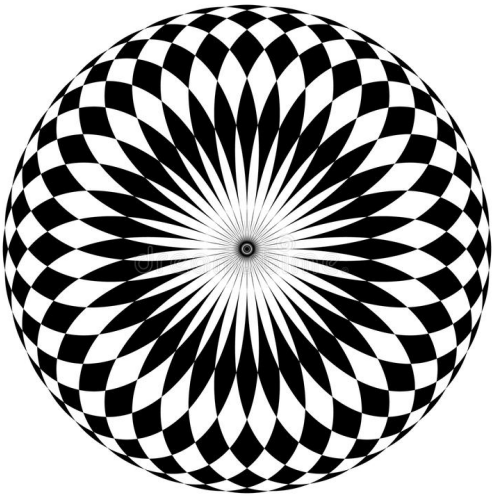
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
КАЗАНСКИЙ ФЕДЕРАЛЬНЫЙ (ПОВОЛЖСКИЙ) УНИВЕРСИТЕТ
РЕСПУБЛИКАНСКИЙ ОЛИМПИАДНЫЙ ЦЕНТР РТ

Олимпиады по информатике школьников Татарстана

2019-2020 и 2020-2021



Казань – 2022

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. 2019-2020 и 2020-2021: Учебно-методическое пособие / М.И. Киндер. — Казань: Казанский федеральный университет, 2022. — 117 с.

Брошюра предназначена для школьников, учителей, преподавателей и тренеров по олимпиадной информатике. В ней представлены задачи, предлагавшиеся в 2019-2020 и 2020-2021 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.

2019-2020 учебный год

Муниципальный этап 32-й Всероссийской олимпиады по информатике среди школьников Республики Татарстан состоялся 14 ноября 2019 г. На муниципальной олимпиаде школьникам 7-11 классов предлагается решить 5 задач. Как правило, первые две задачи в списке заданий опираются, в основном, на логику алгоритмического мышления и не требуют «профессиональных» олимпиадных навыков. Следующие три задачи — более сложные, они охватывают достаточно большой спектр различных стандартных и нестандартных алгоритмов. Автор большинства задач муниципальной олимпиады 2019 г. — доцент Казанского Федерального Университета *М. И. Киндер*.

Региональный этап олимпиады проходил с 16 по 18 января 2020 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 составило 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 Pascal или C++. Для большинства олимпиадных заданий указаны фамилии авторов идеи и фамилии тех, кто готовил эти решения.

Муниципальный этап, 2019-2020

Задача 1. Хамелеоны (7-8 классы)

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

На Солнечном острове живут w белых и b чёрных хамелеонов. Хамелеоны – это животные, способные менять свой цвет. При встрече оба хамелеона меняют свой цвет на противоположный. Например, при встрече двух белых (или чёрных) хамелеонов оба становятся чёрными (белыми), а при встрече хамелеонов белого и чёрного цвета они превращаются в хамелеонов чёрного и белого цвета. В остальных случаях ничего не происходит.

Вам необходимо определить, смогут ли все хамелеоны окраситься в один цвет.

Формат входных данных

В единственной строке записаны два целых числа w и b — количество белых и чёрных хамелеонов соответственно ($1 \leqslant w, b \leqslant \leqslant 10^{18}$).

Формат выходных данных

Выведите `yes`, если все хамелеоны смогут окраситься в один цвет, иначе выведите `no`.

Примеры

стандартный ввод	стандартный вывод
1 1	no
2 1	yes

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	50	$1 \leqslant w, b \leqslant 10^9$	
2	50	$1 \leqslant w, b \leqslant 10^{18}$	1

Задача 2. Два альбома (7-11 классы)

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

У вас есть два альбома, в одном n почтовых марок, а в другом — m почтовых марок. В каждом альбоме все марки разные, но среди них есть такие, которые встречаются в обоих альбомах. Такие марки вы хотели бы обменять при встрече с другими филателистами.

У каждой альбомной марки есть свой уникальный номер — целое число от 1 до 10^9, при этом у одинаковых марок одинаковые номера, а у разных марок — разные номера.

Вам необходимо определить количество совпадающих марок в альбомах, а также номера марок, которые есть и в первом, и во втором альбомах.

Формат входных данных

В первой строке записаны целые n и m — количества марок в первом и втором альбомах соответственно ($1 \leqslant n, m \leqslant 10^5$).

Во второй строке содержатся n различных целых чисел в диапазоне от 1 до 10^9 включительно — уникальные номера марок первого альбома.

В третьей строке содержатся m различных целых в диапазоне от 1 до 10^9 включительно — уникальные номера марок второго альбома.

Формат выходных данных

В первой строке запишите число k — количество совпадаю-

щих марок двух альбомов.

Во второй строке запишите в порядке возрастания k целых чисел — уникальные номера этих совпадающих марок.

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	30	$1 \leqslant n, m \leqslant 5$	
2	30	$1 \leqslant n, m \leqslant 20\,000$	1
3	40	$1 \leqslant n, m \leqslant 100\,000$	1, 2

Примеры

стандартный ввод	стандартный вывод
3 3 10 15 20 1 2 3	0
5 6 2 10 5 13 4 6 2 13 3 10 1	3 2 10 13

Задача 3. Факториал (7-11 классы)

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

На доске записана длинная последовательность цифр — десятичная запись числа $n!$, равного произведению всех целых чисел от 1 до n.

Известный школьный хулиган, лентяй и двоечник Виктор Перестукин стёр одну из цифр в записи этого числа, а вместо неё нарисовал символ #. Запись числа $n!$ была очень длинной, и конечно, Перестукин тут же забыл стёртую цифру, и где она располагалась.

Вскоре выяснилось, что число $n!$ входит в условие олимпиадной задачи по информатике, и теперь нужно срочно восстановить его. После воспитательного разговора с директором школы Виктору удалось вспомнить только, что стёртая цифра была отлична от 0.

Ваша задача — составить программу, которая определяет стёртую цифру в записи числа $n!$.

Формат входных данных

В единственной строке записана непустая строка s, состоящая из цифр от 0 до 9 и символа #. Строка s соответствует десятичной записи некоторого числа вида $n!$ ($1 \leqslant n \leqslant 10\,000$), в которой один ненулевой символ заменен на символ #. Максимальная длина строки — 35660 (соответствует записи числа 10 000!). Первый (слева) символ в строке s отличен от нуля.

Формат выходных данных

Запишите одну ненулевую цифру, которая должна стоять в записи $n!$ вместо символа #. Если решений несколько, выведите любое из них.

Примеры

стандартный ввод	стандартный вывод
2#	4
#20	1

Пояснение к примеру

Во втором примере правильным ответом будет также число 7. (Если вместо символа # записать цифру 7, получится число $720 = 6!$.)

Система оценивания

Баллы за каждую подзадачу начисляются только в случае,

если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. (Обозначение: m — длина строки s.)

