицы как p_i.

Запланирована модернизация сети Флатландии, в результате которой некоторые каналы связи будут заменены на более современные оптические. Оптические каналы могут быть проложены только вместо существующих проводных. Стоимость замены канала, который соединяет город i с городом p_i, равна w_i. Из-за ограничений технологии любой центр подключения может быть подключён оптическими каналами не более чем к k другим центрам.

Министерство связи Флатландии хочет составить такой план модернизации каналов, чтобы после его выполнения связность сети по оптическим каналам связи была как можно выше. Поэтому необходимо выбрать для модернизации как можно больше каналов. Но при этом стоимость модернизации желательно минимизировать, поэтому при равном количестве необходимо выбрать для модернизации каналы с минимальной суммарной стоимостью.

Помогите специалистам министерства выбрать каналы для модернизации.

Формат входных данных

В первой строке ввода находятся два целых числа n и k ($2 \leqslant n \leqslant 10^5$, $1 \leqslant k \leqslant 100$).

В следующих $n-1$ строках заданы описания каналов: $(i-1)$-я из этих строк содержит два целых числа p_i и w_i ($1 \leqslant p_i \leqslant i$, $0 \leqslant w_i \leqslant 10^9$).

Формат выходных данных

Выведите два целых числа *cnt* и *cost* — максимальное число каналов, которое удастся модернизировать, и минимальную стоимость, за которую можно модернизировать такое число каналов.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Для каждой подзадачи сообщается информация «Первая ошибка».

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	5	$n \leqslant 15$, $k = 1$, $w_i = 0$	
2	5	$n \leqslant 15$, $w_i = 0$	1
3	3	$n \leqslant 15$	1, 2
4	7	$k = 1$, $w_i = 0$	1
5	5	$k = 1$	1, 4
6	7	$k \leqslant 2$, $w_i = 0$	1, 4, 5
7	4	$k \leqslant 2$	1, 4, 5, 6
8	11	$n \leqslant 100$, $w_i = 0$	1, 2
9	4	$n \leqslant 100$	1, 2, 3, 8
10	11	$n \leqslant 2\,000$, $w_i = 0$	1, 2, 8
11	4	$n \leqslant 2\,000$	1, 2, 3, 8, 9, 10
12	20	$w_i = 0$	1, 2, 4, 6, 8, 10
13	14		1–12

Пример

стандартный ввод	стандартный вывод
8 2 1 0 1 0 1 0 2 0 2 0 2 0 1 0	4 0
8 3 1 5 1 2 1 4 2 6 2 7 2 2 1 6	6 27

Пояснение к примеру

Конфигурация сети в первом примере до и после модернизации показана на рисунке 2. Каналы, которые необходимо модернизировать, показаны жирными линиями. Максимальное число каналов, которое можно модернизировать, равно 4. Стоимость модернизации любого канала равна 0 и не показана.

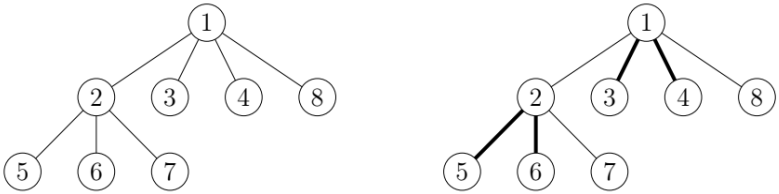


Рис. 2

Есть и другие подходящие решения, в которых модернизируются 4 канала.

Конфигурация сети во втором примере до и после модернизации показана на рисунке 3. Каналы, которые необходимо модер-

низировать, показаны жирными линиями. Максимальное число каналов, которое можно модернизировать, равно 6. Стоимость модернизации канала показана рядом с каналом, суммарная стоимость модернизации каналов в оптимальном решении равна 27.

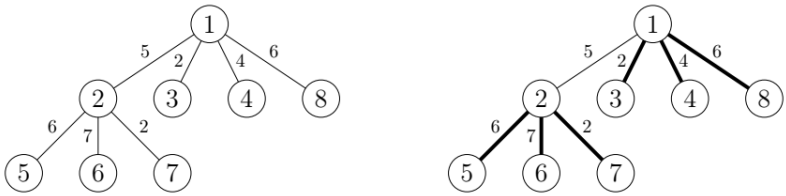


Рис. 3

Задача 8. Подарки

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`512 мегабайт`

Дед Мороз предлагает Вове выбрать подарки на Новый год.

Перед мальчиком лежат n подарков в ряд. Каждый подарок характеризуется целым числом, у i-го подарка оно равно a_i — количество удовольствия, которое подарок принесёт Вове. Удовольствие может быть как положительным, так и отрицательным, а также равным нулю.

Дед Мороз предложил Вове выбрать два числа l и r таких, что $1 \leqslant l \leqslant r \leqslant n$, и взять все подарки с номерами от l до r. Однако k подарков с максимальными характеристиками среди выбранных Вова должен отдать своей младшей сестре Маше. Остальные подарки Вова забирает себе.

Вова хочет выбрать числа l и r так, чтобы суммарное удовольствие от подарков, доставшихся именно ему, было максимальным. Общее удовольствие от набора подарков — это сумма значений a_i для подарков в наборе.

Помогите Вове выбрать числа l и r так, что $1 \leqslant l \leqslant r \leqslant n$, $r - l + 1 \geqslant k$ и общее удовольствие от выбранных подарков без учёта подарков, доставшихся Маше, максимально.

Формат входных данных

В первой строке записаны два целых числа n и k ($1 \leqslant n \leqslant 200\,000$, $0 \leqslant k \leqslant \min(100, n)$) — количество подарков перед Вовой и количество подарков, которые требуется отдать Маше.

Во второй строке заданы n целых чисел через пробел a_1, a_2, $\ldots$, a_n ($-10^9 \leqslant a_i \leqslant 10^9$) — количество удовольствия, приносимого подарками.

Формат выходных данных

Выведите единственное число — общее удовольствие от выбранных Вовой подарков без учёта тех, что достались Маше.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Для каждой подзадачи сообщается информация «Первая ошибка».

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	7	$n \leqslant 200$	
2	8	$n \leqslant 1\,000$	1
3	10	$n \leqslant 6\,000$	1, 2
4	8	$k = 0$	
5	14	$k = 1$	
6	39	$n \leqslant 80\,000$	1–3
7	14		1–6

Пример

стандартный ввод	стандартный вывод
5 0 2 -4 5 -1 7	11
5 1 2 -4 5 -1 7	4
5 2 2 -4 5 -1 7	0

Пояснение к примеру

В первом примере Вова ничего не должен отдавать Маше, поэтому он выберет $l = 3$, $r = 5$, и общее удовольствие от выбранных подарков будет равняться $5 + (-1) + 7 = 11$.

Во втором примере Вова должен будет отдать Маше подарок с самым большим количеством удовольствия. Тогда он так же выберет $l = 3$, $r = 5$, однако общее удовольствие будет равняться $5 + (-1) = 4$.

В третьем примере Вова должен отдать два подарка с наибольшими характеристиками. В таком случае одним из оптимальных вариантов будет выбрать $l = 1$, $r = 2$.

Задача 9. Действия с дробями (7-8 классы)

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`512 мегабайт`

Говорят, что обыкновенные дроби придумали греческие и индийские математики. Говорят, что ещё древние египтяне обладали искусством производить простейшие арифметические операции с дробями.

А ещё говорят, что уже тогда ученики умели решать задачу о нахождении *наименьшей* положительной дроби, при делении которой на каждую из n заданных дробей получаются целые числа.

Впрочем, говорят, эта информация не совсем точная.

И что совершенно точно — эту задачу придётся решить вам.

Формат входных данных

В первой строке записано одно целое число n — количество заданных дробей ($1 \leqslant n \leqslant 6$). В каждой из следующих n строк записано два целых числа a_i, b_i — числитель и знаменатель несократимой дроби ($1 \leqslant a_i \leqslant 10^3$, $1 \leqslant b_i \leqslant 10^{18}$).

Формат выходных данных

В единственной строке запишите через пробел два положительных целых числа — числитель и знаменатель наименьшей несократимой дроби, удовлетворяющей условию задачи.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Для каждой подзадачи сообщается «Полная» информация.

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	20	$1 \leqslant n \leqslant 2$, $1 \leqslant a_i, b_i \leqslant 10^2$	
2	20	$1 \leqslant n \leqslant 2$, $1 \leqslant b_i \leqslant 10^3$	1
3	30	$1 \leqslant n \leqslant 6$, $1 \leqslant b_i \leqslant 10^9$	1, 2
4	30	$1 \leqslant n \leqslant 6$, $1 \leqslant b_i \leqslant 10^{18}$	1, 2, 3

Пример

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

Задача 10. Гирлянда (7-8 классы)

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`512 мегабайт`

Традиции украшения новогодней ёлки насчитывают не одну сотню лет. Люди всегда старались нарядить её так, чтобы она была самой красивой героиней праздника. В ход шли разнообразные игрушки, сахарные палочки, золотая мишура. В середине 17 века для украшения ёлки стали использовать небольшие свечки, которые крепились к елочным ветвям и были похожи на современные гирлянды.

Для празднования Нового года вам нужно составить гирлянду в виде цепочки, в которую можно вкрутить n лампочек. У вас есть b синих и r красных лампочек. Требуется узнать, сколько различных разноцветных гирлянд можно собрать из них.

Искомое количество способов может быть очень велико, поэтому найдите остаток от деления этого количества способов на число $10^9 + 7$.

Формат входных данных

В первой строке записано одно целое число t ($1 \leqslant t \leqslant 10^3$) — количество наборов входных данных. Во второй строке каждого набора входных данных записаны три целых числа n, b и r ($1 \leqslant n \leqslant 10^3$; $0 \leqslant b, r \leqslant 2 \cdot 10^3$).

Формат выходных данных

Для каждого набора входных данных выведите количество различных разноцветных гирлянд, которые можно собрать, в виде остатка по модулю $10^9 + 7$.

Пример

стандартный ввод	стандартный вывод
5 1 0 1 2 1 1 2 1 3 3 3 4 3 0 0	1 2 3 8 0

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Для каждой подзадачи сообщается «Полная» информация.

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	15	$1 \leqslant n \leqslant 4$	
2	20	$1 \leqslant t \leqslant 200$, $1 \leqslant n \leqslant 10$	1
3	20	$1 \leqslant t \leqslant 10$, $1 \leqslant n \leqslant 10^3$, $b + r = n$	1, 2
4	20	$1 \leqslant t \leqslant 10^2$, $1 \leqslant n \leqslant 10^2$	1, 2
5	20	$1 \leqslant t \leqslant 10^3$, $1 \leqslant n \leqslant 10^3$	1, 2, 4

2022-2023 учебный год

Муниципальный этап 35-й Всероссийской олимпиады по информатике среди школьников Республики Татарстан состоялся 18 ноября 2021 г. В олимпиаде приняли участие почти 400 школьников 7-8 классов и 700 школьников 9-11 классов. Школьники младших и старших классов решали соответственно четыре или пять задач.

Региональный этап олимпиады проходил с 21 по 23 января 2023 г. Кроме школьников старших классов, в олимпиаде традиционно участвовали также ученики 7-8 классов, показавшие хорошие результаты на муниципальном этапе Всероссийской олимпиады. Специально для школьников 7-8 классов жюри подготовило несколько задач, сложность которых, по мнению жюри, соответствует этой возрастной категории участников олимпиады.

Материалы муниципального и регионального этапа Всероссийской олимпиады по информатике среди школьников Республики Татарстан (результаты участников, архив жюри с тестами, генераторами тестов и решениями жюри) доступны в интернете по адресу:

http://kpfu.ru/math/olimpiady-dlya-shkolnikov-i-studentov/olimpiady-shkolnikov-po-informatike

Ниже представлены условия и подробные решения задач всех перечисленных олимпиад. Для каждой задачи приведена идея решения, система оценки и описание конкретной реализации на языке `C++` или `Python`. Для большинства олимпиадных заданий указаны фамилии авторов идеи и фамилии тех, кто готовил эти решения.

Муниципальный этап, 2022-2023

Задача 1. Все на олимпиаду! (7-11 классы)

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`256 мегабайт`

Открывается сезон предметных олимпиад и для участия в физико-математической олимпиаде нужно отобрать несколько школьников из вашего класса. В классе учится n учеников, из них a учеников посещают кружок по олимпиадной математике и b учеников — кружок по олимпиадной физике. Ясно, что для участия в физико-математической олимпиаде необходимо выбирать школьников, посещающих оба кружка.

Вам необходимо по заданным числам n, a и b определить минимальное и максимальное количество школьников из класса, которые могли бы пойти на олимпиаду.

Формат входных данных

В первой строке указано одно число n — количество учеников в классе ($1 \leqslant n \leqslant 10^9$). В следующей строке записаны два числа a и b — количество учеников, которые занимаются в кружке по математике и в кружке по физике соответственно ($0 \leqslant a \leqslant n$; $0 \leqslant b \leqslant n$).

Формат выходных данных

Запишите два числа — минимальное и максимальное число школьников, которые могли бы претендовать на участие в олимпиаде.

Примеры

стандартный ввод	стандартный вывод
5 3 4	2 3
5 1 1	0 1

Пояснение к примеру

В первом примере все школьники, кроме одного, посещают кружок по физике, поэтому среди трёх школьников, *кто ходит в кружок по математике*, по крайней мере двое посещают *также и кружок по физике*; значит, наименьшее возможное число претендентов для участия в олимпиаде — 2. С другой стороны, в математическом кружке занимаются трое школьников, поэтому на олимпиаду могут отправиться не более трёх учеников. Значит, наибольшее возможное число претендентов — 3.

Система оценивания

В задаче 20 тестов, каждый тест оценивается в 5 баллов.

Задача 2. Секретное донесение (7-11 классы)

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`256 мегабайт`

Агент Джеймс Бонд собирается передать в Центр секретное сообщение в виде последовательности целых положительных чисел $a_1, a_2, \ldots, a_n$. Чтобы зашифровать послание, он образует новую последовательность чисел по правилу $b_1 = a_1$, $b_2 = a_1 + a_2$, $\ldots$, $b_n = a_1 + a_2 + \ldots + a_n$. Полученный набор чисел он снова преобразует по правилу $c_1 = b_1$, $c_2 = b_1 + b_2$, $\ldots$, $c_n = b_1 + b_2 + \ldots + b_n$. Повторив эту процедуру k раз, он получает, наконец, последовательность чисел $t_1, t_2, \ldots, t_n$. Джеймс Бонд переписывает числа t_i в *произвольном* порядке в файл и отправляет шифровку в Центр.

Вам необходимо составить программу, которая по заданному набору чисел t_i восстанавливает исходную последовательность чисел $a_1, a_2, \ldots, a_n$.

Формат входных данных

В первой строке записано два целых числа n и k — количество чисел в секретном сообщении и количество шагов в процедуре шифрования соответственно ($2 \leqslant n \leqslant 10^5$; $1 \leqslant k \leqslant 3$).

Во второй строке записаны в произвольном порядке n целых положительных чисел $t_1, t_2, \ldots, t_n$, полученных из исходного набора после k шагов шифрования ($1 \leqslant t_i \leqslant 10^{18}$).

Формат выходных данных

Запишите через пробел числа $a_1, a_2, \ldots, a_n$ в том же порядке, в котором они стояли в исходной последовательности. Если решений несколько, выведите любое из них.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	20	$2 \leqslant n \leqslant 5$, $k = 1$	
2	20	$2 \leqslant n \leqslant 10^3$, $k = 1$	1
3	30	$2 \leqslant n \leqslant 10^5$, $k = 1$	1, 2
4	30	$2 \leqslant n \leqslant 10^5$, $1 \leqslant k \leqslant 3$	1, 2, 3

Пример

стандартный ввод	стандартный вывод
3 2 10 19 4	4 2 3

Пояснение к примеру

В примере исходный секретный набор `4, 2, 3` сначала преобразуется в последовательность чисел `4, 4+2=6, 4+2+3=9`, а после второго шага шифрования — в набор `4, 4+6=10, 4+6+9=19`.

Задача 3. Суперфакториальная СС (7-11 кл.)

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`256 мегабайт`

Говорят, что на одной из планет нашей галактики местные жители используют *суперфакториальную* систему счисления. В этой системе любое положительное рациональное число представляется в виде суммы чисел, обратных к факториалам:

$$\frac{p}{q} = a_1 + \frac{a_2}{2!} + \frac{a_3}{3!} + \ldots + \frac{a_n}{n!}\,.$$

Здесь a_1 — целое неотрицательное число, а все остальные целые a_k удовлетворяют неравенствам $0 \leqslant a_k < k$. Незначащие нули в конце суперфакториальной записи числа p/q отбрасываются. Например, дробь $1/5$ записывается в этой системе счисления в виде набора `0 0 1 0 4`, так как

$$\frac{1}{5} = \mathbf{0} + \frac{\mathbf{0}}{2!} + \frac{\mathbf{1}}{3!} + \frac{\mathbf{0}}{4!} + \frac{\mathbf{4}}{5!}\,.$$

Ваша задача – найти представление заданного положительного рационального числа p/q в суперфакториальной системе счисления.

Формат входных данных

В единственной строке записаны через пробел два целых числа p и q ($1 \leqslant p \leqslant 10^6$, $1 \leqslant q \leqslant 10^6$).

Формат выходных данных

Единственная строка должна содержать последовательность разделенных пробелом целых чисел $a_1, a_2, \ldots, a_n$, образующих запись числа $p\,/\,q$ в суперфакториальной системе счисления. Если таких представлений несколько, выведите любое из них.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	10	$1 \leqslant p, q \leqslant 5$	
2	25	$1 \leqslant p, q \leqslant 20$	1
3	25	$1 \leqslant q \leqslant 20$	1, 2
4	40	$1 \leqslant p, q \leqslant 10^6$	1, 2, 3

Примеры

стандартный ввод	стандартный вывод
1 2	0 1
2 10	0 0 1 0 4
10 2	5

Задача 4. Мини-Тетрис 2.0 (7-11 классы)

Имя входного файла: `стандартный ввод`
Имя выходного файла: `стандартный вывод`
Ограничение по времени: `1 секунда`
Ограничение по памяти: `256 мегабайт`

Имеются прямоугольные плитки двух типов — одноклеточные квадраты размером 1×1 и двухклеточные прямоугольники размером 1×2. С помощью этих плиток необходимо замостить полосу размером $2 \times n$, при этом плитки не должны накладываться друг на друга и каждая клетка полосы должна быть покрыта ровно одной плиткой. Плитки разрешается поворачивать.

Вам необходимо подсчитать количество способов замощения полосы $2 \times n$ с помощью плиток указанного вида.

Формат входных данных

Во входных данных записано одно целое число n — длина полосы ($1 \leqslant n \leqslant 10^{18}$).

Формат выходных данных

Выведите количество способов замощения полосы с помощью плиток указанного вида. Ответ запишите по модулю $(10^9 + 9)$.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

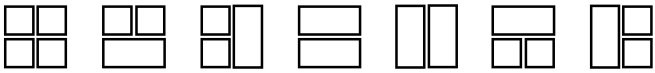
Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	10	$1 \leqslant n \leqslant 10$	
2	30	$1 \leqslant n \leqslant 10^4$	1
3	30	$1 \leqslant n \leqslant 10^6$	1, 2
4	30	$1 \leqslant n \leqslant 10^{18}$	1, 2, 3

Примеры

стандартный ввод	стандартный вывод
2	7
3	22

Пояснение к примеру

В первом примере $n = 2$, дорожка имеет размеры 2×2, и её можно замостить с помощью указанных плиток семью способами:



Задача 5. Идеальные наборы (9-11 классы)

Имя входного файла: `стандартный ввод`
Имя выходного файла: `стандартный вывод`
Ограничение по времени: `2 секунды`
Ограничение по памяти: `256 мегабайт`

На базарах города Старогиреевка все продавцы взвешивают товары с помощью наборов старых гирь, каждая из которых весит целое число граммов, а суммарный вес всех гирь составляет n граммов. Такой набор называется *идеальным*, если любой груз весом в целое число граммов от 1 до n может быть уравновешен некоторым количеством гирь данного набора, и притом единственным образом. Груз всегда кладется на левую чашу весов, гири — на правую. Два способа взвешивания, отличающиеся лишь заменой некоторых гирь на другие того же веса, считаются одинаковыми. Для $n = 5$ идеальных наборов три: $(1, 1, 1, 1, 1)$, $(1, 1, 3)$ и $(1, 2, 2)$.

Каждый продавец на рынке предпочитает иметь собственный набор гирь. Найдите для каждого i-го продавца все идеальные наборы, имеющие наименьшее количество гирь общим весом в n_i граммов.

Формат входных данных

В первой строке указано количество продавцов на базаре t ($1 \leqslant t \leqslant 100$). Во второй строке записаны t натуральных чисел $n_1, n_2, \ldots, n_t$, где n_i — общий вес всех гирь для i-го продавца ($1 \leqslant n_i \leqslant 10^5$).

Формат выходных данных

Для каждого i-го продавца выведите в первой строке два числа: m — количество идеальных наборов, имеющих наименьшее количество гирь, и k — количество гирь в каждом таком наборе, затем в следующих m строках — идеальные наборы из k гирь общим весом n_i граммов, перечисленных в порядке неубывания массы. Все наборы выведите в лексикографическом порядке.

Примеры

стандартный ввод	стандартный вывод
2 1 5	1 1 1 2 3 1 1 3 1 2 2

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены.

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	10	$t = 1,\ \ 1 \leqslant n \leqslant 10$	
2	25	$t = 1,\ 1 \leqslant n \leqslant 100$	1
3	25	$t = 1,\ 1 \leqslant n \leqslant 10^5$	1, 2
4	40	$1 \leqslant t \leqslant 100,\ 1 \leqslant n_i \leqslant 10^5$	1, 2, 3

Региональный этап, 2022-2023

Региональный этап Всероссийской олимпиады школьников состоялся 21-23 января 2023 года. Соревнование проходило в два тура с перерывом (21 января) между ними на день отдыха. На каждый тур олимпиады школьникам предлагалось 4 задачи. Ниже представлены тексты и разбор решения этих задач.

Условия задач, тесты, решения и разбор задач подготовили Александр Бабин, Константин Бац, Никита Голиков, Владимир Новиков, Даниил Орешников, Михаил Первеев, Андрей Станкевич, Григорий Хлытин.

Общая информация о проверке

Во всех задачах баллы за подзадачу начисляются только, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Информация о тестировании приведена в таблице.

Тип информации	Пояснение
Полная	Для каждого теста подзадачи сообщается результат работы программы участника на этом тесте.
Первая ошибка	Для подзадачи сообщается одно из двух: • если все тесты пройдены, то сообщаются баллы за подзадачу; • если хотя бы один тест не пройден, сообщается номер первого не прошедшего теста и результат работы программы участника на этом тесте.
Баллы	Для подзадачи сообщаются баллы за эту подзадачу.

Задача 1. Разделение прямоугольника

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`512 мегабайт`

Аня играет в новую настольную игру «Клетчатое королевство». Рассмотрим прямоугольное клетчатое поле размером $a \times b$. Необходимо разделить его на m прямоугольников вертикальными или горизонтальными разрезами. Прямоугольники не обязательно должны получиться равными. Необходимо суммарно провести ровно k разрезов. Каждый разрез представляет собой прямую линию от одного края поля до другого края поля. Разрезы разрешено делать только по границам клеток — линиям сетки.

Вам необходимо определить количество возможных горизонтальных ($0 \leqslant h < a$) и вертикальных ($0 \leqslant v < b$) разрезов. Если поле можно разрезать несколькими способами, выведите тот, в котором горизонтальных разрезов меньше. Если поле нельзя разрезать требуемым образом, выведите `-1`.

Формат входных данных

В первой строке дано ровно одно целое число t — количество тестов ($1 \leqslant t \leqslant 100$).

В следующих t строках находится описание тестов: в i-й строке через пробел даны четыре целых числа: a, b, k, m — высота и ширина поля, количество разрезов и количество прямоугольников соответственно ($1 \leqslant a, b \leqslant 10^9, 0 \leqslant k \leqslant 2 \cdot 10^9, 1 \leqslant m \leqslant 10^{18}$, $k < m$).

Формат выходных данных

Для каждого теста выведите через пробел ровно два целых числа h и v – количество горизонтальных и количество вертикальных разрезов, если прямоугольное клетчатое поле можно разрезать требуемым образом, в противном случае выведите число `-1`.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач

успешно пройдены. Для всех подзадач сообщается информация «Первая ошибка».

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	18	$a = 1$	
2	19	$1 \leqslant m \leqslant 10^5$	
3	20	$1 \leqslant k \leqslant 10^5$	2
4	21	$1 \leqslant m \leqslant 10^9$	2
5	22	нет	1 – 4

Пример

стандартный ввод	стандартный вывод
3 2 2 1 2 1 2 2 3 3 5 5 12	0 1 -1 2 3

Пояснение к примеру

В приведенном примере содержится три теста:

1) В первом тесте $a = 2, b = 2, k = 1, m = 2$, и поле можно разрезать, как показано на рис. 4.

2) Во втором тесте поле нельзя разрезать требуемым образом.

3) В третьем тесте $a = 3, b = 5, k = 5, m = 12$, и поле можно разрезать, как показано на рис. 5.

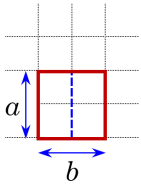


Рис. 4

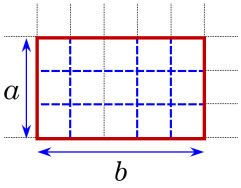


Рис. 5

Задача 2. Произведение Фибоначчи

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`512 мегабайт`

Напомним, что последовательность чисел Фибоначчи определяется следующим образом: $F_0 = 1$, $F_1 = 1$, $F_n = F_{n-2} + F_{n-1}$. Последовательность чисел Фибоначчи начинается так: 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

Дано натуральное число n. Требуется подсчитать количество способов представить его в виде произведения чисел Фибоначчи, каждое из которых больше 1.

Формат входных данных

Первая строка ввода содержит целое число t — количество тестов ($1 \leqslant t \leqslant 50$).

Следующие t строк содержат тесты, каждая строка содержит одно целое число n ($2 \leqslant n \leqslant 10^{18}$).

Формат выходных данных

Для каждого теста вывести одно число — искомое количество способов.

Пример

стандартный ввод	стандартный вывод
5 2 7 8 40 64	1 0 2 2 3

Пояснение к примеру

В примере число 2 можно представить в виде произведения чисел Фибоначчи единственным способом $2 = 2$; число 7 нельзя представить в виде произведения чисел Фибоначчи; числа 8 и 40 — двумя способами: $8 = 2 \cdot 2 \cdot 2 = 8$ и $40 = 2 \cdot 2 \cdot 2 \cdot 5 = 5 \cdot 8$; наконец, число 64 – тремя способами: $64 = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 2 \cdot 2 \cdot 2 \cdot 8 = 8 \cdot 8$.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Для всех подзадач сообщается информация «Первая ошибка».

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	15	$2 \leqslant n \leqslant 100$	
2	17	$2 \leqslant n \leqslant 10^5$	1
3	9	$n = 2^k$	
4	38	$2 \leqslant n \leqslant 10^9$	1, 2
5	38	$2 \leqslant n \leqslant 10^{18}$	1–4

Задача 3. Робот-пылесос

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`512 мегабайт`

Рассмотрим координатную плоскость, которую планируется очищать с использованием робота-пылесоса. Робот-пылесос представляет собой квадрат размером $k \times k$ со сторонами, параллельными осям координат. Изначально левый нижний угол робота находится в точке $(0, 0)$, а правый верхний — в точке (k, k).

Вам дана последовательность из n перемещений робота по плоскости; каждое i-е перемещение характеризуется направлением d_i, принимающим значения `'N'` (вверх, увеличение координаты Y), `'S'` (вниз, уменьшение координаты Y), `'W'` (влево, уменьшение координаты X) или `'E'` (вправо, увеличение координаты X), и целым числом a_i — расстоянием, на которое робот перемещается. На рисунке 6 приведены примеры возможных перемещений робота в каждом направлении.

Робот в каждый момент времени убирает всю площадь под собой. Иными словами, точка считается убранной тогда и только тогда, когда она в какой-то момент времени принадлежала квадрату размера $k \times k$, на котором находился робот. По заданным перемещениям робота посчитайте суммарную площадь всей убранной поверхности.

Формат входных данных

В первой строке ввода через пробел даны два целых числа: размер робота k и количество команд n ($1 \leqslant k \leqslant 10^4; 1 \leqslant n \leqslant 10^5$).

В i-й из следующих n строк через пробел даны направление i-го перемещения d_i и его расстояние a_i (d_i — буква `'N'`, `'S'`, `'W'` или `'E'`; $1 \leqslant a_i \leqslant 10^9$).

Формат выходных данных

Выведите суммарную площадь убранной роботом поверхности.

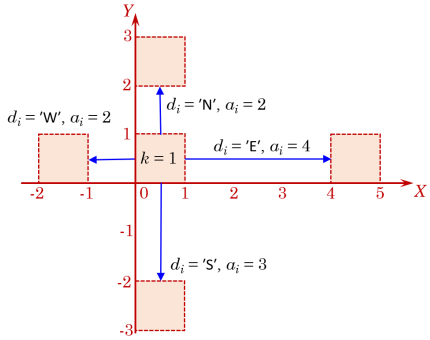


Рис. 6

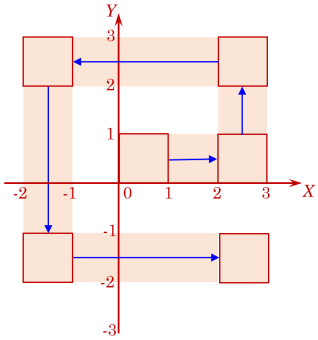


Рис. 7

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Для всех подзадач сообщается информация «Первая ошибка».

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	9	$k = 1$, $n \leqslant 10$, $a_i \leqslant 10$	
2	10	$k \leqslant 10$, $n \leqslant 10$, $a_i \leqslant 100$	1
3	11	$k \leqslant 10^3$, $n \leqslant 10^3$, $a_i = 1$	
4	8	$k \leqslant 10^4$, $n \leqslant 10^5$, $a_i = k$	
5	14	$k = 1$, $n \leqslant 10^3$, $a_i \leqslant 10^9$	1
6	15	$k \leqslant 10^4$, $n \leqslant 10^3$, $a_i \leqslant 10^9$	1–3, 5
7	16	$k = 1$, $n \leqslant 10^5$, $a_i \leqslant 10^9$	1, 5
8	17	$k \leqslant 10^4$, $n \leqslant 10^5$, $a_i \leqslant 10^9$	1–7

Пример

стандартный ввод	стандартный вывод
1 5 E 2 N 2 W 4 S 4 E 4	17
3 4 W 2 N 1 W 1 N 2	27

Пояснение к примеру

На рисунке 7 приведена иллюстрация к перемещениям робота из первого примера. Клетки, которые робот посетил за время своих перемещений, затемнены.

Задача 4. Разноцветные точки

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`512 мегабайт`

Рассмотрим n точек на плоскости, пронумерованных от 1 до n, обозначим их как $P_1, P_2, \ldots, P_n$, координаты i-й точки (x_i, y_i).

Рассмотрим следующий процесс. Выберем номер *начальной* точки i и номер *следующей* за ней точки j $(i \neq j)$, а также целое число t. После этого номер *прицельной* точки k вычисляется по следующему алгоритму. Рассмотрим вектор $\overrightarrow{P_iP_j}$, направленный из точки P_i в точку P_j. Упорядочим все точки, кроме j-й, по углу, отсчитывая против часовой стрелки от направления вектора, равного $\overrightarrow{P_iP_j}$, отложенного из точки j. При равенстве угла будем упорядочивать точки по возрастанию расстояния до точки j. В качестве точки k выбирается точка, являющаяся t-й в данном порядке при нумерации с единицы. Далее точка j становится начальной, а точка k — следующей за ней, после чего, пользуясь тем же алгоритмом, вычисляется номер прицельной точки. Этот процесс повторяется до бесконечности.

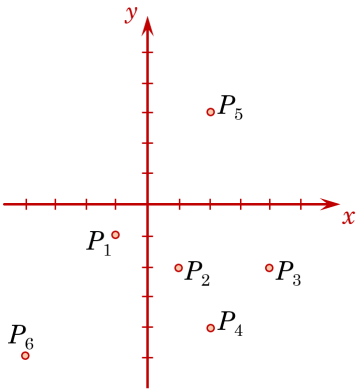


Рис. 8

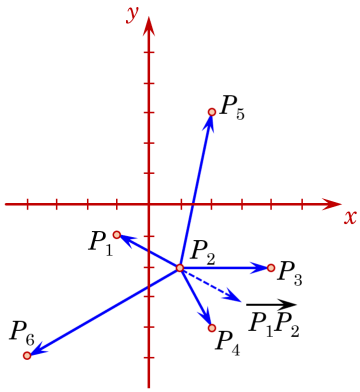


Рис. 9

Для лучшего понимания процесса рассмотрим следующий пример. Пусть имеются 6 точек, изображенных на рис. 8, и $t = 4$. Пусть номер начальной точки равен 1, а номер следующей за ней

точки равен 2. Отложим вектор $\overrightarrow{P_1P_2}$ от точки P_2 и отсортируем все точки, кроме точки P_2, по углу, отсчитывая против часовой стрелки от направления данного вектора. На рис. 9 отложенный вектор обозначен пунктирной линией, а также для удобства проведены векторы из точки P_2 во все остальные точки.

Точки будут упорядочены следующим образом: P_3, P_5, P_1, P_6, P_4. Таким образом, номер прицельной точки равен 6. Далее точка 2 становится начальной, а точка 6 — следующей.

На рис. 10 изображён процесс для начальной точки 2 и следующей точки 6. Точки будут упорядочены следующим образом: P_4, P_3, P_2, P_1, P_5. Обратите внимание, что точка P_1 в этом списке находится раньше, чем точка P_5, так как расстояние от точки P_1 до точки P_6 меньше, чем расстояние от точки P_5 до точки P_6. Прицельная точка будет иметь номер 1.

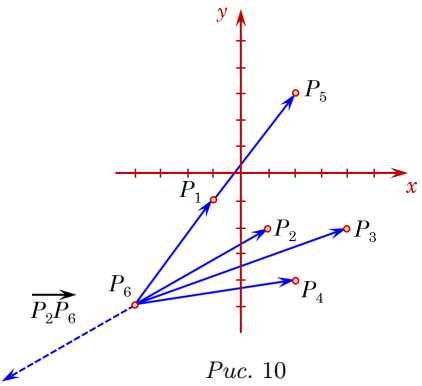


Рис. 10

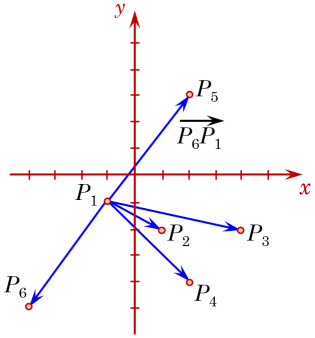


Рис. 11

На рис. 11 изображён процесс для начальной точки 6 и следующей точки 1. Обратите внимание, что в данном случае вектор $\overrightarrow{P_6P_1}$, отложенный из точки P_1 совпадает с вектором $\overrightarrow{P_1P_5}$, отложенным из точки P_1. Эти векторы изображены сплошной линией. Точки будут упорядочены следующим образом: P_5, P_6, P_4, P_2, P_3. Прицельная точка будет иметь номер 2. Таким образом, далее процесс начнётся для начальной точки 1 и следующей точки 2 и зациклится.

Покрасим каждую из n точек в один из трёх цветов. Цвет i-й точки определяется следующим образом:

- Пусть существует такая точка j, что, выбрав точку i в качестве начальной, а точку j в качестве следующей, в результате описанного процесса точка i побывает начальной бесконечное количество раз. В этом случае точка i будет покрашена в **зелёный** цвет.
- Пусть точка i не была покрашена в зеленый цвет и существует такая точка j, что, выбрав точку i в качестве начальной, а точку j в качестве следующей, в результате описанного процесса точка i побывает начальной ещё хотя бы один раз. В этом случае точка i будет покрашена в **синий** цвет.
- Пусть точка i не была покрашена ни в зеленый, ни в синий цвет. В этом случае точка i будет покрашена в **красный** цвет.

Для каждой точки определите, в какой цвет её нужно покрасить.

Формат входных данных

Первая строка содержит два целых числа n и t ($2 \leqslant n \leqslant 1\,000$; $1 \leqslant t \leqslant n - 1$).

Каждая из следующих n строк содержит два целых числа x_i и y_i ($-10^9 \leqslant x_i, y_i \leqslant 10^9$). Гарантируется, что никакие две точки не совпадают.

Формат выходных данных

Выведите строку, состоящую из n символов: i-й символ строки должен обозначать цвет i-й точки. Для зелёной точки выведите букву «`G`», для синей точки — букву «`B`», а для красной точки — букву «`R`».

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Для всех подзадач предоставляется информация «Первая ошибка». В подзадаче 7 все точки являются вершинами строго выпуклого многоугольника и даны в порядке обхода против часовой стрелки.

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	10	$n \leqslant 10$, все точки на одной прямой	
2	15	все точки на одной прямой	1
3	10	$n \leqslant 10$, гарантируется, что нет синих точек	
4	10	$n \leqslant 10$	1, 3
5	15	$n \leqslant 100$, гарантируется, что нет синих точек	3
6	15	$n \leqslant 100$	1, 3, 4, 5
7	5	$n \geqslant 3$, точки – вершины многоугольника	
8	20	нет	1–7

Примеры

стандартный ввод	стандартный вывод
6 4 -1 -1 1 -2 4 -2 2 -4 2 3 -4 -5	GGBBRG
2 1 1 1 2 2	GG

Пояснение к примеру

Рассмотрим некоторые точки из первого примера.

Точка P_1 окрашены в зелёный цвет, потому что можно выбрать точку P_2 в качестве следующей, и процесс посетит точку P_1 бесконечное количество раз. Данный пример был рассмотрен выше в условии задачи.

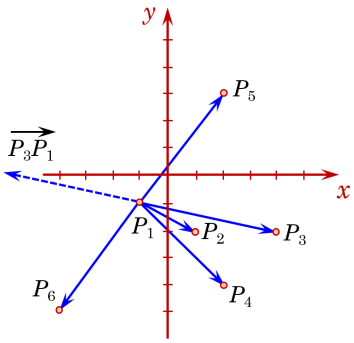


Рис. 12

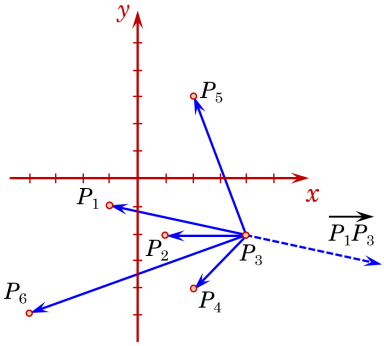


Рис. 13

Можно показать, что точка P_3 не является зелёной, однако она является синей, так как можно выбрать точку 1 в качестве следующей, точка 3 окажется начальной ещё хотя бы один раз. Процесс для начальной точки 1 и следующей точки 3 проиллюстрирован на рисунках 12, 13 и 14.

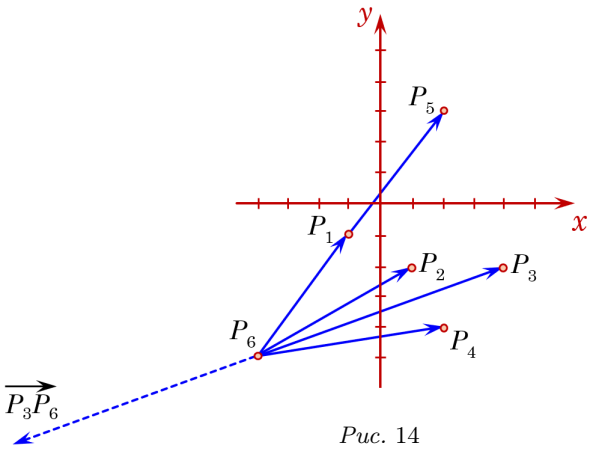


Рис. 14

Для начальной точки 3 и следующей точки 1 точки будут упорядочены следующим образом: P_6, P_4, P_2, P_3, P_5. Точка с номером 3 становится прицельной. Далее для начальной точки 1 и следующей точки 3 точки будут упорядочены следующим образом: P_5, P_1, P_2, P_6, P_4. Точка с номером 6 становится прицельной.

Наконец, для начальной точки 3 и следующей точки 6 точки будут упорядочены следующим образом: P_4, P_3, P_2, P_1, P_5. Точка с номером 1 становится прицельной. Далее процесс продолжится с начальной точкой 6 и следующей точкой 1. Из примера, описанного выше в условии задачи, мы знаем, что процесс зациклится после посещения точек с номерами 6, 1 и 2.

Во втором примере из условия легко показать, что если одна из точек является начальной, а другая — следующей, то прицельной станет точка, которая являлась начальной. Поэтому обе точки будут окрашены в зелёный цвет.

Задача 5. Метрострой

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`512 мегабайт`

Буровая установка «Мегабур 2022» для прокладки туннелей метро Байтсбурга имеет n двигателей. Питание установки устроено таким образом, что на все двигатели подается одно и то же целочисленное напряжение x.

У каждого двигателя есть два режима, если на него подается напряжение x, то i-й двигатель работает в первом режиме, если $x \leqslant z_i$ и во втором режиме, если $x > z_i$.

При этом i-й двигатель характеризуется удельной мощностью a_i в первом режиме и b_i во втором режиме. Это означает, что увеличение напряжения на 1 когда двигатель находится в первом режиме, приводит к увеличению его мощности на a_i, а во втором режиме приводит к увеличению его мощности на b_i. Иначе говоря, при подаче напряжения x, если i-й двигатель находится в первом режиме он работает с мощностью $a_i x$, а если во втором режиме, то с мощностью $a_i z_i + b_i(x - z_i)$.

Для прокладки туннеля суммарная мощность двигателей должна быть не меньше p. Какое минимальное целочисленное напряжение необходимо подать на установку, чтобы суммарная мощность двигателей была больше или равна p?

Формат входных данных

Первая строка ввода содержит целые числа n и p ($1 \leqslant n \leqslant 100$, $1 \leqslant p \leqslant 10^{12}$).

Следующие n строк описывают двигатели и содержат по три целых числа z_i, a_i, b_i ($1 \leqslant z_i \leqslant 10^9, 1 \leqslant a_i, b_i \leqslant 10^4$).

Формат выходных данных

Требуется вывести одно целое число — минимальное напряжение, которые необходимо подать на установку.

Пример

стандартный ввод	стандартный вывод
1 6 4 1 2	5
3 15 2 3 3 4 2 1 5 2 2	3

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Для всех подзадач предоставляется информация «Первая ошибка».

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	20	$n = 1$	
2	20	$a_i, b_i \leqslant 100,\ p \leqslant 10^5$	
3	20	у всех двигателей z_i одинаковые	1
4	20	$n \leqslant 2$	1
5	20	нет	1–4

Задача 6. Красивые последовательности

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`512 мегабайт`

Дано множество A, элементами которого являются различные целые числа от 1 до 8.

Рассмотрим последовательность $[a_1, a_2, \ldots, a_n]$ из n целых чисел, каждое из которых выбрано из множества A. Будем называть эту последовательность *красивой*, если для любого числа x все элементы последовательности, равные x, находятся на расстоянии не меньше x друг от друга. Иначе говоря, для любого числа x и для любых двух индексов $1 \leqslant i < j \leqslant n$, таких, что $a_i = a_j = x$, должно выполняться неравенство $j - i \geqslant x$.

Требуется посчитать количество красивых последовательностей для заданного числа n и множества A, и вывести остаток от деления этого количества на число $10^9 + 7$.

Формат входных данных

В первой строке даны два целых числа n и m – длина последовательности и количество элементов множества A ($1 \leqslant n \leqslant 100$, $1 \leqslant m \leqslant 8$).

Во второй строке даны m различных целых чисел a_i в порядке возрастания – элементы множества A ($1 \leqslant a_i \leqslant 8$, $a_i < a_{i+1}$).

Формат выходных данных

Выведите одно целое число — остаток от деления количества красивых последовательностей на число $10^9 + 7$.

Примеры

стандартный ввод	стандартный вывод
3 2 1 2	5

Пояснение к примеру

В примере красивыми являются последовательности $[1, 1, 1]$, $[1, 1, 2]$, $[1, 2, 1]$, $[2, 1, 1]$, $[2, 1, 2]$. Последовательности $[2, 2, 2]$,

$[1, 2, 2], [2, 2, 1]$ красивыми не являются, так как в каждой из них существуют два элемента со значением 2, находящиеся на расстоянии 1 друг от друга.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Для всех подзадач предоставляется информация «Первая ошибка».