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	20	$1 \leqslant m \leqslant 3$	
2	20	$1 \leqslant m \leqslant 20$	1
3	30	$1 \leqslant m \leqslant 5000$	1, 2
4	30	$1 \leqslant m \leqslant 35660$	1, 2, 3

Задача 4. Сортировка мусора (7-11 классы)

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

Мусор — одно из главных экологических бедствий современной экологии. Каждый год в мире выбрасываются миллионы тонн предметов, загрязняющих атмосферу и почву. Самый безопасный способ обращения с отходами – раздельный сбор и последующая его переработка.

Во многих странах каждый вид отходов собирается в отдельный контейнер: пластик, бумага, алюминиевые банки и тому подобное. Если же в одном контейнере находятся разные виды отходов, их нужно предварительно рассортировать.

Перед вами n контейнеров, в которых находятся n различных видов отходов. Необходимо рассортировать мусор по этим же контейнерам так, чтобы в каждом из них остался только один вид отходов. За каждую операцию можно перенести только одну единицу какого-либо вида мусора из одного контейнера в другой.

Вам нужно вычислить наименьшее число операций, необходимое для сортировки мусора.

Формат входных данных

В первой строке записано целое число n — количество контейнеров и видов мусора ($2 \leqslant n \leqslant 10$).

В каждой из следующих n строк записаны n целых чисел a_{i1}, $a_{i2}, \ldots a_{in}$, указывающих соответственно количество отходов вида 1, вида 2, и так далее, наконец, вида n, в контейнере с номером i ($1 \leqslant i \leqslant n$). Все числа a_{ij} в диапазоне от 1 до $2 \cdot 10^9$ включительно.

Формат выходных данных

Запишите наименьшее количество операций, необходимое для сортировки мусора по контейнерам.

Пример

стандартный ввод	стандартный вывод
3 1 2 1 2 1 1 2 2 2	8

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	$n = 2$	
2	20	$n = 3$	1
3	20	$n = 4$	1, 2
4	40	$1 \leqslant n \leqslant 10$	1, 2, 3

Задача 5. Кенгурёнок и Тигра (9-11 классы)

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

Маленький кенгурёнок Ру из книги английского писателя Алана Милна «Винни-Пух» сидит в точке A, и, как все кенгуру, умеет только прыгать и совершенно не умеет бегать. Длина каждого его прыжка постоянная и равна d. Каждый вечер кенгурёнок отправляется в гости к своему приятелю Тигре, который живёт в точке B.

Вам необходимо составить программу, которая определяет минимальный по длине маршрут перемещения кенгурёнка из точки A в точку B.

Формат входных данных

В первой строке записаны 4 целых числа x_A, y_A и x_B, y_B — координаты двух различных точек A и B ($-150 \leqslant x_A, y_A, x_B, y_B \leqslant 150$).

Во второй строке одно целое число d — длина прыжка кенгурёнка ($1 \leqslant d \leqslant 150$).

Формат выходных данных

В первой строке запишите число n — количество прыжков в минимальном по длине маршруте.

В следующих n строках выведите по два действительных числа — координаты (x_i, y_i), в которых кенгурёнок окажется после i-го прыжка. Ответ считается верным, если абсолютная или относительная погрешность не превосходит 10^{-5}.

Если минимальных по длине маршрутов несколько, выведите любой из них. Если решения не существует, выведите `-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	25	$d > AB$	
2	25	$x_A = x_B$	1
3	25	$y_A = y_B$	1
4	25		1, 2, 3

Примеры

стандартный ввод	стандартный вывод
0 0 1 1 1	2 0.0000000000 1.0000000000 1.0000000000 1.0000000000
0 0 6 8 5	2 3.0000000000 4.0000000000 6.0000000000 8.0000000000

Задача 6. Простые суммы (9-11 классы)

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

Я — простой человек... Делюсь только на единицу и на самого себя.

Я. Графоманов «Простые люди и числа».

В соревнованиях по бадминтону участвуют n игроков. Каждый игрок имеет свой индивидуальный номер — целое число от 1 до 10^6. Перед началом первого тура всех спортсменов разбивают на игровые пары так, чтобы каждый спортсмен участвовал не более, чем в одной паре.

Судья соревнований — большой оригинал, эстет и любитель математики — решил разбить участников соревнования на пары таким образом, чтобы сумма номеров участников в каждой паре была *простым* числом.

Вам необходимо определить наибольшее количество пар, которое можно составить из игроков требуемым образом, и вывести эти пары.

Формат входных данных

В первой строке записано целое число n — количество участников соревнования ($1 \leqslant n \leqslant 2500$).

Во второй строке содержатся n различных целых чисел в диапазоне от 1 до 10^6 включительно — номера участников.

Формат выходных данных

В первой строке выведите число k — наибольшее возможное количество пар участников с простой суммой их номеров.

В следующих k строках описание разбиение участников соревнования на пары. Каждая из этих k строк должна содержать два целых числа — номера участников, из которых составлена пара.

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	30	$1 \leqslant n \leqslant 10$	
2	30	$1 \leqslant n \leqslant 100$	1
3	20	$1 \leqslant n \leqslant 1500$	1, 2
4	20	$1 \leqslant n \leqslant 2500$	1, 2, 3

Примеры

стандартный ввод	стандартный вывод
4 12 9 6 3	0
7 20 1 2 3 12 13 14	2 20 3 2 1

Региональный этап, 2019-2020

Региональный этап Всероссийской олимпиады школьников состоялся 16-18 января 2020 года. Соревнование проходило в два тура с перерывом (17 января) между ними на день отдыха. На каждый тур олимпиады школьникам предлагалось 4 задачи. Ниже представлены тексты и разбор решения этих задач.

Разбор подготовили Андрей Станкевич, Николай Будин, Дмитрий Гнатюк, Илья Збань, Даниил Орешников, Рамазан Рахматуллин, Иван Сафонов.

Условия задач олимпиады, тесты и решения подготовили Михаил Анопренко, Николай Будин, Дмитрий Гнатюк, Александра Дроздова, Илья Збань, Арсений Кириллов, Даниил Орешников, Рамазан Рахматуллин, Иван Сафонов, Дмитрий Саютин, Андрей Станкевич.

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. Разность квадратов

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

Вы участвуете в разработке программного модуля для системы символьных вычислений. Модуль будет использоваться для решения специального вида диофантовых уравнений.

По заданному целому неотрицательному целому числу n разрабатываемый модуль должен найти два натуральных числа x и y, для которых выполнено равенство $x^2 - y^2 = n$. Найденные числа не должны превышать $2^{62} - 1$.

Требуется написать программу, которая по заданному целому неотрицательному числу n находит натуральные числа x и y, не превышающие $2^{62} - 1$, разность квадратов которых равна n.

Формат входных данных

В единственной строке дано одно целое число n ($0 \leqslant n \leqslant 2^{60}$).