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	5	$A = \{1, 2\}$, $n \leqslant 10$	
2	10	$A = \{1, 2\}$, $n \leqslant 30$	1
3	15	$A = \{1, 2\}$	1, 2
4	20	$A = \{1, k\}$ для $2 \leqslant k \leqslant 8$	1–3
5	30	$a_i \leqslant 5$	1–3
6	20	нет	1–5

Задача 7. Камни

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`512 мегабайт`

Перед Бобом выложены в ряд n чёрных камней, пронумерованных от 1 до n. На i-м камне записано целое число a_i. Для каждого числа от 1 до n известно, что оно записано ровно на одном камне, иными словами числа a_i образуют перестановку. Будем называть соседними для i-го камня $(i - 1)$-й и $(i + 1)$-й камни (если они существуют).

Боб выполняет следующие n шагов:

- На первом шаге Боб выбирает произвольное i от 1 до n и красит i-й камень в белый цвет.
- На шагах с номерами от 2 до n Боб смотрит на такие чёрные камни, которые являются соседними для хотя бы одного белого камня, из них он выбирает камень j с минимальным a_j и красит его в белый цвет.

Несложно заметить, что к концу выполнения всех шагов перед Бобом будут лежать n белых камней.

Алиса выбрала q пар значений p_j и k_j. Для каждой пары она хочет выяснить, сколько существует различных способов выбрать камень на первом шаге, которые приведут к тому, что камень с номером p_j станет белым ровно на k_j-м шаге.

Помогите Бобу ответить на q запросов Алисы.

Формат входных данных

На первой строке заданы два целых числа: n — количество камней ($2 \leqslant n \leqslant 10^5$) и q — количество запросов ($1 \leqslant q \leqslant 10^5$).

На второй строке заданы записанные на камнях целые числа $a_1, a_2, \ldots, a_n$ ($1 \leqslant a_i \leqslant n$, все a_i различны).

На следующих q строках заданы запросы, j-й запрос задается парой целых чисел p_j и k_j ($1 \leqslant p_j \leqslant n, 1 \leqslant k_j \leqslant n$) — номером камня и номером шага, на котором этот камень должен быть покрашен в белый цвет.

Формат выходных данных

Для каждого запроса выведите количество значений i, таких что если i-й камень будет покрашен в белый цвет на первом шаге, то p_j-й камень покрасится в белый цвет на k_j-м шаге.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Для всех подзадач предоставляется информация «Первая ошибка».

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	20	$n \leqslant 300$, $q \leqslant 300$	
2	17	$n \leqslant 3\,000$	1
3	12	$n \leqslant 50\,000$, $q \leqslant 10$	
4	6	значения a_i возрастают	
5	16	все значения k_i равны	
6	15	все значения p_i равны	
7	14	нет	1–6

Пример

стандартный ввод	стандартный вывод
6 4 1 4 6 5 2 3 3 1 2 2 6 3 4 3	1 2 1 2
5 3 5 2 3 4 1 2 3 4 4 3 2	0 1 1

Пояснение к примеру

В первом тестовом примере операции выполняются следующим образом:

- Если на первом шаге был выбран 1-й камень:
  1-й шаг: [**1**, 4, 6, 5, 2, 3], 2-й шаг: [**1, 4**, 6, 5, 2, 3], 3-й шаг:

[**1, 4, 6**, 5, 2, 3], 4-й шаг: [**1, 4, 6, 5**, 2, 3], 5-й шаг: [**1, 4, 6, 5, 2**, 3], 6-й шаг: [**1, 4, 6, 5, 2, 3**].

- Если на первом шаге был выбран 2-й камень:
  1-й шаг: [1, **4**, 6, 5, 2, 3], 2-й шаг: [**1, 4**, 6, 5, 2, 3], 3-й шаг: [**1, 4, 6**, 5, 2, 3], 4-й шаг: [**1, 4, 6, 5**, 2, 3], 5-й шаг: [**1, 4, 6, 5, 2**, 3], 6-й шаг: [**1, 4, 6, 5, 2, 3**].
- Если на первом шаге был выбран 3-й камень:
  1-й шаг: [1, 4, **6**, 5, 2, 3], 2-й шаг: [1, **4, 6**, 5, 2, 3], 3-й шаг: [**1, 4, 6**, 5, 2, 3], 4-й шаг: [**1, 4, 6, 5**, 2, 3], 5-й шаг: [**1, 4, 6, 5, 2**, 3], 6-й шаг: [**1, 4, 6, 5, 2, 3**].
- Если на первом шаге был выбран 4-й камень:
  1-й шаг: [1, 4, 6, **5**, 2, 3], 2-й шаг: [1, 4, 6, **5, 2**, 3], 3-й шаг: [1, 4, 6, **5, 2, 3**], 4-й шаг: [1, 4, **6, 5, 2, 3**], 5-й шаг: [1, **4, 6, 5, 2, 3**], 6-й шаг: [**1, 4, 6, 5, 2, 3**].
- Если на первом шаге был выбран 5-й камень:
  1-й шаг: [1, 4, 6, 5, **2**, 3], 2-й шаг: [1, 4, 6, 5, **2, 3**], 3-й шаг: [1, 4, 6, **5, 2, 3**], 4-й шаг: [1, 4, **6, 5, 2, 3**], 5-й шаг: [1, **4, 6, 5, 2, 3**], 6-й шаг: [**1, 4, 6, 5, 2, 3**].
- Если на первом шаге был выбран 6-й камень:
  1-й шаг: [1, 4, 6, 5, 2, **3**], 2-й шаг: [1, 4, 6, 5, **2, 3**], 3-й шаг: [1, 4, 6, **5, 2, 3**], 4-й шаг: [1, 4, **6, 5, 2, 3**], 5-й шаг: [1, **4, 6, 5, 2, 3**], 6-й шаг: [**1, 4, 6, 5, 2, 3**].

Задача 8. Обыкновенная задача про строки

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`512 мегабайт`

Назовём две строки s и t эквивалентными, если для любой строки u длины 2, количество вхождений u в s совпадает с количеством вхождением u в t. Таким образом, строки «`aaaba`», «`abaaa`» и «`baaab`» попарно эквивалентны между собой (строка «`aa`» входит два раза, строка «`ab`» один раз, строка «`ba`» один раз, строка «`bb`» не входит как подстрока), а строки «`abb`» и «`bba`» — нет.

В этой задаче вам будут даны Q строк, состоящих из символов «a», «b» и «c», для каждой из которых надо будет посчитать количество эквивалентных им непустых строк, также состоящих из символов «a», «b» и «c». Так как это количество может быть очень большим, то надо вывести его остаток при делении на $10^9 + 7$.

Формат входных данных

В первой строке дано число G – номер подзадачи, к которой относится текущий тест. Для теста из примера $G = 0$.

На второй строке дано число q ($1 \leqslant q \leqslant 10^5$), затем следуют q строк, состоящих из символов «a», «b» и «c». Суммарная длина строк не превышает 10^6.

Формат выходных данных

Требуется вывести q целых чисел – для каждой строки необходимо вывести количество эквивалентных ей по модулю $10^9 + 7$.

Примеры

стандартный ввод	стандартный вывод
0 4 abaa abca ccbca bacc	3 3 2 1

Пояснение к примерам

Строке «abaa» эквивалентны строки «abaa», «aaba», «baab».

Строке «abca» эквивалентны строки «abca», «bcab», «cabc».

Строке «ccbca» эквивалентны строки «ccbca» и «cbcca».

Строке «bacc» эквивалентна только строка «bacc».

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Для всех подзадач предоставляется информация «Первая ошибка». За n_i обозначена длина i-й строки во

входных данных, за L обозначена сумма длин строк, за w — максимальный ответ (не взятый по модулю) среди всех запросов.

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	11	строка s не содержит символов «c»	
2	13	символы «a» и «c» в строке s не встречаются рядом	1
3	11	$n_i \leqslant 13$	
4	10	$L \leqslant 40$	3
5	9	$L \leqslant 60$	3, 4
6	13	$w \leqslant 100$; $L \leqslant 10^5$	
7	33	нет	1–6

Задача 9. Красивое название (7-8 классы)

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`256 мегабайт`

В городе строится новый IT-центр, и теперь осталось только выбрать для него название.

Как выяснили в Институте Нейролингвистического Программирования, красивое название IT-центра гарантирует его высокую посещаемость, а значит, нужно сделать всё возможное, чтобы его название было красивым.

Из мировой статистики выявлен критерий красоты названия. Красивым можно считать название, в котором между любыми двумя согласными буквами есть хотя бы одна гласная, а между любыми двумя гласными есть как минимум одна согласная буква. Поскольку вариантов названий очень много, требуется автоматизировать проверку названий на красоту.

Вам необходимо по заданному названию выяснить, является ли оно красивым.

Формат входных данных

В первой строке записано единственное целое число n — длина названия ($1 \leqslant n \leqslant 10^5$). Во второй строке записана последовательность из n латинских букв — исследуемое название.

Формат выходных данных

Единственная строка должна содержать «`yes`», если название является красивым, или «`no`» — в противном случае.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Для всех подзадач предоставляется полная информация о прохождении тестов.

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	20	$1 \leqslant n \leqslant 10^2$	
2	25	$1 \leqslant n \leqslant 10^3$	1
3	25	$1 \leqslant n \leqslant 10^4$	1, 2
4	30	$1 \leqslant n \leqslant 10^5$	1, 2, 3

Пример

стандартный ввод	стандартный вывод
5 Banan	yes
12 TatneftArena	no
1 a	yes

Пояснение к примеру

В латинском алфавите шесть гласных букв — `a`, `e`, `i`, `o`, `u`, `y`, остальные являются согласными.

Задача 10. Почти гипотеза 3X+1 (7-8 классы)

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	`1 секунда`
Ограничение по памяти:	`256 мегабайт`

Одной из самых известных нерешённых задач математики является знаменитая гипотеза $3X + 1$, которая утверждает, что с помощью нескольких операций `A` и `B` из любого целого положительного числа n всегда можно получить число 1. Операции `A` и `B` задаются следующим образом:

`A`. Преобразование целого числа x в число $3x + 1$;

`B`. Преобразование целого числа x в целое число $x/2$ (целочисленное деление).

Мы предлагаем вам проверить видоизменённую версию этой гипотезы: *можно ли из* 1 *получить заданное целое число* n *с помощью нескольких операций* `A` *и* `B`?

Например, число $n = 10$ можно получить из 1 с помощью пяти операций: `ABABA` (или `AABBA`), т.к. $1 \xrightarrow{A} 4 \xrightarrow{B} 2 \xrightarrow{A} 7 \xrightarrow{B} 3 \xrightarrow{A} 10$.

Вам необходимо составить программу, которая находит последовательность операций `A` и `B` для получения заданного целого числа n.

Формат входных данных

В единственной строке записано целое число n, которое необходимо получить с помощью последовательности операций `A` и `B` ($2 \leqslant n \leqslant 10^{18}$).

Формат выходных данных

Выведите `-1`, если заданное число n невозможно получить с помощью указанных операций. Единственная строка должна содержать последовательность символов `A` и `B` без пробелов. Если

таких представлений несколько, выведите любое из них. Общее количество символов в строке не должно превышать 1 000.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Для всех подзадач предоставляется информация «Первая ошибка».

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	10	$2 \leqslant n \leqslant 10$	
2	20	$2 \leqslant n \leqslant 10^3$	1
3	30	$2 \leqslant n \leqslant 10^9$	1, 2
4	40	$2 \leqslant n \leqslant 10^{18}$	1–3

Пример

стандартный ввод	стандартный вывод
10	ABABA

Решения задач

Задача 1. Розы

Автор задачи : *Фольклор*
Разработчик : *Киндер М.И.*
Разбор задачи : *Киндер М.И.*

При считывании значения числа лепестков l очередной розы проверяем условие $l > k$. При выполнении этого условия добавляем к текущему значению выручки слагаемое $2r$; в противном случае стоимость розы уменьшаем на величину $(k - l)$ и к значению выручки добавляем $r - (k - l)$.

Приведём код этой программы на языке C++.

```
int k, r, n, l;
cin >> k >> r >> n;

long long ans = 0;
for ( int i = 0; i < n; ++i)  {
  cin >> l;
  if (l > k) ans += 2 * r;
  else ans += r - (k - l);
}
cout << ans;
```

Задача 2. До последней стружки

Автор задачи : *Фольклор.*
Разработчик : *Киндер М.И.*
Разбор задачи : *Киндер М.И.*

Предположим, что имеется x заготовок; подсчитаем количество втулок, которое можно получить из этих заготовок. На первом этапе получится x втулок и x единиц стружки; из них в свою очередь можно получить x/m новых втулок (целочисленное де-

ление). Значит, на втором этапе имеем $s = x + x \,/\, m$ втулок и $x \,/\, m + x \,\%\, m$ единиц стружки. (Число $x \,\%\, m$ равно остатку от деления x на m.) На следующем этапе рассуждения повторяются: из $x_1 = x \,/\, m$ единиц стружки снова получаем $x_1 \,/\, m$ втулок и такое же количество единиц стружки, то есть число втулок увеличится на $x_1 \,/\, m$. Таким образом, для подсчёта числа бронзовых втулок из x заготовок можно использовать процедуру:

```
long long g(long long x) {
    long long s = x;
    while (x >= m) {
        s += x / m;
        x += x / m + x % m;
    }
    return s;
}
```

Как найти необходимое количество заготовок? Например, можно воспользоваться бинарным поиском по ответу. Идея этого алгоритма состоит в следующем. Будем искать нужное количество заготовок в виде неизвестного числа x на отрезке $[1;\, n]$. Выберем в качестве начального значения середину этого отрезка, то есть число $x = n/2$. Если этого количества заготовок недостаточно, будем продолжать дальнейший поиск на отрезке $[n/2; n]$, иначе — на отрезке $[1; n/2]$. На каждом шаге выбираем середину текущего отрезка и сравниваем количество втулок, которое можно получить (процедура g), с требуемым количеством. Если полученных втулок меньше, передвинем *левую* границу искомого отрезка в эту середину, иначе — передвинем *правую* границу. Продолжаем этот процесс до тех пор, пока разность между левой и правой границей текущего отрезка больше 1.

Сложность этого алгоритма — $O(\log n \cdot \log m)$.

Задача 3. Произведение цифр

Автор задачи : *Фольклор.*
Разработчик : *Киндер М.И.*
Разбор задачи : *Киндер М.И.*

В задаче требуется найти наибольшее число, составленное из различных цифр, произведение цифр которого равно заданному значению n ($1 \leqslant n \leqslant 10^9$).

Для небольших значений n задачу можно решить, перебирая числа в порядке возрастания; для каждого из чисел нужно подсчитать произведение цифр в его десятичной записи, затем сравнить полученное произведение с заданным числом n.

Разложим число n в произведение простых множителей $n = p_1^{a_1} p_2^{a_2} \dots p_k^{a_k}$. (Как известно, для этого достаточно выполнить количество операций порядка $\sqrt{n}$.) Так как каждая цифра искомого числа меньше 10, это произведение должно содержать только простые множители $p_i < 10$. Поэтому если в разложение n входят простые числа p_i, большие 10, запишем ответ `-1`.

Итак, в разложение n могут входить только простые множители 2, 3, 5 и 7.

Теперь выясним, какие значения могут принимать показатели a_i этих простых множителей. В десятичной записи искомого числа x могут встретиться только четыре чётные цифры 2, 4, 6, 8, их произведение делится на 2^7, поэтому максимальный показатель простого числа $p = 2$ равен 7. Аналогично, есть только три цифры 3, 6, 9, которые делятся на 3; их произведение делится на 3^4, поэтому максимальный показатель простого числа $p = 3$ равен 4. Для простых чисел 5 и 7 максимальный показатель, очевидно, равен 1. Поэтому если в разложение n простые числа 2, 3, 5 и 7 входят с показателями больше 7, 4, 1 и 1 соответственно, снова запишем ответ `-1`.

Ясно, что цифры 5 и 7 входят в искомое число x только в случае, если показатели простых чисел 5 и 7 равны 1. Осталось разобрать случай $n = 2^{a_2} \cdot 3^{a_3}$, где $0 \leqslant a_2 \leqslant 7$, $0 \leqslant a_3 \leqslant 4$.

При $a_3 = 4$ запись числа x обязательно содержит цифры 3, 6 и 9, причём если $a_2 = 0$ — ответ `-1`. Если же $a_2 = 1$, других чётных цифр, кроме 6, у числа x нет. Для $a_2 = 2$ в записи x будут входить чётные цифры 2 и 6, и так далее. Аналогичным образом, разбираются остальные значения a_3.

Задача 4. Ковбой Джонни

Автор задачи : *China*
Разработчик : *Киндер М.И.*
Разбор задачи : *Киндер М.И.*

Пусть a_i — номер коровы, стоящей на месте i. Будем называть корову, стоящую на месте i, просто коровой i. В задаче требуется подсчитать количество последовательностей $a_1, a_2, \ldots, a_n$, которые удовлетворяют следующим ограничениям:

- $1 \leqslant a_i \leqslant m$ для каждого i от 1 до n;
- $a_i < a_{i+2}$ для каждого i от 1 до $n-2$;
- $a_i < a_{i+3}$ для каждого i от 1 до $n-3$.

Подзадача 1. Переберём все возможные нумерации для каждой коровы i и отберём среди них те, которые удовлетворяют условию задачи.

Подзадача 2. Воспользуемся идеей динамического программирования. Пусть $f_{i-1,j,k,l}$ — количество нумераций всех коров в предположении, что уже выбраны номера для коров от 1 до $i-1$, причём $a_{i-1} = j$, $a_{i-2} = k$ и $a_{i-3} = l$. Интересующие нас нумерации $f_{i,x,j,k}$, где $x = a_i$, можно получить из каждого варианта нумерации $f_{i-1,j,k,l}$ коров добавлением номера коровы i в виде числа x, большего k и l, и номера коровы $i+1$ как числа, большего j и k. Сложность такого решения — $O(nm^4)$.

Подзадача 3. Снова будем опираться на идею динамического программирования. Пусть $f_{i,j,k}$ — количество нумераций всех коров в предположении, что уже выбраны номера для коров от 1 до i, и пусть $a_i = j$ и $\max(a_{i-1}, a_{i-2}) = k$. По условию $a_{i+1} > k$, причём при переходе от i к $i+1$ имеем: $\max(a_i, a_{i-1}) \geqslant \max(j, k)$, то есть наименьшее возможное значение $\max(a_i, a_{i-1})$ равно $\max(j, k)$. Перебирая такие нумерации, как в подзадаче 2, получим решение, сложность которого $O(nm^3)$.

Подзадача 4. Пусть $m = \lceil n/2 \rceil$. Из условия задачи следует, что $a_i < a_{i+2} < a_{i+4} < \ldots$ для любого i. В случае чётного n ($n = 2m$) количество чётных и нечётных мест совпадает с количеством номеров m, значит, и на чётных и на нечётных местах

использованы все номера от 1 до m ровно по одному разу, причём их положение определяется однозначным образом. Другими словами, в случае чётного n существует только одна требуемая нумерация коров.

Если же n — нечётное ($n = 2m - 1$), то количество нечётных мест совпадает с m, и значит, положение номеров на нечётных местах определяется однозначно. Количество же чётных мест равно $m - 1$, поэтому ровно одно из чисел от 1 до m не использовано на этих местах. Значит, в случае нечётного n существует m требуемых нумераций.

Подзадача 5. Пусть $f_{i,j}$ — количество нумераций i коров вида $a_1, a_2, \ldots, a_i$ с номерами в диапазоне $[1; j]$. Начальные значения этой величины очевидны: $f_{0,j} = 1$ и $f_{1,j} = j$. Тогда ответом в задаче будет число $f_{n,m}$.

При ограничениях задачи $1 < i + 1 < j \leqslant n$ должно выполняться $a_i < a_j$.

Для подсчёта вариантов нумераций i коров рассмотрим возможные положения коров с номерами j. Сразу заметим, что если $a_k = j$, то $k \in \{ i-1, i \}$. Действительно, если $k < i - 1$, то $a_i > a_k$, но a_k принимает самое большое возможное значение j в диапазоне $[1; j]$, противоречие.

Все возможные положения коров с номерами j можно описать с помощью четырёх множеств A_j (в соответствии с условием — $a_{i-1} = j$ или $a_i = j$):

1. $A_j = \varnothing$.

В последовательности нет номера j, количество таких нумераций равно $f_{i,j-1}$.

2. $A_j = \{i\}$.

Для коровы i есть только один возможный номер $a_i = j$. Значит, первые $(i - 1)$ чисел просто составляют последовательность с весовым диапазоном $[1; j - 1]$, поэтому для этой ситуации количество нумераций составляет $f_{i-1,j-1}$.

3. $A_j = \{i - 1, i\}$.

Другими словами, $a_{i-1} = a_i = j$. Эта ситуация разбирается досточно просто. Поскольку первые $(i - 2)$ чисел составляют последовательность с весовым диапазоном $[1; j - 1]$, значит, общее число нумераций будет $f_{i-2,j-1}$.

4. $A_j = \{i-1\}$.

Другими словами, $a_{i-1} = j$. В этом случае подсчёт числа нумераций более сложный.

Так как $a_i \neq j$, должны выполняться неравенства $1 \leqslant a_i < j$. Рассматривая нумерации с числом $k = a_i \in [2; j-1]$, заметим, что первые $(i-2)$ числа образуют последовательность с рангом $[1; k-1]$. Всего получаем $\sum_{k=1}^{j-1} f_{i-2,k-1}$ способов нумерации, или, что то же самое, $\sum_{k=0}^{j-2} f_{i-2,k}$ способов.

Таким образом, уравнение перехода имеет вид

$$f_{i,j} = f_{i,j-1} + f_{i-1,j-1} + f_{i-2,j-1} + \sum_{k=0}^{j-2} f_{i-2,k}\,.$$

Используя префикс-суммы, можно оптимизировать подсчёт числа нумераций $f_{i,j}$ по этой формуле. Предварительный подсчёт суммы за $O(nm)$ операций, позволяет получить ответ за $O(1)$ операций. Окончательная сложность алгоритма — $O(n^2 + T)$.

Задача 5. Леон и Ронни

Автор задачи : *Фольклор.*
Разработчик : *Киндер М.И.*
Разбор задачи : *Киндер М.И.*

Упорядочим массив исходных чисел на карточках: $a_1 > a_2 > \ldots > a_{n-1} > a_n$. Сначала разберём

Пример. Заданное слово `RLRL ... RL`.