Обратите внимание, что заданное во вводе число не помещается в 32-битный тип данных, необходимо использовать 64-битный тип данных (например, «`long long`» в `C++`, «`int64`» в паскале, «`long`» в Java).

Формат выходных данных

Если искомые x и y существуют, то необходимо вывести две строки: в первой строке выведите слово «`Yes`», а во второй — искомые x и y.

Если подходящих пар x и y несколько, можно вывести любую из них, но должно выполняться условие $1 \leqslant x, y \leqslant 2^{62} - 1$.

Если решения нет, в единственной строке необходимо вывести слово «`No`».

Примеры

стандартный ввод	стандартный вывод
3	Yes 2 1
2	No

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	10	$0 \leqslant n \leqslant 2^{10}$	
2	20	$0 \leqslant n \leqslant 2^{20}$	1
3	30	$0 \leqslant n \leqslant 2^{30}$	1, 2
4	40	$0 \leqslant n \leqslant 2^{60}$	1, 2, 3

Задача 2. Превышение скорости

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

Превышение скорости является опасным нарушением, значительно увеличивающим вероятность трагических последствий транспортных происшествий. К сожалению, контроль скорости с использованием радаров и камер не решает проблему полностью. Притормаживая перед камерами, водители едут со значительным превышением на участках дорог, где контроль не ведётся. С целью предотвращения такого поведения используется назначение штрафа за гарантирование превышение скорости, основанное на времени проезда дороги.

Рассмотрим дорогу, состоящую из n участков, пронумерованных от 1 до n. Длина i-го участка составляет l_i метров. На i-м из участков установлено ограничение по скорости в v_i $м/с$.

За превышение скорости предусмотрены штрафы. В зависимости от превышения, установлены различные штрафы, величи-

на штрафа вычисляется следующим образом.

Пусть e — максимальное превышение разрешённой скорости в процессе пребывания автомобиля на всей дороге, то есть максимальная разница между скоростью автомобиля и максимальной разрешённой скоростью на участке, где он в этот момент находится. Если превышения скорости не было, то штраф не взимается. В противном случае штраф вычисляется так:

- если $0 < e \leqslant a_1$, то штраф составляет f_1 денежных единиц;
- если $a_1 < e \leqslant a_2$, то штраф составляет f_2 денежных единиц;
- ...
- если $a_{m-2} < e \leqslant a_{m-1}$, то штраф составляет f_{m-1} денежных единиц;
- если $a_{m-1} < e$, то штраф составляет f_m денежных единиц.

Таким образом, есть m диапазонов превышения скорости и соответствующие им штрафы.

Автоматическая система назначения штрафов получила данные о q автомобилях. Для удобства пронумеруем их от 1 до q. Известно, что i-й автомобиль заехал на дорогу в момент времени s_i, проехал все n участков, после чего выехал с нее в момент времени t_i. Отсчёт времени будем вести в секундах с открытия дороги.

Для каждого из автомобилей система должна определить, какой максимальный штраф можно гарантированно выписать этому автомобилю, основываясь только на времени заезда на дорогу и выезда с неё.

Требуется написать программу, которая по описанию границ диапазонов превышения скорости, соответствующих штрафов и временам въезда/выезда автомобилей определяет для каждого автомобиля максимальный штраф, который можно выписать этому автомобилю.

Формат входных данных

Первая строка входных данных содержит единственное целое число n — количество участков на дороге ($1 \leqslant n \leqslant 10$).

Вторая строка содержит n целых чисел v_i — ограничения скорости на участках ($1 \leqslant v_i \leqslant 10^9$).

Третья строка содержит n целых чисел l_i — длины участков ($1 \leqslant l_i \leqslant 10^9$).

Четвертая строка содержит одно целое число m — количество границ диапазонов превышения скорости ($1 \leqslant m \leqslant 10^5$).

Пятая строка содержит $m-1$ целых чисел a_i — границы диапазонов превышения скорости ($1 \leqslant a_i \leqslant 10^9$). Гарантируется, что значения a_i строго возрастают. Обратите внимание, что если $m = 1$, то пятая строка ввода пустая.

Шестая строка содержит m целых чисел f_i — штрафы за диапазоны превышения скоростей ($1 \leqslant f_i \leqslant 10^9$). Гарантируется, что значения f_i возрастают.

Седьмая строка содержит единственное целое число q — количество автомобилей, которые надо обработать ($1 \leqslant q \leqslant 10^5$).

Каждая из следующих q строк содержит два целых числа s_i и t_i — время заезда на трассу и выезда с неё i-го из рассматриваемых автомобилей ($1 \leqslant s_i < t_i \leqslant 10^9$).

Формат выходных данных

Для каждого из q автомобилей выведите в отдельной строке максимальный штраф, который гарантированно можно выписать этому автомобилю, основываясь только на времени его заезда на дорогу и выезда с неё. Если возможна ситуация, что автомобиль ни разу не превысил разрешённую скорость, следует вывести 0.

Гарантируется, что если время въезда или выезда автомобиля изменить не более чем на 10^{-5}, штраф, который можно ему выписать, не изменится.

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	$n = 1, m = 1$	
2	10	$m = 1$	1
3	9	$n = 1, m \leqslant 10$	1
4	12	$n = 1$	1, 3
5	13	$m \leqslant 10,\ a_i \leqslant 10$	
6	14	$m \leqslant 10$	1, 2, 3, 5
7	37		1–6

Пример

стандартный ввод	стандартный вывод
3 10 20 30 400 500 600 6 1 5 10 12 16 100 300 600 800 1000 1500 3 10 100 20 70 45 100	0 800 600

Задача 3. Борьба с рутиной

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

Важным элементом повышения эффективности работы сотрудников является борьба с рутиной. Построим математическую модель разнообразия типов заданий, выполняемых сотрудником в компании. Рассмотрим работу сотрудника в течение n последовательных рабочих дней. Будем считать, что каждый день сотрудник выполняет ровно один тип заданий. Обозначим тип заданий, выполняемый сотрудником в i-й день, целым числом a_i.

Для оценки рутинности работы сотрудника будем использовать следующую характеристику. Зафиксируем целое число d и рассмотрим все отрезки из d подряд идущих рабочих дней. Для каждого такого отрезка найдём количество различных типов заданий, которые работник выполнял на протяжении этих дней, и просуммируем эти значения. Полученную величину обозначим как S_d и будем называть её d-разнообразием. Чем выше d-разнообразие, тем больше различных типов заданий выполнял сотрудник. *Профилем вариативности* сотрудника будем называть массив значений $[S_1, S_2, \ldots, S_n]$.

Требуется написать программу, которая по заданной последовательности a_1, a_2, …, a_n типов выполняемых сотрудником заданий вычисляет его профиль вариативности.

Формат входных данных

В первой строке находится единственное целое число n — количество последовательных рабочих дней, которые необходимо проанализировать ($1 \leqslant n \leqslant 2 \cdot 10^5$).