Распределим карточки с числами так: первым ходом Ронни забирает число a_n (для краткости — `R`), затем Леон забирает число a_{n-1} (для краткости — `L`), число a_{n-2} — `L`, число a_{n-3} заберёт `R`, и так далее. При этом последовательность результатов будет такой: `RLRL ... RL`. Это вытекает из такого утверждения.

Лемма. *Если* $a_1 > a_2 > \ldots > a_{n-1} > a_n > 0$, *то из двух сумм*

$$a_1 + a_3 + a_5 + \ldots \quad \text{и} \quad a_2 + a_4 + a_6 + \ldots$$

(содержащих все n *данных чисел) больше первая, в которую попадает наибольшее число* a_1.

Для её доказательства надо при чётном n сложить неравенства $a_1 > a_2$, $a_3 > a_4$, ..., $a_{n-1} > a_n$, а при нечётном n: $a_1 > a_2$, $a_3 > a_4$, ..., $a_n > 0$.

Это же утверждение, но в применении к отрезку $a_k > a_{k+1} > > \ldots > a_{s-1} > a_s$, состоящему из $s - k + 1$ чисел, будем использовать и в общем случае.

Общий случай. Разберём ситуацию с произвольным словом S. Пусть S — заданное слово из букв `L` и `R`. Выпишем символы слова S в обратном порядке и обозначим через S' полученную последовательность букв `L` и `R`.

Отсортируем в порядке убывания последовательность a_1, a_2, ..., a_n исходных чисел, то есть будем считать, что $a_1 > a_2 > > \ldots > a_n$. Для определенности предположим, что первый символ слова S' — буква `L`. (В противном случае в рассуждениях нужно поменять местами буквы `L` и `R`). Передадим Леону карточки с числами a_1, a_3, a_5, ..., а Ронни — карточки с числами a_2, a_4, a_6, ... Очевидно, что при таком распределении карточек сумма чисел у Леона будет больше (независимо от того, кому достанется наименьшее число).

Теперь, переходя от каждой буквы слова S' к следующей, будем снимать карточки у Леона и Ронни в соответствии со следующим правилом:

удалим карточку Леона с самым большим числом из оставшихся a_i, если очередная буква `L` *меняется на* `R`*, то есть сделаем ход a_i* `L`*. (Если буква* `R` *поменялась на* `L`*, то соответственно самое большое число a_i удалим из карточек Ронни, то есть сделаем ход a_i* `R`*); если очередная буква в слове S' такая же, как предыдущая, то снимаем карточку с наименьшим числом.*

При этом каждый раз остается последовательность из занумерованных подряд карточек, которые по очереди забирают Леон и Ронни; к ней применима лемма.

Пусть, например, задан набор чисел 7, 6, 5, 4 и слово `LLRL`. Преобразованное слово S' имеет вид `LRLL`. Поскольку в слове S' первая буква `L`, распределим карточки между игроками так, чтобы наибольшее число 7 досталось Леону:

Леон	Ронни	Результат
5, 7	4, 6	L
передаём 7 L		
5	4, 6	R
передаём 6 R		
5	4	L
передаём 4 R		
5		L
передаём 5 L		

Для получения ответа осталось записать полученную последовательность ходов в «обратном» порядке, то есть ответом в примере будет: 5 L, 4 R, 6 R, 7 L.

Региональный этап, 2021-2022

Задача 1. Чемпионат по устному счету

Автор задачи : *Первеев М.*
Разработчик : *Жюри ЦПМК.*
Разбор задачи : *Жюри ЦПМК.*

Подзадача 1. Заметим, что если команд второго типа нет, то достаточно поддерживать сумму чисел на доске и при изменении числа вычитать из суммы старое значение и прибавлять новое. Впрочем, ограничения первой подзадачи позволяют после при аккуратной реализации каждого изменения в цикле обновлять сумму массива.

Подзадача 2. Рассмотрим циклический сдвиг массива на 1. Для этого просто выполним присваивание $a[i] = a[i-1]$ для всех i в убывающем порядке, а $a[1]$ присвоим старое значение $a[n]$ (не забудем его сохранить). При ограничениях второй подзадачи это можно сделать за $O(n)$.

Подзадача 3. В этой подазадаче уже требуется выполнять циклический сдвиг для произвольного k. К счастью в условии приведен вид массива после циклического сдвига, осталось его аккуратно реализовать. Снова возможна реализация за $O(n)$.

Подзадача 4. Для решения задачи на полный балл необходимо научиться выполнять команды этого типа быстрее. Будем поддерживать переменную s — на сколько вправо сдвинут исходный массив на данный момент времени. Тогда в случае, если текущая команда второго типа, нужно увеличить число s на k. Осталось понять, как обрабатывать команды первого типа.

Заметим, что если некоторый элемент находился изначально на позиции i, то теперь он находится на позиции $(i + s) \bmod n$, если нумеровать позиции с нуля. Таким образом, если необходимо изменить элемент с номером p на x, изменим элемент массива с номером $(p-s) \bmod n$ на x. Вычислить новую сумму массива легко — нужно из старой суммы вычесть старое значение элемента, а затем прибавить новое значение.

Получили решение за $O(n + q)$.

Задача 2. Прыгающий робот

Автор задачи : *Андреева Е.*
Разработчик : *Жюри ЦПМК.*
Разбор задачи : *Жюри ЦПМК.*

Подзадача 1. Для решения этой подзадачи можно воспользоваться полным перебором. Заметим, что значения $a > \max\{d_i\}$ нет смысла перебирать.

Зафиксируем первую платформу и значение a. Проверим, подходит ли это значение. Асимптотика решения $O(n^2 \max\{d_i\})$.

Подзадача 2. Избавимся от перебора значений a. Пусть мы зафиксировали начальную платформу. Найдем минимальное значение a, которое подходит. Инициализируем значение $a = 0$. Пробежимся по всем платформам, если текущей ловкости не хватает, чтобы перепрыгнуть на следующую, мы знаем, на какую величину её надо увеличить. При этом увеличение ловкости не влияет на возможность совершать прыжки между рассмотренными ранее платформам. Получаем решение за $O(n^2)$.

Подзадача 3. Здесь нам задана начальная платформа, поэтому можно не реализовывать её перебор. Применим решение предыдущей подзадачи и найдём подходящее начальное значение ловкости.

Подзадача 4. Исследуем ещё раз процесс поиска начальной ловкости после фиксирования стартовой платформы i:

```
int a=0;
for (int k=0; k<n; k++) {
    if (d[(i+k)%n] > a) {
        a=d[(i+k)%n];
    }
    a++;
}
a-=n;
```

Изменим немного этот фрагмент, избавимся от операции `a++`, включив накопленное значение в сравнение:

```
int a = 0;
for (int k = 0; k < n; k++) {
```

```
        if (d[(i + k) % n] > a + k) {
            a = d[(i + k) % n] − k;
        }
    }
```

Перенося `+k` на другую сторону неравенства, получаем окончательно код

```
int a=0;
for (int k=0; k<n; k++) {
    if (d[(i+k)%n]-k > a) {
        a=d[(i+k)%n]-k;
    }
}
```

в котором легко узнать поиск максимума в массиве $d[(i+k)\%n]-k$.

Заменим массив d на массив $d'[k] = d[k] - k$ и удвоим его: $d'[k+n] = d'[k]+n$. Тогда, чтобы учесть сдвиг индекса на i, надо взять максимум значений этого массива на полуинтервале $[i; i+n)$ и прибавить к нему значение i.

Таким образом, мы пришли к задаче поиска максимума на отрезке. Эта задача широко известна и в этой подзадаче можно применить любую из известных структур данных, например, дерево отрезков или разреженную таблицу.

Подзадача 5. Это «учебная» подзадача, которая проверяет, что решение участников умеет генерировать массив по формуле для $f = 2$. Поскольку перебирать начальную платформу не надо, решение за линейное время с поиском максимума в массиве работает.

Подзадача 6. В этой подзадаче массив имеет размер порядка 10^7, поэтому разреженные таблицы не помещаются в память, а дерево отрезков сверху не проходит по времени. Здесь можно либо использовать очень оптимизированное дерево отрезков в реализации снизу, дерево Фенвика, либо воспользоваться структурой данных для «максимума в окне».

Есть несколько разных реализаций этой структуры данных, одна из наиболее простых следующая. Нам нужно найти максимумы на полуинтервалах $[0; n)$, $[1; n+1)$, $\ldots$, $[n; 2n)$. Посчитаем суффиксные максимумы на полуинтервалах $[0; n)$, $[1; n)$, $[2; n)$, и

так далее, а также префиксные максимумы на полуинтервалах $[n; n+1)$, $[n; n+2)$, и так далее. Заметим, что максимум на нужном нам полуинтервале это максимум из двух предподсчитанных значений.

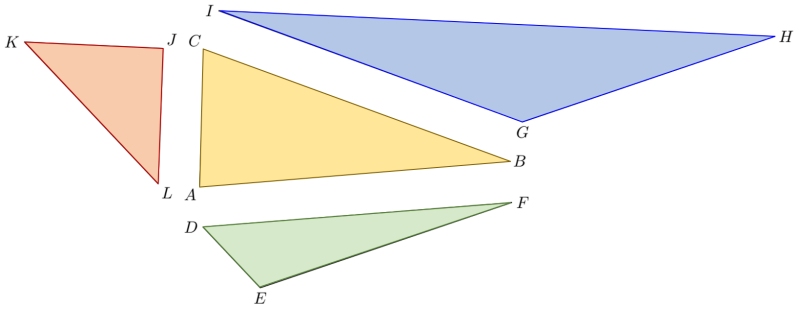
Задача 3. Треугольная головоломка

Автор задачи : *Мамай И.*
Разработчик : *Жюри ЦПМК.*
Разбор задачи : *Жюри ЦПМК.*

Тест 3. Из любой четвёрки треугольников можно собрать треугольник.

Тесты 4 и 5. В этих тестах треугольник можно собрать только из четырёх треугольников, одинаковых с точностью до поворота и сдвига.

Тесты 6– 12. Дополнительные ограничения, наложенные на входные данные, позволяют уменьшить количество вариантов, которые нужно перебрать в решении по сравнению с полным. Так, в тестах 6 и 7, а также 10, 11 и 12 треугольники не надо вращать. В тестах 8 и 9 треугольники может потребоваться повернуть, но легче проверить, что из четырёх треугольников можно собрать один.



Полное решение. Перебором найдём треугольник, который будет располагаться в центре, пусть это ΔABC. Переберём три треугольника, которые будут располагаться в углах и касаться сторон AB, BC и CA. А также, для каждого из трёх треугольников, которые будут располагаться в углах, перебором определим циклический сдвиг вершин. Пусть ΔDEF касается стороны AB

стороной DF, ΔGHI касается стороны BC стороной GI, и ΔJKL касается стороны CA стороной JL. Тогда эти четыре треугольника собираются в один большой, если:

- $AB = DF$,
- $BC = GI$,
- $CA = JL$,
- $\angle BAC + \angle FDE + \angle JLK = 180°$,
- $\angle CBA + \angle IGH + \angle DFE = 180°$,
- $\angle ACB + \angle LJK + \angle GIH = 180°$.

Проверить, что стороны равны по длине можно абсолютно точно, вычислив квадрат длины стороны по формуле Пифагора.

Чтобы проверить, что три угла в сумме равны 180°, нужно вычислить углы. Например, по теореме косинусов, воспользовавшись затем функцией `acos`. И проверить, что сумма углов достаточно близка к 180° — отличается не более чем на ε. При этом, необходимо выбрать достаточно маленькое значение погрешности вычислений ε. А именно, если углы вычисляются в радианах, можно показать, что достаточно взять $\varepsilon \approx 10^{-12}$ при ограничениях на координаты точек до 10^6.

Таким образом, решение перебирает $n^4 \cdot 3^3$ вариантов.

Задача 4. Массивы-палиндромы

Автор задачи :	*Хлытин Г.*
Разработчик :	*Жюри ЦПМК.*
Разбор задачи :	*Жюри ЦПМК.*

Здесь и далее будут использоваться следующие обозначения: $A[l, \ldots, r]$ — массив, полученный из элементов $A[l]$, $A[l+1]$, … , $A[r]$ массива A. Будем называть массив $A[l, \ldots, r]$ *подмассивом* массива A. Индексация всех массивов с нуля, границы отрезков включительно.

Подзадачи 1 и 3. Для решения первой и третьей подзадачи можно воспользоваться следующим алгоритмом.

Будем перебирать позицию pos_1 в массиве A, а затем будем рассматривать массивы $A' = A\left[pos_1, \ldots, \min(n-1, pos_1 + m - 1)\right]$

и $B' = B[0, \ldots, \min(n-1-pos_1, m-1)]$. Так как полученные массивы A' и B' имеют одинаковую длину $k = \min(n - pos_1, m)$, то Кай может поэлементно их просуммировать и получить массив C длины k.

Аналогично будем перебирать позицию pos_2 в массиве B, а затем будем рассматривать массивы $B' = B[pos_2, \ldots, \min(m - 1, pos_2 + n - 1)]$ и $A' = A[0, \ldots, \min(m - 1 - pos_2, n - 1)]$. Так как полученные массивы B' и A' имеют одинаковую длину $k = \min(m - pos_2, n)$, то Кай может поэлементно их просуммировать и получить массив C длины k.

Заметим, что теперь ответ на задачу — длина максимального палиндрома в массиве C.

Так как у каждого палиндрома есть его «центр» в массиве C, переберём (отдельно для чётных и нечётных длин палиндромов) всевозможные центры x ($0 \leqslant x < k$), где k — длина массива C.

Для каждого такого центра x нам нужно найти наибольшую длину палиндрома len с центром в x. Для этого будем изменять $i \geqslant 0$ в разные стороны от него и проверять элементы массива на равенство: $C[x - i] = C[x + i]$ для нечётного случая, либо $C[x - i] = C[x + i + 1]$ — для чётного случая. Если это равенство будет верным для всех $i \leqslant q$, то искомая длина палиндрома равна $len = 2 \cdot q + 1$ в нечётном случае и $len = 2 \cdot q + 2$ — в чётном случае. Таким образом, мы сможем найти длину максимального палиндрома в массиве C со сложностью $O(k^2)$, так как сделаем $O(k)$ операций на перебор всех центров, а потом для каждого центра $O(k)$ операций для поиска наибольшей длины палиндрома с центром в нём.

Таким образом, асимптотика этого решения будет $O((n+m) \cdot k^2) = O((n+m) \cdot \min(n, m)^2)$. Этого хватит для решения первой подзадачи, но не хватит для решения третьей подзадачи.

Подзадачи 2 и 3. Заметим, что мы уже умеем искать наибольшую длину палиндрома len с фиксированным центром x за $O(\log k)$ операций.

Давайте посчитаем полиномиальные хеши

$$h[i] = C[0] \cdot P^{i-1} + C[1] \cdot P^{i-2} + \ldots + C[i-1] \cdot P^0$$

для всех подмассивов $C[0, \ldots, i-1]$ массива C, $h[0] = 0$. Аналогично посчитаем полиномиальные хеши $h_{rev}[i]$ для всех подмассивов

$C_{rev}[0, \ldots, i-1]$ массива C_{rev}, $h_{rev}[0] = 0$, где C_{rev} — массив, полученный переворачиванием массива C. Для фиксированного массива C сложность такого предпосчета $O(k)$.

Тогда для любого подмассива $C[l, \ldots, r]$ за $O(1)$ подсчитаем значения прямого и обратного хеша

$$hash(l, r) = h[r+1] - h[l] \cdot P^{r-l+1},$$
$$has_{hrev}(l, r) = hrev[k-l] - h[k-r-1] \cdot P^{r-l+1}.$$

Теперь найдем ответ при помощи бинарного поиска по длине len палиндрома с центом в x (отдельно для чётных и нечётных длин палиндромов). Будем проверять на равенство прямой и обратный хеш: $hash(x-i, x) = hash_{rev}(x, x+i)$ для нечётного случая, либо $hash(x-i, x) = hash_{rev}(x+1, x+i+1)$ для чётного случая, а затем сдвигать границы бинарного поиска.

Такое решение будет работать со сложностью $O((n+m) \cdot k \cdot \log k)$, что равно $O((n+m) \cdot \min(n, m) \cdot \log(\min(n, m)))$. Заметим, что найти самый длинный палиндром в массиве C можно еще быстрее, например при помощи алгоритма Манакера, который работает со сложностью $O(k)$. Итоговая асимптотика решения в этом случае будет $O((n+m) \cdot \min(n, m))$.

Для решения второй подзадачи заметим, что нам всего лишь нужно найти длину максимального палиндрома в массиве A, так как известно, что все элементы массива B одинаковые.

В массиве A найдем длину максимального палиндрома нечётной длины t_1, а также длину максимального палиндрома чётной длины t_2 при помощи хешей или алгоритма Манакера, как было описано выше.

Обозначим длину массива B через m. Если m меньше чем t_1, то $t_1 = m$ или $t_1 = m - 1$ — чтобы длина t_1 оставалась нечётной. Аналогично, если m меньше чем t_2, то $t_2 = m$ или $t_2 = m - 1$ — чтобы длина t_2 оставалась чётной.

Ответом на задачу будет максимум из длин t_1 и t_2.

Подзадача 4. Предпосчитаем прямые и обратные полиномиальные хеши для каждого из массивов A и B, как это было расписано выше.

Воспользуемся бинарным поиском по ответу ans (отдельно для чётных и нечётных длин палиндромов). Пусть длина палин-

дрома $ans = 2 \cdot q + 2$ или $ans = 2 \cdot q + 1$ в чётном и нечётном случае соответственно. Пусть константа $f = 1$ для чётного случая и $f = 0$ для нечётного случая.

- Инициализируем множество $set = [\ldots]$, в котором будем хранить целые числа.
- Перебираем все центры x массива A. Для каждого такого центра x добавляем (`add`) в `std::set S` разность хешей подмассивов
  $$hash(A[x - q, \ldots, x]) - hash_{rev}(A[x + f, \ldots, x + q + f]).$$
- Перебираем все центры y массива B. Для каждого такого центра y проверяем (`contains`), содержит ли S разность хешей подмассивов
  $$hash_{rev}(B[y + f, \ldots, y + q + f]) - hash(B[y - q, \ldots, y]).$$
  Если содержит, – значит, можно сделать палиндром длины ans, иначе — нельзя.
- Сдвигаем границы бинарного поиска.

Чтобы избежать коллизий, нужно делать вычисления сразу по нескольким простым модулям, например, $M_1 = 10^9 + 7$ и $M_2 = 10^9 + 9$, и в качестве результата хранить пару из двух целых чисел $\{u, v\}$ — вычисления по первому и второму модулю соответственно.

Пусть, для примера, $hash_1 = \{u_1, v_1\}$ и $hash_2 = \{u_2, v_2\}$, тогда $hash_1 + hash_2 = \{(u_1 + u_2) \bmod M_1, (v_1 + v_2) \bmod M_2\}$ и аналогично для других операций.

Решение работает за $O((n+m) \cdot \log(\min(n, m)) \cdot W)$, где $O(W)$ — сложность операции добавления (`insert`) и проверки принадлежности (`count`) для `std::set`.

Если использовать «`std::unordered_set`» в `C++`, то $O(W)$ будет в среднем $O(1)$.

Задача 5. Новый год в детском саду

Автор задачи :	*Акилов Д.*
Разработчик :	*Жюри ЦПМК.*
Разбор задачи :	*Жюри ЦПМК.*

Сначала переформулируем условие задачи в более простой форме: для данных n, a и b нужно посчитать количество пар целых чисел (x, y) таких, что $0 \leqslant x \leqslant a$, $0 \leqslant y \leqslant b$, $x+y > 0$ и $(x+y)$ делится на n. Стоит обратить внимание, что в некоторых подзадачах (4, 5, 7, 8) ответ может быть очень большим, и чтобы не было переполнения в таких языках, как C++, нужно использовать целочисленный тип данных, который умеет работать с числами порядка 10^{18} (в C++ это, например, `long long int`).

Подзадача 1. В первой подзадаче достаточно перебрать все пары (x, y) и для каждой отдельно проверить, подходит она или нет.

```
ans = 0;
for x in range (a + 1):
    for y in range (b + 1):
        ans += (x + y > 0 and (x + y) % n == 0)
print(ans)
```

Подзадача 2. В этой подзадаче нужно найти количество целых чисел $0 < y \leqslant b$, которые делятся на n. Нам подходят значения n, $2n$, $3n$, $\ldots$, $\lfloor \frac{b}{n} \rfloor n$, то есть всего вариантов $\lfloor \frac{b}{n} \rfloor$.

Подзадача 3. Можно заметить, что сумма $x + y$ удовлетворяет неравенствам $0 < (x + y) < 2n$ и делится на n, значит, она равна n. Это значит, что для каждого конкретного значения x мы можем точно сказать, что $y = n - x$. Тогда можно перебрать все x от 0 до a и проверить, что пара $(x, n - x)$ является корректной.

Подзадача 4. На самом деле мы можем перебирать только x и в этой подзадаче. Только теперь мы не знаем, чему точно равно значение y. Но мы можем сказать, какой оно имеет вид: остаток от деления y на n должен быть сравним с $-x$ (чтобы сумма делилась на n). Теперь мы свели исходную задачу к следующей: для данных n, r и b найти количество целых чисел $0 \leqslant y \leqslant b$, которые дают остаток r при делении на n. Эту задачу несложно свести к случаю $r = 0$: можно заметить, что $y \geqslant r$, тогда можно вычесть из y и из b значение r. Другими словами, нужно найти количество значений y, делящихся на n и удовлетворяющих неравенствам $0 \leqslant y \leqslant b - r$. Мы уже научились это делать во второй подзадаче, за исключением некоторых деталей:

- нужно отдельно обрабатывать случай $b - r < 0$;
- значение $y = 0$ теперь подходит, то есть теперь формула для числа решений имеет вид $\lfloor \frac{b-r}{n} \rfloor + 1$;
- нужно в конце вычислений вычесть 1, так как мы посчитали пару $(0, 0)$, которая нам не подходит.

Подзадача 5. Вернемся к решению первой подзадачи. Заметим, что мы можем перебирать не сами значения x и y, а их остатки по модулю n. Тогда для каждой пары остатков можно проверить, подходит она или нет, после чего найти количество вариантов для x и количество вариантов для y.