Во второй строке находится n целых чисел a_1, a_2, …, a_n — типы заданий, которое выполнял сотрудник ($1 \leqslant a_i \leqslant 10^9$).

Формат выходных данных

Выведите n целых чисел $S_1, S_2, \ldots, S_n$.

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	12	$1 \leqslant n \leqslant 50, 1 \leqslant a_i \leqslant 50$	
2	10	$1 \leqslant n \leqslant 50, 1 \leqslant a_i \leqslant 10^9$	1
3	10	$1 \leqslant n \leqslant 500, 1 \leqslant a_i \leqslant 10^9$	1, 2
4	12	$1 \leqslant n \leqslant 5000, 1 \leqslant a_i \leqslant 5000$	1
5	10	$1 \leqslant n \leqslant 5000, 1 \leqslant a_i \leqslant 10^9$	1-4
6	16	$1 \leqslant n \leqslant 2 \cdot 10^5, 1 \leqslant a_i \leqslant 50$	1
7	30	$1 \leqslant n \leqslant 2 \cdot 10^5, 1 \leqslant a_i \leqslant 10^9$	1–6

Примеры

стандартный ввод	стандартный вывод
5 1 3 2 1 2	5 8 8 6 3
3 10 10 10	3 2 1

Пояснение к примерам

Рассмотрим, как вычисляются значения S_d в первом примере.

1-разнообразие: необходимо просуммировать количество различных типов заданий, выполняемых сотрудником, по всем отрезкам, состоящим из одного дня.

Отрезок дней	Типы заданий	Количество различных
1 – 1	1	1
2 – 2	3	1
3 – 3	2	1
4 – 4	1	1
5 – 5	2	1

Значение 1-разнообразия равно $S_1 = 1 + 1 + 1 + 1 + 1 = 5$.

2-разнообразие: необходимо просуммировать количество различных типов заданий, выполняемых сотрудником, по всем отрезкам, состоящим из двух дней.

Отрезок дней	Типы заданий	Количество различных
1 – 2	1, 3	2
2 – 3	3, 2	2
3 – 4	2, 1	2
4 – 5	1, 2	2

Значение 2-разнообразия равно $S_2 = 2 + 2 + 2 + 2 = 8$.

3-разнообразие: необходимо просуммировать количество различных типов заданий, выполняемых сотрудником, по всем отрезкам, состоящим из трёх дней.

Отрезок дней	Типы заданий	Количество различных
1 – 3	1, 3, 2	3
2 – 4	3, 2, 1	3
3 – 5	2, 1, 2	2

Значение 3-разнообразия равно $S_3 = 3 + 3 + 2 = 8$.

4-разнообразие: необходимо просуммировать количество различных типов заданий, выполняемых сотрудником, по всем отрезкам, состоящим из четырёх дней.

Отрезок дней	Типы заданий	Количество различных
1 – 4	1, 3, 2, 1	3
2 – 5	3, 2, 1, 2	3

Значение 4-разнообразия равно $S_4 = 3 + 3 = 6$.

5-разнообразие: необходимо просуммировать количество различных типов заданий, выполняемых сотрудником, по всем отрезкам, состоящим из пяти дней.

Отрезок дней	Типы заданий	Количество различных
1 – 5	1, 3, 2, 1, 2	3

Значение 5-разнообразия равно $S_5 = 3$.

Задача 4. Олимпиада для роботов

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

Жюри чемпионата по скоростному вычислению булевых функций среди роботов готовит задания для участников.

Задание для роботов представляет собой таблицу из m строк и n столбцов, каждая ячейка которой содержит целое число. Обозначим число в i-й строке, j-м столбце таблицы как $x_{i,j}$. В каждом столбце значения в ячейках таблицы образуют перестановку чисел от 0 до $m-1$. Иначе говоря, числа в каждом столбце различны: если $i \neq k$, то $x_{i,j} \neq x_{k,j}$ для всех j, и выполнено условие $0 \leqslant x_{i,j} < m$.

Для каждого столбца таблицы задаётся значение *порога* — целое неотрицательное число z_j от 0 до m. В качестве аргументов булевых функций, которые будут вычислять участники олимпиады, используются значения логических выражений $x_{i,j} < z_j$. Значение такого логического выражения равно 1, если неравенство выполнено, иначе оно равно 0.

В процессе соревнования участники вычисляют значения m булевых функций — по одному для каждой строки. Каждая булева функция задаётся в виде *бесповторной монотонной линейной программы* (БМЛП).

Рассмотрим БМЛП для i-й строки таблицы. Она представляет собой последовательность из $n-1$ инструкции, пронумерованных от 1 до $n-1$, p-я инструкция задаётся тремя числами: a_p, b_p и op_p. Число op_p принимает два возможных значения: 1 для операции `and` — логическое «и», 2 для операции `or` — логическое «или». Числа a_p и b_p являются номерами аргументов p-й инструкции, выполнены неравенства $1 \leqslant a_p, b_p < n + p$.

Рассмотрим массив $val[1\,..\,2n-1]$, каждое из значений которого равно 0 или 1. Инициализируем массив $val[1]\,..\,val[n]$ значениями порогов: $val[j] = 1$, если $x_{i,j} < z_j$, иначе $val[j] = 0$. Значение $val[n+p]$ вычисляется с использованием p-й инструкции.

- Если $op_p = 1$, то $val[n+p] = (val[a_p]$ `and` $val[b_p])$, то есть

значение $val[n+p]$ равно 1, если и только если каждое из значений $val[a_p]$ и $val[b_p]$ равно 1.

- Если $op_p = 2$, то $val[n+p] = (val[a_p]$ `or` $val[b_p])$, то есть значение $val[n+p]$ равно 1, если и только если хотя бы одно из значений $val[a_p]$ и $val[b_p]$ равно 1.

При этом программа является бесповторной, а именно все $2n-2$ значений a_p и b_p для p от 1 до $n-1$ различны. Иначе говоря, $a_p \neq b_p$, а если $p \neq q$, то $a_p \neq a_q$, $a_p \neq b_q$, $b_p \neq a_q$ и $b_p \neq b_q$.

Результатом исполнения программы является значение $val[2n-1]$.

Жюри олимпиады подготовило таблицу $x_{i,j}$, выбрало булевы функции для каждой строки и записало их в виде БМЛП. Теперь осталось выбрать значение порога для каждого столбца, чтобы получить задание для олимпиады. Жюри считает задание сбалансированным, если ровно s из m программ для строк таблицы возвращают единицу, а остальные $m-s$ возвращают ноль. Ваша задача — помочь жюри найти подходящие значения порогов.

Требуется написать программу, которая по заданным значениям в ячейках таблицы и БМЛП для строк таблицы определяет такие значения порогов z_j, чтобы значения ровно s из m заданных функций были равны 1. Можно доказать, что при описанных в условии задачи ограничениях требуемые значения порогов всегда можно подобрать.