Подзадача 6. Воспользуемся решением третьей подзадачи. У нас для каждого остатка x подходит только один остаток y.

```
def num_of_r(x, r, m):
    return 0 if x<r else (x-r)//m+1

t = int(input())
for tn in range(t):
    n, a, b = map(int, input().split())
    ans = 0
    for ra in range(n):
        rb = (n-ra)%n
        ans += num_of_r(a, ra, n)*num_of_r(b, rb, n)
    print(ans-1)
```

Подзадача 7. Рассмотрим 4 группы пар (x, y):

1. $k_x n \leqslant x < (k_x + 1)n < a$; $k_y n \leqslant y < (k_y + 1)n < b$;
2. $\lfloor \frac{a}{n} \rfloor n \leqslant x \leqslant a$; $k_y n \leqslant y < (k_y + 1)n < b$;
3. $k_x n \leqslant x < (k_x + 1)n < a$; $\lfloor \frac{b}{n} \rfloor n \leqslant y \leqslant b$;
4. $\lfloor \frac{a}{n} \rfloor n \leqslant x \leqslant a$; $\lfloor \frac{b}{n} \rfloor n \leqslant y \leqslant b$.

По сути, мы разбили возможные значения x и y на блоки размера n. (Последний блок может быть меньшего размера, если число не делится нацело на n). Нам это поможет, так как нам на самом деле неважно, какой именно из полных блоков мы рассматриваем.

В первом случае у нас для каждого целого x ровно один подходящий вариант ($x = k_x n + r$, тогда $y = k_y n + n - r$), всего пар полных блоков $\lfloor \frac{a}{n} \rfloor \cdot \lfloor \frac{b}{n} \rfloor$, в каждом блоке есть n подходящих значений x.

Во втором случае у нас также для каждого целого x будет ровно одно подходящее значение y (так как второй блок полный). В этом случае у нас всего $a \,\%\, n + 1$ возможных значений x.

Третий случай аналогичен второму, только x и y поменялись местами.

В четвёртом случае нам подходят только те x, для которых $x + b \,\%\, n \geqslant n$, то есть $n - b \,\%\, n \leqslant x \leqslant a \,\%\, n$, для них также будет всего один подходящий y.

Суммируя все 4 случая, получаем формулу:

```
t = int(input())
for tn in range(t):
  n, a, b = map(int, input().split())
  print((a // n) * (b // n) * n + (a % n + 1) * (b // n) +
    (b % n + 1) * (a // n) + max(0, a % n + b % n - n + 1))
```

Задача 6. Сортировка дробей

Автор задачи :	*Царёв Ф.*
Разработчик :	*Жюри ЦПМК.*
Разбор задачи :	*Жюри ЦПМК.*

Сначала отметим, что поскольку значения в массивах не превышают 10^6, две неравные дроби различаются хотя бы на 10^{-12}. Это даёт нам возможность при сравнении двух дробей использовать вещественные числа, хотя в целом возможно и решение этой задачи в целых числах. Для сокращения дроби перед выводом необходимо воспользоваться алгоритмом Евклида, во всех решениях необходимо это сделать непосредственно перед выводом ответа.

Подзадачи 1 и 2. Для их решения достаточно сложить все дроби в массив, отсортировать по возрастанию и выводить ответы на запросы. В подзадаче 1 можно воспользоваться любой

сортировкой, в том числе квадратичной. В подзадаче 2 нужно воспользоваться быстрой сортировкой, например, встроенной в язык программирования.

Начиная с подзадачи 3 значение n может достигать 10^5 и сгенерировать все дроби, сложив их в массив или просто рассмотрев каждую по разу, не хватает времени и памяти. Разобьём дроби на классы по значению знаменателя b_i. Отсортируем массивы a и b по возрастанию.

Подзадача 3. В этой задаче все запросы не превышают 100, то есть надо найти первые 100 дробей. Научимся генерировать дроби в порядке возрастания.

Заметим, что при фиксированном знаменателе минимальное значение — значение с минимальным числителем. При этом некоторые из этих значений могут быть уже использованы при генерации. Получаем следующий алгоритм: для каждого знаменателя b_i храним минимальный не использованный числитель $a_{pos[i]}$. Чтобы получить значение следующей по величине дроби, находим минимальное значение $a_{pos[i]} \,/\, b_i$ за $O(n)$, отправляем эту дробь в вывод, а значение $pos[i]$ увеличиваем на 1.

Следует особо обратить внимание, что в процессе генерации дробей в порядке возрастания дроби с каким-либо знаменателем могут «закончиться» и следует аккуратно рассмотреть этот случай. Решение работает за $O(n \max\{c_j\})$.

Подзадача 4. В этой подзадаче можно также сгенерировать необходимое количество дробей в порядке возрастания, так как $c_j \leqslant 10^5$. Однако генерация одной дроби за $O(n)$ уже не подходит, необходимо получать минимальную неиспользованную дробь быстрее.

Сложим все дроби $a_{pos[i]} \,/\, b_i$ в приоритетную очередь, дерево отрезков или `std::set`. Теперь мы можем находить минимальную неиспользованную дробь за $O(\log n)$, поэтому получаем решение за $O(n + \max\{c_j\} \log n)$.

Подзадача 5. Рассмотрим теперь два решения полной версии задачи.

Решение 1. Рассмотрим один запрос c_k. Найдем вещественное значение ответа f с использованием двоичного поиска. Чтобы проверить, верно ли, что $f \geqslant m$ для некоторого значения m, най-

дём d — количество дробей, меньших f. Для этого используем метод двух указателей: заметим, что при фиксированном знаменателе b_j подходят дроби с числителем, $a_i < b_j \cdot m$. При переходе к $b'_j > b_j$ старые значения a_i также подходят и, возможно, появляются новые. Получаем следующий код:

```
i = 0;
for (int j = 0; j < n; j++) {
    while (i < n && a[i] < b[j] * m) {
        i++;
    }
    d += i;
}
```

Если $d < c_k$, то $f > m$, иначе $f \leqslant m$.

Завершив двоичный поиск, мы получаем вещественное число f, осталось выяснить, какой дробью это значение реализуется. Следует отметить, что f найдено не точно, но достаточно близко, если отрезок двоичного поиска сужен менее чем до 10^{-12}.

Чтобы найти подходящую дробь, возьмем минимальное значение $a_i \,/\, b_j \geqslant f$ и максимальное значение $a_i \,/\, b_j < f$. Проверим их оба и выберем то, которое подходит.

Время ответа на один запрос $O(\log(1/\varepsilon) \cdot n)$, на все запросы $O(\log(1/\varepsilon) \cdot n \cdot q)$, тут играет роль ограничение $n \cdot q \leqslant 10^5$.

Решение 2. Недостаток предыдущего решения — использование вещественной арифметики. Рассмотрим полностью целочисленное решение на основе классического *алгоритма Хоара поиска k-й порядковой статистики* в массиве.

Напомним основную идею алгоритма: как в алгоритме быстрой сортировки, выбираем разделяющий элемент x и делим массив на две части: меньшие или равные x и большие x. После этого алгоритм быстрой сортировки запускается рекурсивно от обеих половин массива, а алгоритм поиска k-й порядковой статистики — только от той части, где лежит ответ.

В нашей задаче мы не можем в явном виде хранить массив. Но снова посмотрим на дроби с фиксированным знаменателем. Обратим внимание, что подходящие числители ещё оставшихся в рассмотрении дробей с этим знаменателем образуют отрезок отсортированного массива a. Будем хранить границы этого отрезка для каждого знаменателя и уточнять его границы с помощью

двоичного поиска. В качестве x можно взять случайным образом выбранную дробь из остающихся в рассмотрении.

Алгоритм работает за $O(n \log^2 n)$ (один log от двоичного поиска для разделения каждого массива, второй — от глубины рекурсии).

Мы снова получили оценку времени работы только для одного запроса, но умножая на число запросов и учитывая ограничение $n \cdot q \leqslant 10^5$, получаем приемлемое время работы.

Задача 7. Оптические каналы связи

Автор задачи : *Станкевич А.*
Разработчик : *Жюри ЦПМК.*
Разбор задачи : *Жюри ЦПМК.*

Задачу можно сформулировать так: требуется удалить часть рёбер из дерева так, чтобы степень каждой вершины была не больше k. При этом требуется оставить как можно больше рёбер, а при равном количестве оставить ребра с максимальной суммой.

Большинство подзадач этой задачи требуют той или иной степени применения технологии динамического программирования на дереве. Основная идея следующая: состояние — это вершина дерева, для поддерева которой решается задача, а также информация о решении в поддереве, которая необходима для использования решения для этого поддерева снаружи от него.

Подзадачи 1, 2, 3. В этих подзадачах можно реализовать перебор всех рёбер, которые мы оставляем за $2^n \cdot poly(n)$.

Подзадачи 1, 4, 5. В этих подзадачах требуется найти паросочетание в дереве. Невзвешенная задача решается жадным алгоритмом, а для подзадачи 5 надо использовать динамическое программирование. Состояние динамического программирования для поиска максимального взвешенного паросочетания: (u, b), где u — вершина дерева, а b — флаг для определния, можно ли задействовать вершину u в паросочетании с её родителем.

Подзадачи 6 и 7. В этих подзадачах оставшиеся рёбра образуют пути. Каждый путь в дереве состоит из двух частей: вверх и вниз (одна из этих частей может отсутствовать). Используя динамическое программирование, определим состояние (u, b), где u

— вершина дерева, а b — флаг для определения, можно ли продолжить из вершины u путь вверх.

Рассмотрим теперь решение для произвольного k.

Заметим общую тенденцию решения подзадач для $k = 1$ и $k = 2$: флаг b в состоянии динамического программирования показывает, были ли все k рёбер для корня поддерева u уже сохранены в выбранном оптимальном решении для поддерева, либо можно выбрать ребро из u в родителя.

Обобщим это для полного решения, получив состояние — пару (u, b), в которой флаг b показывает, были ли все k рёбер для корня поддерева u сохранены для решения в поддереве.

Как пересчитать значение динамического программирования?

Рассмотрим вершину u. Чтобы посчитать $(u, 1)$ надо выбрать не более $k - 1$, а чтобы посчитать $(u, 0)$ — не более k рёбер в поддерево из корня. Для каждого выбранного ребра uv к значению добавляется $(v, 0)$, а для невыбранного — $(v, 1)$.

Для вычисления значений воспользуемся вспомогательным динамическим программированием, подобным задаче о рюкзаке.

Найдём оптимальное решение $opt[i][j]$, которое получается после просмотра первых i рёбер, причём взяли из них j рёбер. Переход выполняется за $O(1)$, мы либо берём очередное ребро, либо — нет. Вычисление массива opt для вершины, у которой t детей, работает за $O(t \cdot k)$, суммируя по всем вершинам получаем время $O(n \cdot k)$.

Значение основного динамического программирования для состояния $(u, 0)$ равно максимуму по j от 0 до k, а для состояния $(u, 1)$ — от 0 до $k - 1$.

Наконец, сформулируем чётко, что мы храним в качестве значения динамического программирования. Если задача невзвешенная (подзадачи 1, 2, 4, 6, 8, 10, 12, где $w_i = 0$), значение динамического программирования — это количество взятых рёбер. Если же задача взвешенная, то в качестве значения будем хранить пару из количества взятых рёбер и суммарного их веса. Количество рёбер будет первым критерием оптимизации, а их суммарный вес — вторым, при равенстве количества рёбер.

Менее эффективные решения, которые реализуют динамическое программирование за $O(n^2k)$ или $O(nk^2)$ могут проходить подзадачи 8 и/или 9, и 10 и/или 11, соответственно.

Задача 8. Подарки

Автор задачи : *Григорьев С.*
Разработчик : *Жюри ЦПМК.*
Разбор задачи : *Жюри ЦПМК.*

Подзадача 1. Самое наивное решение данной задачи — перебрать все возможные пары l и r, найти k максимальных чисел на отрезке $[l, r]$ массива (например, сортировкой), а затем вычислить сумму всех чисел без k максимумов. Такое решение работает за $O(n^3 \log n)$ и проходит первую группу тестов.

Подзадача 2. Для прохождения второй группы тестов заметим следующее: зафиксируем левую границу отрезка l и будем увеличивать правую границу r. В таком случае надо уметь поддерживать k максимумов (и их сумму) и добавлять по одному числу к отрезку. Такую задачу можно решить при помощи `std::set` с асимптотикой $O(n^2 \log n)$.

Подзадача 4. Займёмся решением задачи для $k = 0$. Нам требуется найти подотрезок исходного массива с максимальной суммой. Посчитаем префиксные суммы $pref_i = a_1 + a_2 + \ldots + a_i$. Будем перебирать правую границу отрезка r, тогда нам нужно выбрать такое l, $l \leqslant r$, что $a_l + \ldots + a_r = pref_r - pref_{l-1}$ было максимальным. Для этого возьмём минимальное значение $pref_i$ среди всех $i < r$, обозначим его за m, тогда ответ для фиксированной границы r — это $pref_r - m$. Итоговая асимптотика — $O(n)$.

В дальнейшем будем считать, что $k \geqslant 1$, так как случай $k = 0$ разобран. Заметим, что ответ всегда $\geqslant 0$, так как можно взять любой отрезок длины k и все подарки отдать Маше.

Подзадача 3. Для третьей подзадачи сделаем следующее: пусть все подарки исходно не помечены. Будем помечать их в порядке возрастания характеристики (если несколько подарков имеют одинаковую характеристику, то их помечаем в любом порядке). В получившемся массиве рассмотрим все отрезки, на которых есть хотя бы k непомеченных подарков, они являются кандидатами для ответа.

Действительно, для любого отрезка настанет момент, когда на нём будут не помечены только k максимумов. В то же время,

если на отрезке не помечены более k максимумов, то его общее удовольствие будет только меньше (это неверно, если какой-то из непомеченных подарков имеет отрицательную характеристику, однако в таком случае все помеченные точно имеют отрицательное общее удовольствие, что меньше 0 и не влияет на ответ).

Пусть зафиксированы помеченные элементы. Решим задачу методом двух указателей: будем перебирать правую границу r, и пусть l — максимальная левая граница, такая что на $[l, r]$ есть хотя бы k непомеченных элементов. При увеличении r граница l не может уменьшиться. Тогда в качестве ответа для фиксированного r надо взять такое l_0, $l_0 \leqslant l$, что $pref_r - pref_{l_0-1}$ — максимально (здесь $pref_i$ обозначает не сумму первых i чисел массива, а сумму помеченных чисел на префиксе i). Это делается аналогично подзадаче 4. Итоговая асимптотика $O(n^2)$ позволяет пройти тесты в третьей группе.

Подзадача 5. Разберём случай $k = 1$. Так же будем помечать элементы от меньшего к большему. Дополнительно поддерживаем `std::set` отрезков из подряд идущих помеченных элементов. Тогда при рассмотрении очередного элемента x мы должны выбрать суффикс с максимальной суммой на отрезке слева от x и префикс с максимальной суммой на отрезке справа от x, а после объединить эти два отрезка. При объединении максимальный префикс и суффикс пересчитываются с помощью вычисленных значений для предыдущих отрезков: обозначим левый отрезок за s, а правый — за t, тогда

$$maxpref_s = \max(maxpref_s, sum_s + x + maxpref_t),$$
$$maxsuf_t = \max(maxsuf_t, sum_t + x + maxsuf_s),$$

где sum_s и sum_t обозначают сумму на отрезках s и t соответственно. Заметим, что это можно реализовать и без помощи сета, для этого в первом элементе отрезка будем хранить информацию о максимальном префиксе и о последнем элементе, а в последнем — информацию о максимальном суффиксе и первом элементе. Эта информация так же пересчитывается при объединении. Асимптотика $O(n \log n)$.

Подзадачи 6 и 7. Перейдём к полному решению задаче: как и ранее будем помечать подарки в порядке возрастания удовольствия. Кандидаты на ответ — отрезки с k непомеченными

элементами. Также будем хранить отрезки подряд идущих помеченных элементов, как в подзадаче 5. Пусть мы зафиксировали, какие k непомеченных элемента лежат на искомом отрезке. Тогда левее самого левого надо взять максимальный суффикс среди помеченных, правее самого правого надо взять максимальный префикс среди помеченных, а между — сумму помеченных.

При рассмотрении очередного элемента x этот элемент превращается из непомеченного в помеченный, следовательно, появляются новые возможности выбрать k непомеченных элементов для отрезка. Таких возможностей не более $k+1$ варианта. Переберём их все и обновим ответ. Чтобы уметь для непомеченных элементов сдвигаться вперёд/назад на k шагов от заданного, будем поддерживать их в связном списке. Решения с `std::set` или другими более тяжеловесными структурами данных проходят только подзадачу 6.

Итоговая асимптотика $O(nk + n \log n)$.

Задача 9. Действия с дробями

Автор задачи : *Киндер М.И.*
Разработчик : *Киндер М.И.*
Разбор задачи : *Киндер М.И.*

Обозначим искомую дробь через x/y, где x и y — натуральные взаимно простые числа. Разделим её на каждую из данных дробей:

$$\frac{x}{y} : \frac{a_i}{b_i} = \frac{x \cdot b_i}{y \cdot a_i}, \qquad 1 \leqslant i \leqslant n.$$

Эти дроби должны быть целыми числами, поэтому $x \cdot b_i$ делится на a_i и на y.

Поскольку числа a_i и b_i — взаимно простые, отсюда следует, что x делится на a_i. Другими словами, x является общим кратным чисел a_i $(1 \leqslant i \leqslant n)$.

Поскольку x и y — взаимно простые числа, b_i должно делиться на y. Значит, y является общим делителем чисел b_i $(1 \leqslant i \leqslant n)$.

Искомая дробь x/y будет наименьшей, если её числитель — наименьший, а знаменатель — наибольший из всех возможных.

Следовательно,

$$x = \text{НОК}(a_1, a_2, \ldots, a_n), \qquad y = \text{НОД}(b_1, b_2, \ldots, b_n).$$

Таким образом, для нахождения наименьшей положительной дроби x/y нужно в цикле вычислить $(n-1)$ наибольших общих делителей и столько же наименьших общих кратных.

Примерный код на C++:

```
long long gcd(long long x, long long y);
{     if (y==0) return(x);
      else return (gcd(y, x % y));  }

long long lcm(long long x, long long y);
{     return (x / gcd(x, y) * y);  }

int main();
.......................
long long l=a[0];
long long g=b[0];
for (int i = 1; i < n; ++i)
{     l = lcm(l, a[i]);
      g = gcd (g, b[i]);
}
cout << l << ' ' << g;
```

Задача 10. Гирлянда

Автор задачи :	*Валеев И.М.*
Разработчик :	*Валеев И.М.*
Разбор задачи :	*Валеев И.М.*

Подзадача 1. Сначала рассмотрим случай $1 \leqslant n \leqslant 4$. Заметим, что если красных лампочек больше чем n, то это не добавляет новых разноцветных гирлянд, то есть если $b > n$, то можно полагать, что $b = n$. Аналогично и для r. Произведём перебор вариантов и запишем результаты в программе для решения первой подгруппы.

Подзадача 2. Решение задачи в этом случае можно получить с помощью рекурсивного перебора вариантов за $O(t \cdot 2^n)$ операций. Другой способ — использовать битовый перебор вариантов за $O(t \cdot n \cdot 2^n)$ операций.

Подзадача 3. Пусть $b + r = n$. Представим число n в виде суммы n единиц. Выберем среди них b штук, причём порядок этих единиц не важен. Значит, ответом в задаче будет количество сочетаний из n по b, то есть необходимо подсчитать $C_n^b = \frac{n!}{b! \cdot (n-b)!}$. Сложность этого решения — $O(n \cdot t)$.

Подзадача 4. Пусть $b+r > n$. Будем перебирать i от 0 до b, проверяя каждый раз условие $n - i \leqslant r$. В каждом таком случае добавляем к ответу слагаемое C_n^i. Сложность — $O(t \cdot n^2)$.

Подзадача 5. Приведём два способа полного решения задачи. Для этого можно использовать мемоизацию или динамическое программирование. Например, после вычисления каждого слагаемого C_n^k сохраняем это значение. избавляясь от необходимости пересчитывать его несколько раз. Другой способ — заранее подсчитать значения C_n^k, например, с помощью формулы динамического перехода

$$C_n^k = C_{n-1}^{k-1} + C_n^{k-1}.$$

В качестве базы используем начальные значения $C_i^i = 1$ и $C_i^0 = 1$ $(0 \leqslant i \leqslant n)$. Итоговая сложность такого решения — $O(t \cdot n + n^2)$.

Муниципальный этап, 2022-2023

Задача 1. Все на олимпиаду!

Автор задачи : *Фольклор.*
Разработчик : *Киндер М.И.*
Разбор задачи : *Киндер М.И.*

Основные темы задачи:

- *простая логика;*
- *комбинаторика;*
- *максимумы и минимумы.*

Как видно из условия задачи, *наибольшее* число школьников равно $\min(a, b)$.

Для нахождения *наименьшего* возможного числа претендентов подсчитаем количество школьников, которые посещают одновременно оба кружка. Пусть A и B — множество школьников кружка по математике и физике соответственно, тогда

$$|A \cap B| = |A| + |B| - |A \cup B| \geqslant a + b - n.$$

(Здесь $|A|$ — количество элементов в множестве A.) Если $a + b - n < 0$, это число следует считать равным 0. Другими словами, наименьшее возможное число участников равно $\max(0, a + b - n)$.

Приведём основную часть кода на языке C++:

```
int min (int a, int b);
{   return (a < b) ? a : b ;   }
int max (int a, int b);
{   return (a > b) ? a : b ;   }

int main();
    int n, a, b;
    cin >> n;
    cin >> a >> b;
    cout << max(0, a+b-n) << ' ';
    cout << min(a, b);
    return 0;
}
```

Задача 2. Секретное донесение

Автор задачи : *Фольклор.*
Разработчик : *Киндер М.И.*
Разбор задачи : *Киндер М.И.*

Основные темы задачи:

- *сортировка;*
- *алгебраические преобразования;*
- *логика;*

Разберём сначала случай $k = 1$ (подзадачи 1–3). Отсортируем все числа массива $t[]$ в порядке возрастания, так что после сортировки получим массив значений $t[1]$, $t[2]$, $\ldots, t[n]$, причём $t[1] < t[2] < \ldots < t[n]$. Тогда наименьшее число $t[1]$ в этом наборе соответствует сумме из одного слагаемого, то есть равно первому числу $a[1]$ исходной последовательности. Элемент $t[2]$ получается как сумма из двух слагаемых исходного набора, то есть равен $a[1] + a[2]$. Значит, разность $t[2] - t[1]$ даёт второе число $a[2]$ исходного набора. Аналогичным образом, доказываем, что $t[i] - t[i-1] = a[i]$ для всех i от 2 до n.

Разберём ситуацию в общем случае. Предположим, что к массиву $a[]$ применили процедуру шифрования k раз. Тогда с помощью описанного выше алгоритма *дешифровки* можно восстановить массив, из которого получился набор $t[]$, а из него, в свою очередь, восстановить последовательность на предыдущем шаге, и так далее.

Таким образом, для получения исходного массива $a[]$ нужно k раз применить процедуру дешифрования к отсортированному набору чисел $t[]$.

Задача 3. Суперфакториальная СС

Автор задачи : *Фольклор.*
Разработчик : *Киндер М.И.*
Разбор задачи : *Киндер М.И.*

Основные темы задачи:

- *системы счисления;*
- *факториальная система;*

Подзадача 1. Если знаменатель дроби p/q не превосходит 5, то факториальная запись содержит не более пяти цифр и задачу можно решить несложным перебором значений. Для этого с помощью нескольких циклов подберём значения цифр a_1, a_2, a_3, a_4 и a_5.

Подзадачи 2 и 3. Предварительно вычислим значения факториалов $n!$ для всех чисел n от 1 до 20. Сначала найдём целую часть данного числа $a_1 = p\,/\,q$ — это и будет первой цифрой в его факториальной записи. Затем сравним $p\,/\,q$ с числом $1/2!$. Если разность этих чисел положительна, вторая цифра a_2 будет равна 1, иначе — 0. Затем рассмотрим разность $s = p/q - a_1 - a_2/2!$, по значению которой сможем определить значение a_3. Для этого умножим s на 3! и вычислим целую часть полученного выражения, и так далее. Это решение проходит тесты второй и третьей группы.

Подзадача 4. Полное решение задачи не требует дополнительных идей, связанных с предварительным подсчётом значений факториалов или длинной арифметики.

Первая цифра a_1 в суперфакториальной записи совпадает с целой частью данной дроби $p\,/\,q$, то есть равна результату целочисленного деления p на q. (Действительно, поскольку $a_k \leqslant k-1$, $k \geqslant 2$, сумма всех слагаемых в суперфакториальной записи будет не больше

$$a_1 + \frac{1}{2!} + \frac{2}{3!} + \ldots + \frac{n-1}{n!}\,.$$

Каждую дробь в правой части можно представить в виде $\frac{k-1}{k!} = \frac{k}{k!} - \frac{1}{k!} = \frac{1}{(k-1)!} - \frac{1}{(k)!}$, и значит, $\frac{p}{q} - a_1 \leqslant 1 - \frac{1}{n!} < 1$, то есть целое a_1 — это целая часть числа $p\,/\,q$.) Для вычисления следующей цифры a_2 умножим разность $s_1 = p\,/\,q - a_1$ на 2:

$$2 \cdot \left(\frac{p}{q} - a_1\right) = a_2 + \frac{a_3}{3} + \frac{a_4}{3 \cdot 4} + \ldots + \frac{2 \cdot a_{n-1}}{n!}\,.$$

Сумма дробей в правой части также меньше 1, поэтому число a_2 равно целой части $2s_1$. Для вычисления следующей цифры a_3 умножаем разность $s_2 = 2s_1 - a_2$ на 3:

$$3 \cdot (2s_1 - a_2) = a_3 + \frac{a_4}{4} + \ldots + \frac{2 \cdot 3 \cdot a_{n-1}}{n!}\,,$$

и значит, a_3 совпадает с целой частью $3s_2$, и так далее.

Приведём основную часть кода на языке C++:

```
int p, q;
cin >> p >> q;

long long a = p;
int k = 1;
while (a > 0)  {
    long long fss = a / q;
    cout << fss << " ";
    k++;
    a = k*(a - q*fss);
}
```

Замечание. Указанную систему счисления также называют *фактор-адической.*

Задача 4. Мини-Тетрис 2.0

Автор задачи :	*Фольклор.*
Разработчик :	*Киндер М.И.*
Разбор задачи :	*Киндер М.И.*

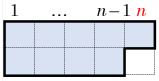
Основные темы задачи:

- *динамическое программирование по профилю;*
- *рекурсия;*
- *бинарное возведение в степень.*

Решение основано на идее динамического программирования по профилю. Основная трудность — получение рекуррентного соотношения для подсчета числа способов замощения и техническая реализация подсчёта.

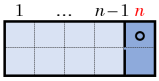
Пусть $A(n)$ — искомое количество способов замощения полосы $2 \times n$. Для вывода рекуррентного соотношения понадобится ещё значение $B(n)$ — количество способов замощения полосы $2 \times n$ с удаленной угловой клеткой (рис. 15):

Рассмотрим столбец с номером n. Его верхняя клетка накрыта «вертикальной» или «горизонтальной» плиткой 1×2, или плиткой 1×1.



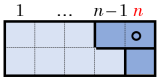
B(n) способов

Рис. 15



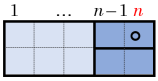
A(n−1) способов

Рис. 16



B(n−1) способов

Рис. 17



A(n−2) способов

Рис. 18

1 случай. Верхняя клетка последнего столбца накрыта «вертикальной» плиткой 1×2 (рис. 16). Число способов замощения оставшейся части полосы равно $A(n-1)$.

2 случай. Верхняя клетка последнего столбца накрыта «горизонтальной» плиткой 1×2. В этом случае нижняя клетка последнего столбца накрыта или плиткой 1×1 (рис. 17), или «горизонтальной» плиткой 1×2 (рис. 18). Число способов теперь равно $B(n-1)+A(n-2)$.

3 случай. Верхняя клетка последнего столбца накрыта плиткой 1×1. Тогда нижняя клетка последнего столбца накрыта или плиткой 1×1, или «горизонтальной» плиткой 1×2. Число способов равно $A(n-1)+B(n-1)$.

Итак, *общее количество способов замощения полосы* $2\times n$ *с помощью плиток* 1×2 *и* 1×1 *равно*

$$A(n)=2A(n-1)+A(n-2)+2B(n-1),$$

причём $A(0)=1, A(1)=2, B(1)=1$. Аналогичным образом получаем соотношение для числа $B(n)$ способов покрытия полосы с удалённой угловой клеткой: $B(n)=A(n-1)+B(n-1)$. Эту систему равенств можно упростить, например, так. Перепишем первое равенство:

$$A(n)=A(n-2)+2\left(A(n-1)+B(n-1)\right)=A(n-2)+2B(n),$$

и значит, $2B(n)=A(n)-A(n-2)$. Тогда $2B(n-1)=A(n-1)-A(n-3)$. Подставим это соотношение в первое равенство:

$$A(n)=3A(n-1)+A(n-2)-A(n-3),$$

причём $A(0)=1, A(1)=2, A(2)=7$ (пример из условия). Это линейное рекуррентное соотношение. Значение $A(n)$ легко находит-

ся в цикле. Сложность алгоритма — $O(n)$. Это решение проходит первые три группы тестов (до $n \leqslant 10^6$).

Для полного решения необходима более тонкая реализация алгоритма вычисления чисел $A(n)$. Например, можно воспользоваться бинарным возведением в степень. Более подробно об этом можно почитать, например, в статье

https://e-maxx.ru/algo/binary_pow

Задача 5. Идеальные наборы

Автор задачи : *Киндер М.И.*
Разработчик : *Киндер М.И.*
Разбор задачи : *Киндер М.И.*

Основные темы задачи:

- *теория чисел;*
- *комбинаторика;*
- *генерация перестановок.*

В каждом идеальном наборе содержится по крайней мере одна гиря весом 1 грамм.

Предположим, что в наборе содержится $x_1 - 1$ гирь весом 1 грамм. Тогда любой вес, меньший x_1, взвешивается единственным образом, и значит, x_1 будет следующей гирей. Пусть теперь в наборе будет $x_2 - 1$ гирь весом x_1 грамм. Тогда любой вес от 1 до $(x_1 - 1) + x_1(x_2 - 1) = x_1 \cdot x_2 - 1$ можно взвесить единственным образом с помощью гирь 1 и x_1 граммов, и значит, следующей в идеальном наборе будет гиря массой x_1x_2 граммов. Продолжая таким образом, приходим к идеальному набору вида

$$\{\,(x_1-1)\cdot 1,\ (x_2-1)\cdot x_1,\ (x_3-1)\cdot x_1x_2,\ \ldots,\ (x_s-1)\cdot x_1x_2\ldots x_{s-1}\,\}.$$

Вес всех гирь, входящих в идеальный набор, равен n, поэтому

$$n = (x_1 - 1) + (x_2 - 1)x_1 + (x_3 - 1)x_1x_2 + \ldots + (x_s - 1)x_1x_2\ldots x_{s-1} = \\ = x_1x_2\ldots x_{s-1}x_s - 1,$$

то есть $x_1, x_2, \ldots, x_s$ — делители числа $n + 1$, причем все $x_i > 1$.

Общее число гирь в идеальном наборе равно

$$k = (x_1 - 1) + (x_2 - 1) + \ldots + (x_s - 1) = \sum_{i=1}^{s} x_i - s.$$

Когда идеальный набор содержит наименьшее число гирь? Только в случае, когда все x_i — простые числа. Если $x_i = a \cdot b$ — составное, то соответствующее слагаемое в сумме равно $x_i - 1 = a \cdot b - 1 > (a - 1) + (b - 1)$, то есть $ab + 1 > a + b$, и значит, количество гирь в этом наборе можно уменьшить.

Реализация алгоритма состоит из трёх шагов:

1. *Разложение числа* $n + 1 = p_1^{k_1} p_2^{k_2} \ldots p_s^{k_s}$ *на простые множители* p_i.

2. *Подсчет количества идеальных наборов минимальной длины.*

3. *Генерация всех возможных упорядоченных перестановок показателей* $(k_1, k_2, \ldots, k_s)$ *простых множителей и вывод идеальных наборов минимальной длины.*

Количество наборов равно количеству *упорядоченных* разложений числа $n+1$ на простые множители. Например, для $n+1 = 12$ будет три упорядоченных разложения:

$$12 = 2 \cdot 2 \cdot 3 = 2 \cdot 3 \cdot 2 = 3 \cdot 2 \cdot 2.$$

(Для лексикографического вывода идеальных наборов множители в этом разложении лучше сразу записать в невозрастающем порядке: $12 = 3 \cdot 2 \cdot 2$, т.е. $3 - 1 = 2$ гири в 1 грамм, $2 - 1 = 1$ гиря в 3 грамма и $2 - 1 = 1$ гиря в $2 \cdot 3 = 6$ граммов. Число всех гирь:

$$k = (3 - 1) + (2 - 1) + (2 - 1) = 4.$$

Если простые множители в разложении $n+1$ повторяются, то есть $n + 1 = p_1^{k_1} p_2^{k_2} \ldots p_s^{k_s}$, приходим к стандартной задаче генерации в лексикографическом порядке наборов вида $(k_1, k_2, \ldots, k_s)$. Для этого нужно сгенерировать все перестановки сомножителей с повторениями, при этом количество идеальных наборов для веса n находится по формуле для полиномиальных коэффициентов

$$\frac{(k_1 + k_2 + \ldots + k_s)!}{k_1!\, k_2! \ldots k_s!}.$$

Замечание. В энциклопедии последовательностей количество упорядоченных факторизаций числа n вычисляется так: oeis.org/A074206.

Региональный этап, 2022-2023

Задача 1. Разделение прямоугольника

Автор задачи : *Станкевич А.*
Разработчик : *Жюри ЦПМК.*
Разбор задачи: *Жюри ЦПМК.*

Пусть было проведено x горизонтальных и y вертикальных разрезов $(0 \leqslant x < a; 0 \leqslant y < b)$.

Подзадача 1. Так как по условию подзадачи $a = 1$, то единственно возможное количество горизонтальных разрезов $x = 0$. Тогда количество вертикальных разрезов $y = k$. Остается проверить: если выполняются условия $k < b$ и $m = k + 1$, то можно провести $x = 0$ горизонтальных и $y = k$ вертикальных разрезов так, чтобы получилось ровно $m = k + 1$ частей, в противном случае поле нельзя разрезать требуемым образом. Асимптотика такого решения — $O(1)$.

Подзадачи 2 и 3. Переберём возможное количество горизонтальных разрезов x $(0 \leqslant x \leqslant k)$, по возрастанию. Вычислим количество вертикальных разрезов $y = k - x$. Заметим, что если мы провели ровно x горизонтальных и ровно y вертикальных разрезов, то поле разделилось на $p = (x+1) \cdot (y+1)$ частей. Остается проверить: если $p = m$, то поле можно разрезать требуемым образом и мы нашли ответ $\{x, y\}$. Если для всех возможных x верно, что $p \neq m$, то поле нельзя разрезать требуемым образом. Асимптотика такого решения — $O(k)$.

Подзадача 4. Аналогично предыдущей подзадаче займёмся перебором возможного количества горизонтальных разрезов x $(0 \leqslant x \leqslant k)$. Будем делать это с учетом того, что число m должно делиться на число $x + 1$. Для этого нужно перебирать все делители числа m: пусть d — делитель числа m, тогда $x = d - 1$. Асимптотика такого решения — $O(\sqrt{m})$, так как чтобы перебрать все делители d числа m, достаточно перебирать их до $\sqrt{m}$.

Подзадача 5. Составим следующие уравнения:

$$\begin{cases} x + y = k, \\ (x + 1)(y + 1) = m. \end{cases}$$

Выразим переменную $y = k - x$ из первого уравнения и подставим во второе, получим $(x + 1)(k - x + 1) = m$. Раскрывая скобки, приходим к квадратному уравнению относительно x:

$$x^2 - kx + (m - k - 1) = 0.$$

Если дискриминант $D = k^2 + 4(k + 1 - m) = (k + 2)^2 - 4m$ не является квадратом некоторого целого числа q ($q^2 = D$), то задача не имеет решений, и значит, разрезать клетчатое поле нельзя. Если же число $q = \sqrt{D}$ – целое, получим два решения квадратного уравнения x_1 и x_2:

$$x_1 = \frac{k + \sqrt{D}}{2} = \frac{k + q}{2}, \quad y_1 = k - x_1;$$
$$x_2 = \frac{k - \sqrt{D}}{2} = \frac{k - q}{2}, \quad y_2 = k - x_2.$$

Остаётся проверить, какое из решений $\{x_1, y_1\}$, $\{x_2, y_2\}$ подходит под ограничения $0 \leqslant x < a$ и $0 \leqslant y < b$, x, $y \in \mathbb{Z}$ и включить его в ответ. В случае, если под ограничение подходят оба решения, необходимо выбрать в ответ то, в котором количество горизонтальных разрезов x_i минимально.

Асимптотика такого решения — $O(1)$.

Задача 2. Произведение Фибоначчи

Автор задачи : *Станкевич А.*
Разработчик : *Жюри ЦПМК.*
Разбор задачи : *Жюри ЦПМК.*

Сделаем несколько ключевых наблюдений.

Во-первых, числа Фибоначчи растут с экспоненциальной скоростью. Существует лишь 85 чисел Фибоначчи, больших 1 и не превышающих 10^{18}.

Во-вторых, числа Фибоначчи имеют достаточно мало общих делителей. На самом деле можно доказать, что если нумеровать числа Фибоначчи с 1, считая $F_1 = 1$, $F_2 = 1$, $F_3 = 2$, ... , то $\text{НОД}(F_a, F_b) = F_{\text{НОД}(a,b)}$.

Подзадача 1. Все числа Фибоначчи, не превышающие 100, которые нас интересуют: 2, 3, 5, 8, 13, 21, 34, 55, 89. Можно либо в ручную разобрать все варианты, либо применить динамическое программирование, аналогичное решению подзадачи 2.

Подзадача 2. Будем использовать динамическое программирование. Обозначим через $d[n][k]$ количество способов представить n в виде произведения чисел Фибоначчи с номерами не больше k. Тогда

$$d[n][k] = \sum_{\substack{i \leqslant k; \\ n \% F_i = 0}} d[n/F_i][i].$$

Используя эту формулу, можно получить ответ за $O(nk)$, где k — количество чисел Фибоначчи, не превышающих n (как мы уже упоминали, оно небольшое).

Подзадача 3. Заметим, что лишь три числа Фибоначчи являются степенями двойки: 1, 2 и 8. Это можно доказать, либо сгенерировать все числа Фибоначчи до 10^{18} и убедиться, что других степеней двойки среди них нет. Значит, количество способов представить число 2^k в виде произведения чисел Фибоначчи зависит от количества чисел 8 в произведении, их может быть не больше $\lfloor k/3 \rfloor$. Таким образом, ответ $\lfloor k/3 \rfloor + 1$.

Подзадача 4. Заметим, что в динамическом программировании, которое решает подзадачу 2, нас интересуют только значения, которые соответствуют делителям n. Делители n можно найти за время $O(\sqrt{n})$; далее, для хранения значений динамики можно воспользоваться, например, `std::map`. Таким образом, время работы получается $O(\sqrt{n} + dk)$, где d — количество делителей n (оно, очевидно, не больше $2\sqrt{n}$, а на самом деле для чисел до 10^9 не превышает 1400).

Подзадача 5. Удивительно, но на первый взгляд менее эффективный чем динамическое программирование, метод рекурсивного перебора полностью решает нашу задачу для достаточно больших n. Используя интуитивное понимание, что количество способов не слишком большое, напишем рекурсивную функцию подсчёта числа произведений:

```
long long bt(long long n, int p) {
  if (n == 1) return 1;
  if (p >= fib.size()) return 0;
  if (n < fib[p]) return 0;

  long long res = bt(n, p + 1);
  if (n % fib[p] == 0) {
    res += bt(n / fib[p], p);
  }
  return res;
}
```

Задача 3. Робот-пылесос

Автор задачи : *Рогачева Е.*
Разработчик : *Жюри ЦПМК.*
Разбор задачи : *Жюри ЦПМК.*

Подзадачи 1 и 2. Для решения первых двух подзадач достаточно завести множество, хранящее все клетки, посещённые роботом. Для этого можно использовать структуру данных `std::set`. Каждое перемещение робота на a_i можно представить как a_i перемещений на 1 в соответствующем направлении, и после каждого перемещения добавлять все k^2 покрытых роботом клеток в множество. Это решение работает за $O(nk^2 \sum a_i)$.

Подзадача 3. В третьей подзадаче решение для подзадач 1 и 2 работает за время $O(nk^2)$ и не проходит по времени. Однако можно заметить, что при каждом перемещении робота на 1 в любом направлении появляется не более k новых клеток, которые необходимо добавить в множество — полоса $1 \times k$ или $k \times 1$, в направлении которой робот сдвинулся. Таким образом, добавляя в множество только потенциально новые клетки, получаем решение третьей подзадачи, работающее за время $O(nk)$.

Подзадача 4. В этой подзадаче робот каждый раз перемещается на $a_i = k$. Разобьём плоскость на квадраты размера $k \times k$ и будем воспринимать каждый квадрат как самостоятельную клетку, «сжав» плоскость в k раз по каждой координате. На такой плоскости достаточно решить задачу о роботе размера 1×1, кото-

рый каждый раз перемещается на расстояние, равное $a_i = 1$. Для такой задачи также подойдёт «наивное» решение с множеством всех посещенных клеток. В конце достаточно умножить полученный ответ на k^2, потому что каждая «клетка» новой плоскости на самом деле состоит из k^2 клеток исходной плоскости.

Подзадача 5. В пятой подзадаче $k = 1$, но робот может перемещаться на большие расстояния. Смоделируем перемещения робота и выпишем отдельно горизонтальные и вертикальные «полосы» перемещений. Сгруппируем все вертикальные полосы с одинаковой X-координатой вместе, и сгруппируем все горизонтальные полосы с одинаковой Y-координатой вместе.

Количество посещённых клеток можно посчитать как объединение всех полос минус количество их пересечений. Объединить все полосы с общей координатой можно с помощью сортировки и одного линейного прохода (например, $[l_1, r_1]$ и $[l_2, r_2]$ с $l_1 \leqslant l_2 \leqslant r_1$ объединяются в $[l_1, r_2]$, и так далее).

Найти все пересечения полос в данной подзадаче можно за время $O(n^2)$, вручную проверив на пересечение каждую пару из вертикальной и горизонтальной полос. Полосы $[l, r]$ на координате y и $[d, u]$ на координате x пересекаются, если $l \leqslant x \leqslant r$ и $d \leqslant y \leqslant u$.

Подзадача 6. В подзадаче 6 перемещения робота теперь не разбиваются на сравнительно небольшое количество полос ширины или высоты 1. В этой подзадаче необходимо сделать ключевое наблюдение для полного решения задачи. При каждом перемещении клетки, посещаемые роботом, образуют прямоугольник. Например, если левый-нижний угол робота находится в точке с координатами $(0, 0)$, и совершается перемещение на a_i вверх, робот посетит все клетки в прямоугольнике с углами в $(0, 0)$ и $(k, a_i + k)$. Для того, чтобы найти количество убранных клеток, достаточно найти площадь объединения таких прямоугольников.

Это можно сделать с помощью метода сканирующей прямой за время $O(n^2)$. Сожмём координаты. Заведём для каждого прямоугольника события его «начала» (левая сторона) и «конца» (правая сторона), отсортируем их по X-координате, и обработаем по очереди.

В полосе между двумя событиями на X-координатах x_1 и x_2

прямоугольники покрывают одно и то же множество Y-координат клеток, так что достаточно добавить к ответу $(x_2 - x_1) \cdot w$, где w равно высоте объединения прямоугольников, покрывающих эту полосу. После каждого события значение w можно пересчитать за время $O(n)$.

Для решения последних двух подзадач потребуется ускорить решения подзадач 5 и 6, соответственно.

Подзадача 7. Научимся в решении подзадачи 5 для каждой вертикальной полосы находить количество пересекающих её горизонтальных за время $O(\log n)$. Это можно сделать с помощью дерева отрезков. Используем метод сканирующей прямой, прямая перемещается по увеличению $X-$координаты, а в Y-координатах, соответствующих горизонтальным полосам, храним 0, если сейчас в соответствующей координате есть полоса, либо 0, если её нет. Тогда события — это начала и концы горизонтальных полос, а также вертикальные полосы. Для события вертикальной полосы необходимо найти сумму в дереве отрезков в соответствующем ей отрезке Y-координат.