Формат входных данных

В первой строке входных данных заданы целые числа n, m и s ($1 \leqslant n \leqslant 3 \cdot 10^5$, $1 \leqslant m \leqslant 3 \cdot 10^5$, $n \cdot m \leqslant 3 \cdot 10^5$, $0 \leqslant s \leqslant m$).

Далее следует m блоков по $n-1$ строке в каждом, каждый блок задает бесповторную монотонную линейную программу для одной строки таблицы. В каждом блоке p-я строка содержит три целых числа: a_p, b_p и op_p ($1 \leqslant a_p < n+p$, $1 \leqslant b_p < n+p$; гарантируется, что в одном блоке все значения a_p и b_p попарно различны, $op_p = 1$ или $op_p = 2$).

Последние m строк задают таблицу, i-я строка содержит n целых чисел, j-е из которых равно $x_{i,j}$ ($0 \leqslant x_{i,j} \leqslant m-1$; в каждом столбце все числа различны, то есть если $i \neq k$, то $x_{i,j} \neq x_{k,j}$ для всех j).

Формат выходных данных

Выведите n целых чисел — искомые значения порогов z_1, z_2, …, z_n ($0 \leqslant z_j \leqslant m$). Если подходящих вариантов несколько, выведите любой из них.

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	$n \leqslant 2,\ m \leqslant 10^3$	
2	10	$n \leqslant 2,\ m \leqslant 10^5$	1
3	10	$n \leqslant 10,\ m \leqslant 2$	
4	5	$x_{i,j} = i - 1$	
5	5	$op_p = 1$, только операции «и»	
6	20	$n \leqslant 100$	1, 2, 3
7	10	БМЛП для всех строк одинаковые	
8	30	нет	1 – 7

Пример

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

Пояснение к примеру

Входные данные содержат $m = 3$ блока по $n - 1 = 3$ строки в каждом, каждой строке соответствует формула. Необходимо найти четыре порога так, чтобы ровно две формулы возвращали 1, а оставшаяся — 0.

Рассмотрим, как будет вычисляться массив val для первой строки. Первые четыре значения вычисляются на основе чисел в этой строке и порогов:

- $val[1] = (x_{1,1} < z_1) = (0 < 0) = 0$;
- $val[2] = (x_{1,2} < z_2) = (1 < 1) = 0$;
- $val[3] = (x_{1,3} < z_3) = (2 < 2) = 0$;
- $val[4] = (x_{1,4} < z_4) = (2 < 3) = 1$.

Далее выполняем линейную программу для первой строки:

- $val[5] = (val[1]$ `and` $val[2]) = (0$ `and` $0) = 0$;
- $val[6] = (val[3]$ `and` $val[4]) = (0$ `and` $1) = 0$;
- $val[7] = (val[5]$ `or` $val[6]) = (0$ `or` $0) = 0$.

Таким образом значение булевой функции для первой строки равно 0. Кстати, если эту функцию записать формулой, то получится:

$$(((x_{1,1} < z_1)\ \texttt{and}\ (x_{1,2} < z_2))\ \texttt{or}\ ((x_{1,3} < z_3)\ \texttt{and}\ (x_{1,4} < z_4))).$$

Аналогично, булева функция для второй строки равна:

$$(((x_{2,1} < z_1)\ \texttt{or}\ (x_{2,2} < z_2))\ \texttt{and}\ ((x_{2,3} < z_3)\ \texttt{or}\ (x_{2,4} < z_4))),$$

а для третьей строки:

$$(((x_{3,1} < z_1)\ \texttt{and}\ (x_{3,4} < z_4))\ \texttt{or}\ ((x_{3,2} < z_2)\ \texttt{and}\ (x_{3,3} < z_3))).$$

При подстановке порогов $z_1 = 0$, $z_2 = 1$, $z_3 = 2$, $z_4 = 3$ получим следующие выражения.

Вторая строка:

$$\begin{gathered}((((2 < 0)\ \texttt{or}\ (2 < 1))\ \texttt{and}\ (1 < 2))\ \texttt{or}\ (0 < 3)) = \\ = (((0\ \texttt{or}\ 0)\ \texttt{and}\ 1)\ \texttt{or}\ 1) = (0\ \texttt{or}\ 1) = 1.\end{gathered}$$

Третья строка:

$$\begin{gathered}(((1 < 0)\ \texttt{and}\ (1 < 3))\ \texttt{or}\ ((0 < 1)\ \texttt{and}\ (0 < 2))) = \\ = ((0\ \texttt{and}\ 1)\ \texttt{or}\ (1\ \texttt{and}\ 1)) = (0\ \texttt{or}\ 1) = 1.\end{gathered}$$

Отметим, что это не единственный подходящий набор порогов, также подойдут, например, значения $z_1 = 0$, $z_2 = 0$, $z_3 = 3$, $z_4 = 3$.

Задача 5. Максимальное произведение

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

Задан массив натуральных чисел $[\,a_1, a_2, \ldots, a_n\,]$. *Весом* массива назовём сумму его элементов.

Необходимо разрезать заданный массив на два непустых массива $[\,a_1, a_2, \ldots, a_i\,]$ и $[\,a_{i+1}, a_2, \ldots, a_n\,]$ так, чтобы произведение их весов было как можно больше.

Требуется написать программу, которая по заданному массиву определяет, после какого элемента его необходимо разрезать, чтобы произведение весов получившихся массивов было максимальным.

Формат входных данных

В первой строке входных данных записано целое число n — количество элементов в массиве ($2 \leqslant n \leqslant 2 \cdot 10^5$). В следующей строке находятся n целых чисел $a_1, a_2, \ldots, a_n$ — элементы массива ($1 \leqslant a_i \leqslant 10^9$).

Формат выходных данных

Выведите одно число — номер элемента, после которого необходимо разрезать заданный массив. Если оптимальных вариантов ответа несколько, можно вывести любой из них.

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	10	$2 \leqslant n \leqslant 5000$, сумма a_i не больше 10^9	
2	10	$2 \leqslant n \leqslant 5000$, все a_i равны	
3	20	$2 \leqslant n \leqslant 5000$, $a_i \leqslant 10^9$	1, 2
4	20	$2 \leqslant n \leqslant 200\,000$ сумма a_i не больше 10^9	1
5	20	$2 \leqslant n \leqslant 200\,000$, все a_i равны	2
6	20	$2 \leqslant n \leqslant 200\,000$, $a_i \leqslant 10^9$	1 – 5

Пример

стандартный ввод	стандартный вывод
3 1 2 3	2

Пояснение к примеру

Если сделать разрез после первого элемента, произведение весов равно $1 \cdot (2+3) = 5$, а если после второго, то $(1+2) \cdot 3 = 9$.

Задача 6. Планировка участка

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

Учёные планируют участок для испытательного полигона. Участок должен иметь форму прямоугольника $a \times b$, а полигон должен иметь форму прямоугольника $c \times d$. С точными значениями чисел a, b, c и d ученые пока не определились, однако известно следующее:

- Длины сторон a, b, c, d должны быть натуральными числами и выражаться в километрах.
- Для безопасности эксперимента длина и ширина участка должны отличаться от значения x, то есть должны выполняться неравенства $a \neq x$, $b \neq x$.
- Участок будет огражден забором, а полигон должен полностью помещаться внутри участка, то есть должны выполняться следующие условия: $a > c$, $b > d$.
- Площадь участка, не занятого полигоном, должна быть ровно n квадратных километров, то есть должно выполняться следующее условие: $a \cdot b - c \cdot d = n$.

Учёные хотят понять, сколько у них способов выбрать подходящие значения a, b, c и d.

Требуется написать программу, которая по заданным n и x определяет количество способов выбрать числа a, b, c и d так, чтобы все описанные условия выполнялись.

Формат входных данных

В первой строке ввода содержится число n — площадь свободного участка без полигона ($1 \leqslant n \leqslant 3000$).

Во второй строке ввода содержится целое число x — запрещённая длина стороны участка ($0 \leqslant x \leqslant 3000$). Значение $x = 0$ означает, что ограничений на длины сторон нет (так как длины сторон должны быть натуральными числами, и, следовательно, больше 0).

Формат выходных данных

В единственной строке выведите количество способов выбрать числа a, b, c и d так, что все описанные условия выполняются.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Для подзадач 1 и 2 сообщается информация «Первая ошибка», для подзадач 3 – 6 — «Баллы».

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	11	$1 \leqslant n \leqslant 50, x = 0$	
2	10	$1 \leqslant n \leqslant 50$	1
3	20	$1 \leqslant n \leqslant 500, x = 0$	1
4	22	$1 \leqslant n \leqslant 500$	1, 2, 3
5	17	$1 \leqslant n \leqslant 3000, x = 0$	1, 3
6	20	$1 \leqslant n \leqslant 3000$	1–5

Примеры

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

Пояснение к примерам

В первом тестовом примере подходят только $a = 2$, $b = 2$, $c = 1$, $d = 1$.

Во втором тестовом примере подходят следующие ответы:

- $a = 2$, $b = 3$, $c = 1$, $d = 1$;
- $a = 2$, $b = 4$, $c = 1$, $d = 3$;
- $a = 3$, $b = 2$, $c = 1$, $d = 1$;
- $a = 3$, $b = 3$, $c = 2$, $d = 2$;
- $a = 4$, $b = 2$, $c = 3$, $d = 1$;

Во третьем тестовом примере подходят ответы:

- $a = 2$, $b = 4$, $c = 1$, $d = 3$;
- $a = 4$, $b = 2$, $c = 3$, $d = 1$;

В остальных ответах из предыдущего теста либо a, либо b равны 3.

Задача 7. Банкомат

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

Ведётся разработка универсального банкомата, который сможет работать с любой денежной системой. Пусть в денежной системе некоторой страны используются n типов купюр, номиналы которых равны a_1, a_2, …, a_n. При этом все номиналы различны и перечислены в порядке возрастания: если $i \geqslant 2$, то $a_{i-1} < a_i$, а также $a_1 = 1$.

Банкомат использует следующий жадный алгоритм для выдачи купюр. Пусть клиент запросил у банкомата сумму c. Изначально есть пустой набор выдаваемых купюр. На каждом шаге алгоритм добавляет в набор купюру максимально возможного

номинала так, чтобы сумма номиналов купюр в наборе не превышала c. Когда сумма номиналов купюр в наборе стала равна c, алгоритм останавливается. Отметим, что поскольку существует купюра с номиналом $a_1 = 1$, алгоритм всегда заканчивает работу за конечное число шагов.

Чтобы оценить эффективность данного алгоритма, требуется выяснить, какое максимальное число купюр может потребоваться выдать за один раз, если максимальная сумма, которую можно запросить, равна b. Поскольку максимальная сумма может зависеть от категории обслуживания клиента, необходимо ответить на q запросов для сумм b_1, b_2, ..., b_q.

Требуется написать программу, которая по списку номиналов купюр денежной системы и максимальной сумме, которую можно запросить, определяет сумму, которую необходимо запросить в банкомате, чтобы получить максимальное число купюр, и искомое число купюр.

Формат входных данных

Первая строка содержит целое число n — количество номиналов купюр ($1 \leqslant n \leqslant 200\,000$). Вторая строка содержит n различных целых чисел a_i ($1 = a_1 < a_2 < \ldots < a_n \leqslant 10^{18}$).

Третья строка содержит целое число q — количество запросов ($1 \leqslant q \leqslant 200\,000$). Следующие q строк содержат по одному целому числу b_i ($1 \leqslant bi \leqslant 10^{18}$).

Формат выходных данных

Для каждого запроса выведите два числа — сумму, которую необходимо запросить в банкомате, чтобы получить максимальное число купюр, и искомое число купюр.

Если существует несколько сумм, не превышающих максимального значения, для которых будет выдано максимальное число купюр, требуется 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 каждой подзадачи сообщается информация «Первая ошибка».

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	13	$n \leqslant 500$, $q \leqslant 5$, $a_i \leqslant 500$, $b_i \leqslant 500$	
2	18	$n = 60$, $q \leqslant 5$, $a_i = 2^{i-1}$	
3	20	$q \leqslant 5$, $b_i \leqslant 2 \cdot 10^5$	1
4	21	$q \leqslant 5$	1, 2, 3
5	28		1–4

Пример

стандартный ввод	стандартный вывод
4 1 5 10 50 3 2 8 50	2 2 8 4 49 9

Пояснение к примеру

В примере сумма 2 будет выдана двумя купюрами — $1 + 1$, сумма 8 будет выдана четырьмя купюрами — $5+1+1+1$, сумма 49 будет выдана 9-ью купюрами: $10+10+10+10+5+1+1+1+1$.

Задача 8. Плакаты

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

Друзья готовятся встретить национальную команду, возвращающуюся с международной олимпиады по информатике. Для этого они подготовили множество красочных плакатов. Осталось только продумать детали поздравления.

Для приветствия команды n друзей встают в круг. Пронумеруем их от 1 до n в порядке их расположения по кругу. Таким образом, для всех i от 1 до $n-1$ друзья с номерами i и $i+1$ стоят рядом, также рядом стоят друзья с номерами n и 1. У каждого из друзей есть плакат. Каждый плакат характеризуется своей красочностью — целым неотрицательным числом. Плакат у друга с номером i имеет красочность a_i.

Когда поздравление начнётся, некоторые из друзей поднимут свои плакаты и будут показывать их команде. Для того, чтобы члены команды не растерялись и смогли рассмотреть все плакаты, не должно быть четырёх или более стоящих подряд друзей с поднятым плакатом.

Друзья планируют изменять плакаты в процессе встречи. Всего будет внесено q изменений в плакаты. После i-го из них плакат друга с номером p_i будет иметь красочность v_i. После каждого из изменений друзья хотят определить, какую максимальную суммарную красочность могут иметь поднятые плакаты, если нельзя нарушать установленное ограничение.