Подзадача 8. Полное решение заключается в объединении прямоугольников на плоскости. Это достаточно классическая задача, которая имеет много аналогичных решений с использованием сканирующей прямой и дерева отрезков.

Приведём одно из возможных решений. Будем хранить в дереве отрезков для каждой Y-координаты, сколько прямоугольников её покрывают, и возвращать количество Y-координат, покрытых хотя бы одним прямоугольником. Альтернативный подход – использовать дерево отрезков на минимум и суммарный вес минимумов; тогда если минимум равен 0, то это означает, что полоса не покрыта прямоугольниками. Так можно посчитать суммарную непокрытую площадь в некотором достаточно большом объемлющем прямоугольнике, а далее вычесть её из его площади.

Решение с деревом отрезков и сканирующей прямой работает за $O(n \log n)$.

Замечание. Отметим, что координаты, в которых может оказываться робот, могут достигать значений порядка $10^5 \cdot 10^9 = 10^{14}$, но площадь каждого посещенного прямоугольника не больше $10^4 \cdot 10^9 = 10^{13}$. Значит, их общая площадь не превышает $10^5 \cdot 10^4 \cdot 10^9 = 10^{18}$ и помещается в 64-битном типе данных. Однако следует с осторожностью относиться к промежуточным значениям, чтобы не произошло переполнения 64-битного типа данных.

Задача 4. Разноцветные точки

Автор задачи : *Станкевич А.*
Разработчик : *Жюри ЦПМК.*
Разбор задачи : *Жюри ЦПМК.*

Для начала рассмотрим идеи, общие для некоторых подзадач.

Во-первых, рассмотрим ориентированный граф, в котором сопоставим вершину каждой паре (i, j) $(i \neq j)$, где i — номер начальной точки, а j — номер следующей точки. Таким образом, получится $O(n^2)$ вершин. Из каждой вершины графа, соответствующей паре (i, j), проведем единственное ребро в вершину, соответствующую паре (j, k), где k — номер прицельной точки для начальной точки i и следующей точки j. Таким образом, из каждой вершины графа исходит ровно одно ребро, а в терминах построенного графа процесс, описанный в условии задачи, выглядит как перемещение по ребрам.

Во-вторых, научимся определять цвета точек, пользуясь построенным графом.

Утверждение 1. *Точка i будет окрашена в зелёный цвет тогда и только тогда, когда в построенном графе существует вершина, соответствующая паре (i, j) для некоторого j, которая лежит на некотором цикле.*

Действительно, если такая вершина существует, то вершина i будет зелёной, потому что можно выбрать точку j в качестве следующей, и точка i побывает в статусе начальной бесконечное количество раз. В противном случае не существует такой точки j, что вершина, соответствующая паре (i, j) лежит на каком-то цикле. Так как граф конечен, а процесс продолжается до бесконечности, рано или поздно, перемещаясь по ребрам, мы попадём в какой-либо цикл и будем обходить его до бесконечности. При этом на этом цикле нет вершин, которые соответствуют состояниям, в которых i — начальная точка. Поэтому точка i не будет окрашена в зелёный цвет.

Утверждение 2. *Точка i будет окрашена в синий цвет тогда и только тогда, когда она не была окрашена в зелёный цвет и существуют такие вершины v_1 и v_2, соответствующие парам (i, j_1) и (i, j_2), что вершина v_2 достижима по ребрам из вершины v_1.*

Действительно, если такие вершины существуют, то вершина i будет синей, потому что можно выбрать точку j_1 в качестве следующей, и в момент, когда процесс окажется в вершине v_2 графа, точка i снова станет начальной, а точка j_2 — следующей. В противном случае, так же как и в предыдущем утверждении, процесс рано или поздно дойдёт до некоторого цикла и будет обходить его до бесконечности, не оказавшись снова в состоянии, когда точка i является начальной.

Все остальные вершины будут окрашены в красный цвет.

В зависимости от оптимальности алгоритма построения графа, а также от проверки описанных критериев для определения цветов точек, решение получает различное количество баллов. Теперь рассмотрим подробно решение конкретных подзадач.

Подзадача 1. Рассмотрим построение графа. Так как все точки расположены на одной прямой, это делается достаточно просто. Для начала отсортируем все точки вдоль прямой. Для этого, например, можно упорядочить точки по возрастанию x-координат, а при равенстве — по возрастанию y-координат. Также для каждой точки сохраним её номер в порядке сортировки.

Теперь рассмотрим начальную точку i и следующую точку j. Пусть номер точки i в порядке сортировки равен pos_i, а номер точки j в порядке сортировки равен pos_j. Рассмотрим случай, когда $pos_i < pos_j$, обратный случай рассматривается аналогично. Порядок, в котором точки упорядочены по углу относительно вектора $\overrightarrow{P_iP_j}$, выглядит следующим образом: $pos_j + 1, \ldots, n$, $pos_j - 1, \ldots, 1$ (имеются ввиду номера точек в порядке сортировки). Рассмотрим точку, находящуюся на t-й позиции в этом списке, и проведем соответствующее ребро в графе. Данная часть решения работает за $O(n \log n + n^2)$.

Теперь рассмотрим процесс определения цветов точек. Его можно реализовать наивно, пользуясь критериями, описанными выше.

Для того, чтобы проверить, что точка i является зеленой, переберем номер следующей точки j и проверим, что вершина, соответствующая паре (i, j), лежит на цикле. Для этого будем переходить по ребрам до тех пор, пока не попадем в очередной раз в вершину (i, j), либо не совершим n^2 переходов по ребрам.

Если спустя n^2 переходов по ребрам мы ни разу не оказались в вершине (i, j), то она не лежит на цикле, так как количество вершин в графе меньше, чем n^2.

Для проверки того, что точка i является синей, будем делать аналогичные действия. В этом случае нужно остановить процесс перехода по ребрам, если текущая вершина соответствует некоторой паре (i, k) $(k \neq j)$. Данная часть решения работает за $O(n^4)$.

Подзадача 2. Данная подзадача отличается от предыдущей ограничением на количество точек, поэтому необходимо оптимизировать вторую часть решения.

Как было описано ранее, построенный граф обладает интересным свойством — из каждой вершины исходит ровно одно ребро. Можно заметить, что такие графы выглядят как набор непересекающихся циклов, в которые «врастают» деревья.

Теперь, пользуясь этим наблюдением, выделим все зелёные точки. При помощи обхода в глубину можно найти все циклы в графе. Теперь, если некоторая вершина (i, j) лежит на цикле, покрасим точку i в зеленый цвет. Данная часть решения работает за $O(n^2)$.

Осталось найти все синие точки. Для этого научимся проверять, является ли точка i синей, за $O(n)$, тогда суммарно все проверки будут работать за $O(n^2)$.

Выделим все вершины графа, соответствующие парам (i, j), для фиксированного i. Таких вершин $O(n)$. Теперь нужно быстро проверить, существуют ли две выделенные вершины, одна из которых достижима из другой. Таким вершины не могут лежать на циклах, так как циклы графа мы обработали ранее. Поэтому такие вершины могут лежать только на деревьях. Более того, эти вершины должны лежать в одном и том же дереве, так как деревья не пересекаются друг с другом. Нетрудно понять, что в этом случае одна из вершин должна являться предком другой в дереве.

Таким образом, мы получили следующую классическую задачу: дано множество вершин, нужно определить, существует ли пара вершин, в которой одна из вершин является предком другой. Данная задача решается следующим образом. Запустим обход в глубину от корней всех деревьев и вычислим время входа и время

выхода для каждой вершины. Упорядочим вершины по возрастанию времени входа. Теперь для каждой пары соседних вершин в полученном порядке нужно проверить, является ли одна из них предком другой. Нетрудно понять, что вершина v_1 является предком v_2 тогда и только тогда, когда $tin_{v_1} < tin_{v_2} < tout_{v_2} < tout_{v_1}$, где tin — время входа, а $tout$ — время выхода.

Подзадачи 3–6. Данная подзадача отличается от предыдущих тем, что теперь точки не лежат на одной прямой. Поэтому нужно изменить первую часть решения и оставить без изменений вторую часть решения.

Построим граф следующим образом. Зафиксируем начальную точку i и следующую точку j. Построим векторы из точки j во все остальные точки и отсортируем полученный набор векторов по углу. Также для удобства можно добавить в данный набор вектор $\overrightarrow{P_iP_j}$. Для сортировки векторов по углу можно воспользоваться функцией `atan2`, но для того, чтобы все вычисления производились в целых числах, можно воспользоваться другим способом. При сравнении двух векторов будем сначала сравнивать их координатные четверти. Если у векторов v_1 и v_2 совпадают координатные четверти, то отсортируем их по предикату $[v_1, v_2] > 0$, где $[v_1, v_2]$ — косое произведение. В случае, если косое произведение равно нулю, отсортируем векторы по квадрату их длины.

После сортировки найдем вектор $\overrightarrow{P_iP_j}$ в полученном списке. Пусть позиция этого вектора равна pos. Тогда вектор, направленный в прицельную точку, будет иметь номер $(pos + t) \pmod n$ в полученном списке. Данный способ построения графа работает за $O(n^3 \log n)$.

В зависимости от того, насколько оптимально реализована вторая часть решения, данное решение может проходить различный набор подзадач.

Подзадача 7. В данной подзадаче все точки являются вершинами выпуклого многоугольника. Оказывается, что в таком случае все точки являются зелёными.

Для доказательства этого факта достаточно заметить, то для начальной точки i и следующей точки j прицельной точкой будет такая точка k, которая является t-й в порядке обхода многоугольника, начиная с точки j, вне зависимости от точки i. Значит по-

лучится следующий процесс: $(i, j) \to (j, j+t) \to (j+t, j+2t) \to \ldots$ (номера всех точек следует брать по модулю n). Рано или поздно процесс окажется в состоянии $(j+s\cdot t, j+(s+1)\cdot t)$, где $j+s\cdot t \equiv i \pmod n$. Осталось показать, что всегда можно выбрать точку j так, что данное уравнение имеет решение относительно s. Действительно, $s\cdot t \equiv i-j \pmod n$, поэтому можно выбрать j таким образом, чтобы было верно равенство $i-j \equiv t \pmod n$.

Подзадача 8. Для решения данной подзадачи осталось оптимизировать процесс построения графа. Научимся делать это за $O(n^2 \log n)$. Для этого рассмотрим некоторую следующую точку j. Отсортируем векторы из точки j во все остальные точки по углу при помощи способа, описанного выше. Теперь переберём начальную точку i и научимся быстро находить номер прицельной точки k. Для этого воспользуемся двоичным поиском и найдём в отсортированном по углу списке первый вектор, угол которого больше либо равен, чем угол вектора $\overrightarrow{P_iP_j}$. Пусть номер этого вектора равен pos. Тогда номер вектора, ведущего в прицельную точку, равен $(pos+t) \pmod n$.

Комбинируя эту часть решения с оптимальной второй частью решения, можно решить задачу на 100 баллов. Итоговое решение работает за $O(n^2 \log n)$.

Задача 5. Метрострой

Автор задачи : *Станкевич А.С.*
Разработчик : *Жюри ЦПМК.*
Разбор задачи : *Жюри ЦПМК.*

Подзадача 1. Заметим, что если двигатель всего один, то возможны два случая: необходимая мощность достигается при работе двигателя в первом режиме или во втором. При этом легко понять, какой режим необходим, если сравнить z_1a_1 и p. Таким образом, решение первой подзадачи выглядит так:

```
if p <= z[0]*a[0]:
    print(ceil(p/a))
else:
    print(z + ceil((p-z[0]*a[0])/b)
```

Подзадача 2. Ограничения этой подзадачи позволяли явно перебрать и найти минимальное значение x, удовлетворяющее условию задачи.

```
x = 1
while True:
    q = 0
    for i in range (n):
        if x <= z[i]
            q += a[i] * x
        else:
            q += a[i] * z[i] + b[i] * (x - z[i])
    if q >= p:
        break:
    c += 1
print(x)
```

Подзадача 3. Третья подзадача также предполагала два случая: необходимую мощность можно получить, если двигатели работают в первом режиме, или необходим второй режим.

	Условие	**Ответ**
Первый случай	$\sum_{i=1}^{n} a_i z_i \geqslant p$	$\lceil p \,/\, \sum_{i=1}^{n} a_i \rceil$
Второй случай	$\sum_{i=1}^{n} a_i z_i < p$	$\lceil \left(p - \sum_{i=1}^{n} a_i z_i \right) / \sum_{i=1}^{n} b_i \rceil$

Подзадача 4. Будем считать, что $z_1 \leqslant z_2$ (если это не так, мысленно перенумеруем двигатели). Тогда:

1. либо оба двигателя будут работать в первом режиме ($x \in [0, z_1)$), тогда ответ — $\texttt{ceil}\left(\frac{p}{a_1+a_2}\right)$;
2. либо двигатель 1 будет работать во втором режиме, а двигатель 2 — в первом ($x \in [z_1, z_2)$), тогда ответ —

$$\texttt{ceil}\left(\frac{p - (a_1 + a_2) \cdot z_1}{b_1 + a_2}\right);$$

3. либо оба будут прокладывать туннель во втором режиме ($x \in [\, z_2; \infty)$), ответ в этом случае —

$$\texttt{ceil}\left(\frac{p - (a_1 + a_2) \cdot z_1 - (b_1 + a_2) \cdot (z_2 - z_1)}{b_1 + b_2}\right).$$

Давайте для каждого случая вычислим минимальное значение x и выберем первый случай, в котором x попадает в заданный полуинтервал возможных значений x.

Здесь намеренно не рассматривается отдельно случай, когда $z_1 = z_2$, это возможно так как не существует таких x, которые попадают в полуинтервал $[\, z_1, z_2) = \varnothing$.

Полное решение, 1 способ. Займёмся обобщением решения предыдущей подзадачи на случай $n > 2$. Отсортируем двигатели по z_i. Точно так же у нас будет $n + 1$ непересекающихся полуинтервалов, объединение таких полуинтервалов будет давать числовой луч $[\, 0, +\infty)$.

В каждом интервале $[\, z_i; z_{i+1})$ (считаем, что $z_0 = 0$ и $z_{n+1} = 1$) можно находить минимальное значение x, при котором достигается необходимая мощность, как округлённое вверх число

$$\frac{p - S(i)}{\sum_{i=1}^{j-1} b_i + \sum_{i=j}^{n} a_i}.$$

Здесь $S(i)$ — мощность, которая достигается при $x = z_i$; значение $S(i)$ можно вычислить по формуле

$$\sum_{k=1}^{i-1} (z_k - z_{k-1}) \left(\sum_{i=1}^{k-1} b_i + \sum_{i=k}^{n} a_i\right).$$

Таким образом, в решении нужно последовательно рассматривать интервалы до тех пор, пока не найдем значение x, дающее достаточную мощность и находящееся в этом интервале.

Время работы такого решения $O(n^2)$ или $O(n \log n)$, если немного оптимизировать.

Полное решение, 2 способ. Заметим, что мощность, которую дают двигатели, монотонно возрастает при увеличении x. Поэтому для нахождения минимального значения x можно воспользоваться бинарным поиском, на каждой итерации проверяя, в каком режиме работает каждый из двигателей.

Время работы такого решения $O(n \log n)$.

Задача 6. Красивые последовательности

Автор задачи : *Саблина Р.*
Разработчик : *Жюри ЦПМК.*
Разбор задачи : *Жюри ЦПМК.*

Подзадача 1. Для решения данной подзадачи можно было воспользоваться полным перебором. Для каждой позиции переберём, чему равен элемент (1 или 2), после чего наивно проверим выполнение условия для всех пар индексов. Асимптотика решения составит $O(2^n \cdot n^2)$.

Общая идея решения следующих подзадач заключается в использовании динамического программирования. Пусть мы набрали префикс красивой последовательности, и хотим добавить в его конец элемент $x \in A$. Тогда при переходе нам необходимо и достаточно потребовать, чтобы последнее вхождение x на префиксе было хотя бы на расстоянии $x - 1$ от конца префикса.

Подзадачи 2–4. Определим $\texttt{dp}[i][j]$ как число красивых последовательностей длины i, в которых последнее вхождение числа k было на расстоянии j от конца префикса. Тогда для пересчета динамики достаточно рассмотреть два варианта: мы добавляем элемент 1 или элемент k.

Первый переход возможен всегда, и увеличивает расстояние до последнего вхождения на единицу. Второй переход возможен, если $j \geqslant k-1$, и в результате него расстояние до последнего вхождения k становится равным нулю. Не забудем, что все арифметические операции нужно выполнять по модулю. Таким образом, псевдокод перехода в динамике выглядит так:

```
dp[i + 1][j + 1] += dp[i][j]
if (j >= k - 1) {
    dp[i + 1][0] += dp[i][j]
}
```

Таким образом, получаем динамическое программирование с n^2 состояниями, и $O(1)$ переходов из каждого. Тогда асимптотика такого решения составит $O(n^2)$. Заметим, что во второй подзадаче максимальный ответ небольшой, поэтому число после-

довательностей можно было считать рекурсивно или перебором с отсечениями по ходу рекурсии.

Подзадачи 5–6. Обозначим за C максимальное ограничение на a_i ($C = 5$ в подзадаче 5, и $C = 8$ в подзадаче 6).

Для полного решения задачи обобщим решение предыдущих подзадач. В состоянии динамического программирования будем для каждого числа x от 1 до n хранить расстояние до его последнего вхождения на префиксе, обозначим его за $d[x]$. Заметим следующий факт: если в какой-то момент расстояние для числа x стало больше, чем $x-1$, то мы можем считать, что оно равно $x-1$, так как такого расстояния до предыдущего вхождения достаточно. Таким образом, количество возможных массивов расстояний $d[x]$ не превышает $1 \cdot 2 \cdot \ldots C = C!$.

Переход в данном динамическом программировании выглядит так: переберём, какой элемент $x \in A$ мы добавим к префиксу. Тогда такой переход возможен, если $d[x] \geqslant x - 1$, и в результате перехода происходит следующее: для всех $y \neq x$ их значение $d[y]$ увеличивается на единицу, а значение $d[x]$ становится равным нулю.

Для реализации данного решения нужно хранить отображение из массива расстояний $d[x]$ в количество последовательностей в ассоциативной структуре данных. Например, на языке C++ подойдет «`std::map`», или же «`std::unordered_map`», если реализовать хеш-функцию для массива расстояний.

Таким образом, динамическое программированием даёт решение с $O(n \cdot C!)$ состояниями, причём переход выполняется за $O(C^2)$. Заметим, что на самом деле достижимых состояний значительно меньше, так как, например, в достижимых состояниях все значения $d[x] < x - 1$ различны (для $C = 8$ их количество не превышает 4140). Тогда решение можно ускорить, если не делать переходы из состояний, количество последовательностей для которых равно нулю. В зависимости от эффективности решения и наличия такой оптимизации решения могли проходить только пятую подзадачу.

Задача 7. Камни

Автор задачи : *Новиков В.*
Разработчик : *Жюри ЦПМК.*
Разбор задачи : *Жюри ЦПМК.*

Подзадача 1. Заметим, что на каждой итерации множество камней, покрашенных в белый цвет, является последовательным отрезком. Перебрав стартовую позицию i и двигая отрезок в соответствии с условием задачи будем увеличивать счетчик ответа, если текущая длина отрезка равняется k и p-й камень был последним перекрашенным в белый цвет.

Подзадача 2. Нет смысла заново запускать алгоритм расширения отрезка после каждого запроса. Будем поддерживать массив ответа $ans[p][k]$. Достаточно заранее запустить алгоритм для каждого стартового i и на k-й итерации смотреть на последний добавленный камень p'. В таком случае увеличим $ans[p'][k]$ на единицу и продолжить выполнение алгоритма. Иначе говоря, заранее выполним передвижения отрезков для всех стартовых позиций, такой предподсчёт будет работать за $O(n^2)$, а ответы на запросы за $O(1)$ — достаточно обратиться к нужной ячейке массива ответа.

Подзадача 4. В четвертой подзадаче достаточно заметить, что почти всегда мы двигаем левую границу отрезка (за исключением случая, когда левая граница отрезка достигла конца массива). Простым разбором случаев замечаем, что к ответу добавляется 1, когда $p+k-1 < n$, и в ответ добавляется p, если $p+1 = k$. Первый случай соответствует ситуации, когда отрезок постоянно двигает левую границу, а второй — ситуации, когда мы уже не можем двигать левую границу.

Подзадачи 3, 5 и 6. Для решения следующих подзадач посмотрим на отрезок, который будет после выполнения k операций, если k-м элементом был добавлен элемент p. Тогда этот отрезок имеет вид $[p-k+1, p]$, либо $[p, p+k-1]$. Так как данные случаи симметричны, в разборе будем описывать лишь случай с $[p, p+k-1]$.

Посмотрим на позицию максимума m на данном отрезке.

Рассмотрим три случая:

1. Стартовый камень $i < m$. Иными словами, i лежит левее m. Но мы знаем, что a_m – максимальный элемент отрезка. Из этого следует, что элемент с номером m будет добавлен в расширяющийся отрезок после элемента p. Получаем, что в такой ситуации p никак не может быть добавлен в отрезок k-м.
2. Стартовый камень $i = m$. В этом случае достаточно проверить, что p является вторым максимумом отрезка и то, что $p + k$ будет добавлен позже p (иными словами $a_{p+k} > a_p$). Если данные условия выполняются, то p будет добавлен в отрезок на k-й итерации.
3. Стартовый камень $i > m$. В таком случае, если $a_{p+k} < a_m$, то мы выйдем за границы нашего отрезка $[p, p + k - 1]$ до того, как добавим p в ответ. В случае же, если $a_{p+k} > a_m$, мы всегда добавим a_{p+k-1} в отрезок, а далее будем добавлять элементы со стороны левой границы, так как a_{p+k} будет больше всех элементов отрезка. Остается заметить, что все это будет происходит вне зависимости от выбора i при условии, что $i \in [m + 1, p + k - 1]$.

В зависимости от эффективности метода поиска максимума и второго максимума на всех отрезках длины k, мы получаем решение для различных подзадач.

Для полного решения необходимо использовать вполне эффективную структуру данных для максимума на отрезке, например, дерево отрезков, разреженные таблицы, максимум в окне любым эффективным способом.

Итоговая асимптотика: $O(n \log n + q)$ или $O(n + q)$.