Требуется написать программу, которая по заданной начальной красочности плакатов и последовательности их изменений определяет в начале, а также после каждого изменения, какой максимальной суммарной красочности поднятых плакатов можно добиться, не нарушая условие, что поднято не более трёх плакатов подряд.

Формат входных данных

Первая строка содержит целое число n ($4 \leqslant n \leqslant 40\,000$) — количество друзей.

Вторая строка содержит n целых чисел a_i ($0 \leqslant a_i \leqslant 10^9$) — исходные значения красочности плакатов у друзей.

Третья строка содержит одно целое число q ($0 \leqslant q \leqslant 40\,000$) — количество изменений плакатов, которые вносили друзья.

Каждая из следующих q строк содержит два целых числа p_i и v_i ($1 \leqslant p_i \leqslant n$; $0 \leqslant v_i \leqslant 10^9$) — номер друга, плакат которого изменился, и новая красочность этого плаката.

Формат выходных данных

Выведите $q+1$ число. Перед первым изменением, а также после каждого изменения плакатов выведите одно целое число — максимальное суммарное значение красочности поднятых плакатов, если нельзя поднимать больше трёх плакатов подряд.

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	11	$n \leqslant 10,\ q = 0$	
2	12	$n \leqslant 10,\ q \leqslant 10$	1
3	13	$n \leqslant 1\,000,\ q \leqslant 1\,000$	1, 2
4	17	$n \leqslant 40\,000,\ q = 0$	1
5	47	$n \leqslant 40\,000,\ q \leqslant 40\,000$	1, 2, 3, 4

Пример

стандартный ввод	стандартный вывод
6 1 2 3 4 5 6 2 6 0 2 5	17 13 15

Пояснение к примеру

Перед первым изменением плакаты следует поднять друзьям с номерами $2, 4, 5, 6$. Суммарная красочность поднятых плакатов будет равняться 17.

После первого изменения плакат друга с номером 6 имеет красочность 0. Теперь плакаты следует поднять друзьям с номерами $1, 3, 4, 5$. Суммарная красочность будет равняться 13.

После второго изменения плакат друга с номером 2 имеет красочность 5. Плакаты следует поднять друзьям с номерами $1, 2, 4, 5$. Суммарная красочность будет равняться 15.

Задача 9. Шпион, выйди вон! (7-8 классы)

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

Шпион Васечкин любил свою работу. Каждый раз, приходя на свой суперсекретный завод, он старался пройти незамеченным мимо бдительного вахтёра Пасечкина, который всегда строго отмечал в своём журнале приход и уход каждого сотрудника завода.

Вахтёр Пасечкин тоже любил свою работу. В своём журнале учёта он записывал номера сотрудников и твёрдо помнил, что каждый из них в течение смены обязан отметиться ровно k раз.

Но в тот день что-то пошло не так. Каким образом шпиону Васечкину удалось прошмыгнуть мимо Пасечкина — никому не известно. Но факт остаётся фактом: один из сотрудников завода отметился в журнале учёта меньше, чем k раз.

Ваша задача — определить номер сотрудника-шпиона Васечкина, который отметился в журнале вахтёра Пасечкина менее k раз.

Формат входных данных

В первой строке записаны два целых числа n и k — количество записей в журнале учёта и количество обязательных отметок о приходе и уходе сотрудников ($3 \leqslant n \leqslant 10^6$; $2 \leqslant k < n$). Вторая строка содержит n целых чисел в диапазоне от 1 до 10^9 — номера сотрудников, которые отметились в журнале вахтёра. Каждое число записано ровно k раз, за исключением одного числа, которое встречается в строке менее k раз.

Формат выходных данных

Выведите одно число — номер сотрудника-шпиона, который отметился в журнале менее k раз.

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	$3 \leqslant n \leqslant 10^5$, $k = 2$	
2	20	$3 \leqslant n \leqslant 10^5$, $k = 3$	1
3	20	$3 \leqslant n \leqslant 10^5$, $k = 4$	1, 2
4	20	$3 \leqslant n \leqslant 10^5$, $2 \leqslant k < n$	1, 2, 3
5	20	$3 \leqslant n \leqslant 10^5$, $2 \leqslant k < n$	1, 2, 3, 4

Пример

стандартный ввод	стандартный вывод
5 2 1 8 13 1 8	13

Задача 10. Сломанный калькулятор (7-8 классы)

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

Для чего нужен чердак старого дома? Конечно, для того чтобы хранить всякий ненужный хлам. Вот сегодня я обнаружил там сломанный калькулятор. Старый, добрый калькулятор в пластмассовом корпусе, увы, уже не работал, хотя некоторые признаки жизни ещё подавал.

В калькуляторе работали только две кнопки. С помощью этого устройства можно было делать только два действия — умножать на любое целое положительное число и извлекать корень из числа, то есть из числа n можно было получить $\sqrt{n}$ при условии, что $\sqrt{n}$ — целое число. Если это оказывалось не так, калькулятор зависал, и его приходилось включать заново.

Выбрасывать находку было жалко. Может, сделать из этого антикварного прибора головоломку? Например, для заданного целого положительного n найти наименьшее число, которое можно получить из него с помощью указанных операций.

Ваша задача — определить это наименьшее число, а также минимальное количество операций, необходимых для получения этого наименьшего числа.

Формат входных данных

Входные данные содержат целое число n ($1 \leqslant n \leqslant 2 \cdot 10^9$).

Формат выходных данных

Выведите два числа — наименьшее положительное число, которое можно получить из данного, и минимальное количество операций для получения этого числа.

Примеры

стандартный ввод	стандартный вывод
50	10 2
72	6 3

Пояснение к примеру

В первом примере из числа 50 можно получить наименьшее число 10 за 2 операции: $50 \to 50 \times 2 = 100 \to \sqrt{100} = 10$. Во втором примере из числа 72 можно получить наименьшее число 6 за 3 операции: $72 \to 72 \times 18 = 1296 \to \sqrt{1296} = 36 \to \sqrt{36} = 6$.

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	30	$1 \leqslant n \leqslant 100$	
2	30	$1 \leqslant n \leqslant 10^6$	1
3	40	$1 \leqslant n \leqslant 2 \cdot 10^9$	1, 2

2020-2021 учебный год

Муниципальный этап 33-й Всероссийской олимпиады по информатике среди школьников Республики Татарстан состоялся 24 ноября 2020 г. На олимпиаде школьникам 7-11 классов предлагается решить 4-5 задач. Как правило, первые две задачи в списке заданий муниципального этапа опираются, в основном, на логику алгоритмического мышления и не требуют «профессиональных» олимпиадных навыков. Следующие, более сложные задачи охватывают достаточно большой спектр различных стандартных и нестандартных алгоритмов.