Задача 8. Обыкновенная задача про строки

Автор задачи : *Бабин А.*
Разработчик : *Жюри ЦПМК.*
Разбор задачи : *Жюри ЦПМК.*

Общие замечания. Перед дальнейшими рассуждениями рассмотрим случай, когда длина строки s равна 1. Понятно, что

при таком исходе ответ будет равен 3, так как строки «a», «b» и «c» попарно эквивалентны.

Также активно будет использоваться идея о «сокращении» строки. *Сокращением* строки s будем называть такую строку $\eta(s)$, которая получается при сокращении блоков из одинаковых символов в строке s. Например, $\eta(\text{«abaacccca»}) = \text{«abaca»}$. Нетрудно убедиться, что если s и t эквивалентны, то эквивалентна и пара строк $\eta(s)$, $\eta(t)$.

Количество подстрок «aa» у строки s будем обозначать через k_a, аналогично определим k_b и k_c. Такие подстроки будем в дальнейшем называть повторами символа «a», «b» или «c», соответственно.

Подзадача 1. Пускай строка s имеет длину 2 и больше, а также состоит только из символов «a» и «b». Тогда $\eta(s)$ состоит из чередующихся символов «a» и «b». Рассмотрим несколько случаев:

- Если $\eta(s)$ имеет чётную длину, то среди подстрок длины 2 у нее будут только «ab» и «ba». При этом одна из них встретится на один раз больше чем другая. Таким образом, легко видеть, что $\eta(s)$ эквивалентна только по отношению к самой себе.
- Если $\eta(s)$ имеет нечётную длину, то среди подстрок длины 2 у нее будут только «ab» и «ba», которые встречаются одинаковое количество раз. Таким образом, строке $\eta(s)$ будут эквивалентны две строки такой же длины, у которых также чередуются символы «a» и «b». Например, если $\eta(s) = \text{«ababa»}$, то ей эквивалентны строки «ababa» и «babab».

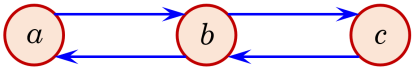
Нам надо посчитать количество строк t, которые эквивалентны s. Заметим, что $\eta(t)$ эквивалентна $\eta(s)$, а также, что для $\eta(s)$ можно найти не более двух эквивалентных строк. Таким образом, можно перебрать $\eta(t)$. Осталось научиться обрабатывать повторы в строке t. (Напомним, что количество повторов в строках t и s должны совпадать, так как они эквивалентны.) Но тогда задача состоит в том, чтобы для заданной строки $w = \eta(t)$ посчитать количество строк t, у которых k_a повторов символа «a» и k_b повторов символа «b». Пусть в строке w символ «a» встречается

m_a раз, а символ «b» — m_b раз. Тогда в строке t будет m_a блоков из символов «a» с суммарной длинной $m_a + k_a$ символов и m_b блоков с суммарной длиной $m_b + k_b$. Количество строк t совпадает с количеством разбиений числа $m_a + k_a$ на m_a натуральных слагаемых и разбиений числа $m_b + k_b$ на m_b слагаемых.

Таким образом, если воспользоваться методом шаров и перегородок, то понятно, что количество подходящих строк t равняется:

$$C_{m_a+k_a-1}^{k_a} \cdot C_{m_b+k_b-1}^{k_b}.$$

Подзадача 2. Решение этой подзадачи похоже на решение предыдущей; основное отличие в том, что количество строк, которые эквивалентны $\eta(s)$, может быть достаточно большим. Поэтому сначала определим, какой вид они могут принимать. Если до этого мы избегали графической (от слова «граф») сущности задачи, то сейчас сделать это уже не получится. Пусть строка «ab» встречается k_{ab} раз, строка «ac» — k_{ac} и так далее. По условию подгруппы $k_{ac} = k_{ca} = 0$. Тогда можно рассмотреть следующий ориентированный граф, состоящий из трёх вершин a, b, c:



Из вершины a в вершину b будет вести k_{ab} кратных рёбер, аналогичное будет верно и для любой другой пары вершин в этом графе. Тогда заметим, что последовательность вершин любого эйлерова пути этого графа будет соответствовать какой-то строке, которая эквивалентна $\eta(s)$. При этом если w эквивалентна $\eta(s)$, то среди символов $w_2, w_3, \ldots, w_{|w|}$ символ «a» встретится k_{ba} раз, символ «b» встретится $k_{ab} + k_{cb}$ раз, а символ «c» — k_{bc} раз. Таким образом, если известен первый символ строки w, то известно также и количество вхождений в него символов «a», «b», «c».

Итак, разберём три случая:

- w_1 = «a». Этот случай имеет место быть, если эйлеров путь может начинаться с вершины a, так происходит, если a входит в строку s хотя бы один раз, а также выполняется s_1 = «a» или $s_1 = s_n$. Легко видеть, что все чётные символы строки w, то есть w_2, w_4, w_6, …, равны «b». Все

нечётные символы, кроме первого и последнего (который может иметь как чётный, так и нечётный номер), можно выбирать свободно, причём на этих позициях «a» встретится $k_{ab} - 1$ раз, а «b» — k_{cb} раз. Количество таких строк w равняется $C^{k_{ab}}_{k_{ab}-1+k_{cb}}$.

- w_1 = «b». Аналогично первому случаю проверяется, что строка w может начинаться с символа «b». Количество подходящих строк w будет равно $C^{k_{ab}}_{k_{ab}+k_{cb}}$.
- w_1 = «c». Разбирается аналогично случаю w_1 = «a».

Теперь можно сформулировать общую схему алгоритма. Для начала мы перебираем первый символ строки w и считаем количество строк w с заданным первым символом, которые эквивалентны строке $\eta(s)$. Для каждой такой строки из заданной группы строк мы, как и в первой подзадаче, легко можем вычислить количество строк t, эквивалентных s, таких что $\eta(t) = w$. Так как в строках из одной группы символы «a», «b» и «c» встречаются одинаковое количество раз, то и соответствующих строк t будет одинаковое количество, таким образом можно воспользоваться правилом умножения.

С помощью предпосчитанных факториалов и обратных факториалов по модулю $10^9 + 7$ за время $O(n)$ можно разобрать каждый случай за время $O(1)$. Таким образом, итоговая асимптотика алгоритма $O(n)$.

Подзадача 3. Если $n \leqslant 13$, то достаточно научиться перебирать всевозможные строки s длиной не больше 13 и для каждой такой строки сохранить массив, состоящий из 9 чисел: количество вхождений каждой из возможной строк длины 2 в строку s. Затем можно посчитать количество вхождений каждого из этих массивов в итоговый список массивов. После вышеописанных действий можно тривиальным образом отвечать на запросы.

Тем не менее хранить и сравнивать массивы недостаточно эффективно. Поэтому все эти массивы можно занумеровать числами от 0 до $2^{21} - 1$ следующим образом. Пусть нам надо получить номер массива $a_1, \ldots, a_9$, тогда заметим, что так как $a_1 + \ldots + a_9$ не превышает 12, то ему можно сопоставить бинарную строку следующего вида:

$$1\underbrace{00\ldots00}_{a_1\text{ нулей}}1\underbrace{00\ldots00}_{a_2\text{ нулей}}\ldots1\underbrace{00\ldots00}_{a_9\text{ нулей}}.$$

Длина этой строки не превышает 21, значит, ей соответствует бинарное число в промежутке от 0 до $2^{21}-1$. Легко видеть, что таким образом каждому массиву будет сопоставлено свое уникальное число.

Подзадача 4. Вспомним подзадачу 3 и сопоставим исходной строке массив $a_1, a_2, \ldots, a_9$. Заметим, что $a_1+\ldots+a_9 \leqslant 39$ из чего следует, что

$$U = (a_1+1)\cdot\ldots\cdot(a_9+1) \leqslant 5^6\cdot 6^3 < 4\cdot 10^6.$$

Таким образом, можно решить задачу методом динамического программирования, а именно, для каждого массива $b_1, \ldots, b_9$ такого, что $0 \leqslant b_i \leqslant a_i$ и каждого символа $x \in \{a, b, c\}$ вычислить количество строк, оканчивающихся символом x, которым соответствует массив b. Такую динамику можно посчитать за время $O(U) = O((\frac{1}{9}n)^9)$.

Подзадача 5. Немного улучшим решение предыдущей подзадачи, упростив её следующим образом. Для каждого начального символа $x \in \{a, b, c\}$ вычислим количество строк, начинающихся с x и эквивалентных строке $\eta(s)$. Дело в том, что в строке $\eta(s)$ нет подстрок «`aa`», «`bb`» и «`cc`», а значит, ей можно сопоставить массив a длины 6, в котором встречаются количества вхождений подстрок «`ab`», «`ac`», …, «`cb`». Таким образом, суммарное количество состояний в динамике не будет превышать:

$$U = (a_1+1)\cdot\ldots\cdot(a_6+1) \leqslant 11^4\cdot 12^2 < 3\cdot 10^6.$$

Теперь, когда мы знаем количество строк, которые эквивалентны $\eta(s)$ и начинаются с символа x, мы знаем также и то, что все они будут содержать одинаковое количество букв «`a`» — m_a, букв «`b`» — m_b, и букв «`c`» — m_c. Как и в подзадаче 2, мы можем также посчитать количество строк t, которые эквивалентны s и начинаются с буквы x. Итого, получили решение, которое работает за время $O(U) = O((\frac{1}{6}n)^6)$.

Подзадача 6. Воспользуемся тем, что количество строк, которые эквивалентны s не превышает 100. Тогда можно попробовать рекурсивно перебрать эквивалентные ей строки t. Сначала

переберём символ t_1, в каждом из рекурсивных вызовов переберём символ t_2, и так далее. Наивный алгоритм будет работать за $O(3^n)$, но при этом, если мы сделаем отсечения, указанные ниже, то он будет работать за время $O(nw)$:

- Для каждой строки длины 2 будем поддерживать разность u количества её вхождений в строку s и в строку $t_1t_2\dots t_h$ (пусть мы находимся на h-м по вложенности рекурсивном вызове). Тогда если хотя бы одно $u < 0$, то в поддереве данного рекурсивного вызова строк, которые эквивалентны строке t, мы не найдем, и можно сразу вернуть 0.
- Также необходимо проверить, что оставшийся граф эйлеров. Условие на степени вершин выполняется автоматически, его проверять не надо. Однако также надо проверить достижимость всех вершин, из которых выходит хотя бы одно ребро (с учётом того, что по некоторым ребрам мы уже «прошлись» в родительских рекурсивных вызовах и их учитывать не надо).

Данный алгоритм будет работать за $O(nw)$, так как в дереве перебора строк t будет ровно w листьев, которым соответствуют строки, который эквивалентны s, и каждый такой лист будет находиться на глубине n, соответственно всего будет совершенно не более $O(nw)$ рекурсивных вызовов, каждый из которых работает за $O(1)$.

Подзадача 7. При решении подзадачи 2 мы рассматривали строку t, которая эквивалентна строке s, как последовательность вершин в эйлеровом обходе некоторого графа с кратными рёбрами. Так же как и в предыдущих группах для каждого начального символа x посчитаем количество строк w, которые эквивалентны $\eta(s)$, и уже потом с помощью полученных значений и метода шаров и перегородок решим исходную задачу. Займёмся «структурированием» строки w следующим образом.

Пусть w — некоторый эйлеров обход графа из вершин a, b, c, тогда выделим в нём самый левый подцикл и сожмём до одной вершины, будем продолжать этот процесс до тех пор, пока w не превратится в простой путь. Для наглядности приведем пример:

1. «abacbcabcabcbc»;

2. «acbcabcabcbc», цикл «aba» сжат в символ «a»;
3. «acabcabcbc», цикл «cbc» сжат в символ «c»;
4. «abcabcbc», цикл «aca» сжат в символ «a»;
5. «abcbc», цикл «abca» сжат в символ «a»;
6. «abc», цикл «bcb» сжат в символ «b».

В ходе этого процесса строка w превратилась в простой путь; чтобы не терять информацию об исходной строке, для каждого символа, начиная с конца, запомним путь, после свёртки которого остался заданный символ. Всю это информацию можно представить в виде дерева, которое для вышеуказанного примера выглядит так (рис. 19):

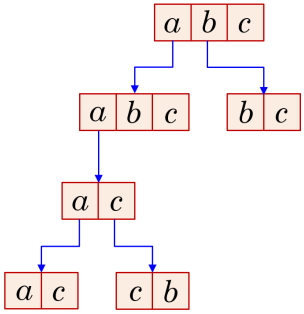


Рис. 19

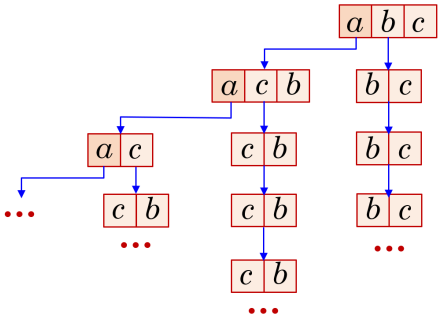


Рис. 20

Легко видеть, что по заданному дереву легко однозначно восстановить исходную строку. Изучим это дерево подробнее. Его корневая вершина представляет собой некоторый простой путь, а любая другая — некоторый простой цикл. В исходном графе может быть всего 5 различных циклов с точностью до циклического сдвига: «abca», «acba», «aba», «aca» и «bcb».

Так же легко видеть, что если мы рассматриваем цикл вида «xyz», который подвешен за вершину x, то в поддеревьях вершин этого цикла y и z не может присутствовать символ x. Так происходит из-за того, что при построении дерева мы всегда выделяли самый левый цикл в последовательности. Аналогично, в поддереве вершины z этого цикла не может присутствовать символ y. Но

тогда поддерево вершины z пустое, так как не существует циклов, состоящих из одной вершины z.

Подобные рассуждения можно провести и при рассмотрении некоторого цикла длины 2 вида «xy», который подвешен за вершину x. В таком случае в поддереве y также не должны присутствовать циклы, содержащие символ x.

Теперь предположим, что мы рассматриваем строку w, начинающуюся с символа «a», тогда все циклы, содержащие символ «a» будут образовывать непрерывную цепочку, то есть вертикальный путь, который подвешен к первой вершине простого пути. Единственный оставшийся цикл «bcb» (или, что тоже самое «cbc»), может образовывать сразу несколько цепочек, которые могут быть подвешены либо за исходный простой путь, либо какую-то вершину-цикл, содержащую символ «a».

На рисунке 20 приведёна иллюстрация дерева, построенного описанным выше образом:

Итак, каждой строке w, которая эквивалентна $\eta(s)$ и начинается с символа «a», мы сопоставили некоторое уникальное дерево. Это значит, что если мы вычислим количество способов составить такие деревья, то мы также найдем количество искомых строк. Как мы уже поняли, дерево можно «собрать» из какого-то простого пути, который начинается с символа «a», а также какого-то набора циклов. При этом каждое ребро в графе должно встретиться ровно один раз либо в простом пути, либо в каком-нибудь из циклов. Обработаем каждое такое разбиение графа на путь и простые циклы отдельно.

Для начала переберём простой путь, затем вычтем его рёбра из графа. Если у какой-то вершины в графе при этом количество исходящих рёбер не совпадает с количеством входящих, то с таким простым путём собрать дерево не получится. В ином случае заметим, что выполняется равенство:

$$k_{ab} - k_{ba} = k_{bc} - k_{cb} = k_{ca} - k_{ac} = \Delta.$$

Предположим, что мы разбили граф на t_{ab} циклов «aba», t_{ac} циклов «aca», t_{bc} циклов «bcb», t_{abc} циклов «abca» и t_{acb} циклов «acba». Тогда $\Delta = t_{abc} - t_{acb}$, так как $k_{ab} = t_{abc} + t_{ab}$ и $k_{ba} = t_{acb} + t_{ab}$. Переберём количество циклов t_{abc}, тогда легко видеть, что однозначно восстанавливаются и количества всех остальных

циклов: $t_{acb} = t_{abc} - \Delta$, $t_{ab} = k_{ab} - t_{abc}$, $t_{bc} = k_{bc} - t_{abc}$, $t_{ac} = k_{ca} - t_{abc}$. При $\max\{\Delta, 0\} \leqslant t_{abc} \leqslant \min\{k_{ab}, k_{bc}, k_{ca}\}$ мы получаем корректное разбиение на циклы. Легко видеть, что существует всего $O(n)$ различных разбиений оставшегося графа на циклы.

Теперь у нас есть циклы, из которых мы можем собрать дерево. Введём переменную G, которая равна 1, если фиксированный на первом шаге простой путь состоит из двух или трёх вершин, и равна 0 — в ином случае. Пусть $K = G + t_{abc} + t_{acb} + t_{ab} + t_{ac}$. Для начала расставим циклы, содержащие символ «a», количество возможных способов равно:

$$\frac{(t_{abc} + t_{acb} + t_{ab} + t_{ac})!}{t_{abc}! \cdot t_{acb}! \cdot t_{ab}! \cdot t_{ba}!}.$$

Теперь к нашему дереву стоит подвесить циклы «bcb», которых t_{bc} штук. Все подвешенные циклы будут образовывать цепочки, которые могут начинаться в K различных позициях, вне зависимости от того, в каком порядке мы расставили циклы, содержащие символ «a». Так как каждый такой способ эквивалентен некоторому разбиению числа t_{bc} на K неотрицательных целых слагаемых, то это можно сделать $C^{K-1}_{t_{bc}+K-1}$ различными способами. Таким образом, для каждого выбранного простого пути и каждого разбиения на циклы мы можем вычислить количество заданных деревьев за время $O(1)$, а с учётом того, что мы рассмотрим $O(1)$ простых путей, для каждого из которых рассмотрим всего лишь $O(n)$ разбиений на циклы, поэтому искомое количество строк w, которые эквивалентны строке $\eta(s)$ и начинаются с символа «a» мы найдём за время $O(n)$ по следующей формуле:

$$\sum_{t_{abc}, t_{acb}, \ldots} \frac{(t_{abc} + t_{acb} + t_{ab} + t_{ac})!}{t_{abc}! \cdot t_{acb}! \cdot t_{ab}! \cdot t_{ba}!} C^{K-1}_{t_{bc}+K-1}.$$

Теперь, как и в подгруппе 2, подсчитаем количество строк t, которые начинаются с символа «a» и эквивалентны строке s. Аналогично поступим и для вычисления количества строк t, которые начинаются с символа «b» или «c».

Задача 9. Красивое название

Автор задачи : *Фольклор.*
Разработчик : *Киндер М.И.*
Разбор задачи : *Киндер М.И.*

Несложно понять, что исследуемая строка будет красивой только в случае, когда гласные и согласные буквы в этой строке *чередуются* между собой. Поэтому нужно проверить, что в строке нет двух гласных и нет двух согласных букв, стоящих рядом друг с другом.

Для уменьшения количества проверок целесообразно воспользоваться функцией, которая выполняет преобразование прописных букв в строчные. Такая функция преобразует параметр в строчный эквивалент, если символ с заглавной буквы. Если же символ уже строчный, то преобразование не выполняется и его значение остаётся неизменным.

В `C++` это действие выполняет функция `tolower()`.

Задача 10. Почти гипотеза 3X+1

Автор задачи : *Фольклор.*
Разработчик : *Киндер М.*
Разбор задачи : *Киндер М.*

Полное решение задачи использует идею рекурсии.

Числа $2, 3, 4$ из 1 мы получать умеем. Покажем, как получить произвольное целое число $n > 4$, если умеем получать все меньшие числа. Число n представимо в одном из трёх видов: $3x$, $3x+1$ или $3x+2$, где x — целое положительное.

Если целое n имеет вид $3x+1$, то его можно получить из меньшего числа x с помощью одной операции `A`: $x \xrightarrow{\texttt{A}} 3x+1$. Другими словами, последняя операция, которая приводит к числу $n = 3x+1$ — это операция `A`. Запомним этот символ `A`, добавив его к строке s, которая будет сохранять искомую последовательность шагов. Теперь те же рассуждения применим к меньшему числу x, и если оно тоже имеет вид $3x_1+1$, добавим к строке s ещё один символ `A`, и применим те же рассуждения к числу x_1, и так далее.

Если целое n имеет вид $3x+2$, то его можно получить из меньшего числа $2x+1$ с помощью операций `A` и `B`: $2x+1 \xrightarrow{\texttt{A}} 3(2x+1)+1 \xrightarrow{\texttt{B}} 3x+2 = n$. Значит, к строке s добавляем символы `BA`, и к меньшему числу $2x+1$ снова применяем аналогичные рассуждения. (Добавление символов `BA`, а не `AB`, связано с тем, что в окончательной записи ответа символы строки s будут записаны в обратном порядке!)

Наконец, если целое n имеет вид $3x$, то его снова получим из меньшего числа $2x$ с помощью операций `A` и `B`: $2x \xrightarrow{\texttt{A}} 3 \cdot 2x + 1 \xrightarrow{\texttt{B}} \xrightarrow{\texttt{B}} (6x+1)/2 = 3x = n$. Значит, к строке s добавляем символы `BA`, и к меньшему числу $2x$ применяем те же рассуждения.

Пример. Пусть $n = 28$. Поскольку число $n = 3 \cdot 9 + 1$ даёт остаток 1 при делении на 3, оно было получено из числа 9 с помощью операции `A`, значит, $s =$`"A"`. Далее, число $n_1 = 9 = 3 \cdot 3$ кратно 3, поэтому его можно получить из числа $2 \cdot 3 = 6$ с помощью операций `A` и `B`: $6 \xrightarrow{\texttt{A}} 19 \xrightarrow{\texttt{B}} 9 \xrightarrow{\texttt{A}} 28$, при этом $s =$ `"A"` $+$ `"BA"` $=$ `"ABA"`. Следующим шагом получим число $n_2 = 6 = 3 \cdot 2$ из числа $2 \cdot 2 = 4$: $4 \xrightarrow{\texttt{A}} 13 \xrightarrow{\texttt{B}} 6$, $s = s +$ `"BA"` $=$ `"ABABA"`. Наконец, число $n_3 = 4$ получено из 1: $1 \xrightarrow{\texttt{A}} 4$, при этом $s = s +$ `"A"` $=$ `ABABAA`. Для получения окончательного ответа осталось записать символы строки s в обратном порядке:

$$\texttt{reverse}(s) = \texttt{AABABA}.$$

(Проверка: $1 \xrightarrow{\texttt{A}} 4 \xrightarrow{\texttt{A}} 13 \xrightarrow{\texttt{B}} 6 \xrightarrow{\texttt{A}} 19 \xrightarrow{\texttt{B}} 9 \xrightarrow{\texttt{A}} 28$.)

Оглавление