Региональный этап олимпиады проходил с 16 по 18 января 2021 г. Кроме школьников 9-11 классов — традиционных участников регионального этапа — на олимпиаду приглашаются также ученики 7-8 классов, показавшие хорошие результаты на муниципальном этапе Всероссийской олимпиады. Специально для школьников 7-8 классов жюри составляет 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 `Pascal`. Для большинства олимпиадных заданий указаны фамилии авторов идеи и фамилии тех, кто готовил эти решения.

Муниципальный этап, 2020-2021

Задача 1. Подарки Деда Мороза (7-11 классы)

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

Скоро, скоро Новый год!

И у Деда Мороза полно дел — ему нужно приготовить подарки. В мешке у Деда a пряников, b конфет и c мандаринов. В каждый подарок нужно положить ровно два неодинаковых угощения — один пряник и одну конфету, или один пряник и один мандарин, или одну конфету и один мандарин.

Вам необходимо подсчитать, сколько подарков с двумя неодинаковыми угощениями сможет приготовить Дед Мороз.

Формат входных данных

В строке записаны три целых числа a, b и c — количества пряников, конфет и мандаринов соответственно ($0 \leqslant a, b, c \leqslant 10^9$).

Формат выходных данных

Выведите одно число — количество подарков, которые сможет приготовить Дед Мороз.

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	$a = 0$	
2	30	$a = b = c$	1
3	50	$0 \leqslant a, b, c \leqslant 10^9$	1, 2

Примеры

стандартный ввод	стандартный вывод
1 2 3	3

Пояснение к примеру

В примере 1 пряник, 2 конфеты и 3 мандарина, и Дед Мороз сможет подготовить всего три подарка — два подарка, в которых по одной конфете и одному мандарину, и один подарок с пряником и мандарином.

Задача 2. Больше-меньше (7-8 классы)

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

У Васи есть комплект из $n+1$ карточек с различными натуральными числами, по одному числу на каждой карточке. Вася любит раскладывать карточки на бумаге, а затем вписывать между каждыми двумя числами на *соседних* карточках знак < или > так, чтобы получились верные неравенства. Вот и сегодня он в очередной раз разложил карточки в ряд и записал значки < или > между ними. Неожиданный телефонный звонок отвлёк Васю, и он не заметил, как его младший брат перемешал все карточки, так что остались только n записанных на бумаге значков < и >.

Сможете ли вы помочь Васе и снова расставить карточки в таком порядке, чтобы все записанные на бумаге неравенства остались верными?

Формат входных данных

В первой строке записано одно число n — количество символов < и >, которые успел написать Вася на бумаге. Во второй строке записаны $n+1$ различных целых чисел a_i ($1 \leqslant a_i \leqslant 10^9$) из исходного комплекта карточек. В третьей строке записана строка из n символов < и >.

Формат выходных данных

Запишите $n+1$ данных чисел a_i в таком порядке, чтобы при записи между ними соответствующих символов < и > все получаемые неравенства оказались верными. Если это сделать невозможно, запишите -1. Если требуемых расстановок карточек 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	15	$1 \leqslant n \leqslant 2$	
2	20	$1 \leqslant n \leqslant 10$	1
3	25	$1 \leqslant n \leqslant 100$	1, 2
4	40	$1 \leqslant n \leqslant 10^5$	1, 2, 3

Пример

стандартный ввод	стандартный вывод
4 7 1 5 2 4 <><>	2 5 4 7 1

Пояснение к примеру

Расстановка чисел данного массива в указанном порядке удовлетворяют условию: 2 < 5 > 4 < 7 > 1, все символы < и > между числами образуют заданную в условии строку <><>.

Задача 3. Отмерь и отрежь (7-11 классы)

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

«Семь раз отмерь, один раз отрежь». Эта пословица заставляет нас обдумывать свои действия, принимать правильные и тщательно отмеренные взвешенные решения. Смысл этой мудрости заключается в том, что прежде надо думать, а потом делать(или не делать).

Так случилось и сегодня. Электрику Петрову нужно было нарезать куски провода длиной $l_1, l_2, \ldots, l_n$. Но то ли день не задался, то ли звёзды сложились неудачно, но кто-то уже разрезал провод на (несколько) m частей. И теперь Петрову нужно разобраться, можно ли из этих частей вырезать куски требуемой длины.

Другими словами, он должен быть уверен, что это сделать можно, и теперь ему необходимо вычислить такую *наименьшую* длину исходного провода, что при любом способе его разрезания на m частей гарантировано удастся вырезать n кусков с длинами $l_1, l_2, \ldots, l_n$.

Формат входных данных

В первой строке записаны два целых числа: m — количество частей, на которые был разрезан провод ($2 \leqslant m \leqslant 10^5$), и n — количество кусков, которые нужно получить после разрезания провода на m частей ($1 \leqslant n \leqslant 10^5$).

Вторая строка содержит n целых чисел в диапазоне от 1 до 10^6 — длины кусков провода, которые требуется получить.

Формат выходных данных

Выведите одно число — наименьшую длину провода, из которого после разрезания любым способом на m частей можно получить куски требуемой длины.

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	5	$m = 2,\ n = 1$	
2	10	$m = 2,\ 1 \leqslant n \leqslant 2$	1
3	20	$m = 2,\ 1 \leqslant n \leqslant 10^5$	1, 2
4	25	$2 \leqslant m \leqslant 3,\ 1 \leqslant n \leqslant 10^5$	1, 2, 3
5	40	$2 \leqslant m \leqslant 10^5, 1 \leqslant n \leqslant 10^5$	1, 2, 3, 4

Примеры

стандартный ввод	стандартный вывод
2 4 2 1 2 3	9

Задача 4. Самое красивое число (7-11 классы)

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

Красивыми суммами называют суммы нескольких подряд идущих положительных целых чисел. Например, суммы $7 + 8$ и $4+5+6$ — красивые, а сумма $3+5+7$ — некрасивая, хотя результат суммирования во всех случаях равен 15. (Сумма из одного слагаемого 15 тоже считается красивой.) Исходя из этого, *красотой* целого положительного числа будем называть количество представлений этого числа в виде красивых сумм. Например, красота числа 15 равна 4, поскольку 15 представляется в виде красивых сумм четырьмя способами: $15 = 7+8 = 4+5+6 = 1+2+3+4+5$.

Из двух целых чисел более красивым считается то, у которого больше представлений в виде красивых сумм. При равенстве количеств таких представлений предпочтение в красоте отдаётся меньшему из них. Например, у чисел 15 и 30 красота одинаковая (равна 4), однако более красивым считается число 15.

Вам необходимо составить программу, которая в заданном наборе целых положительных чисел находит самое красивое число и определяет его красоту.

Формат входных данных

В первой строке записано целое n — количество чисел в наборе ($2 \leqslant n \leqslant 10^3$). Во второй строке содержится n целых положительных чисел a_i, каждое из которых не превосходит $2 \cdot 10^9$.

Формат выходных данных

Выведите два целых числа — самое красивое из всех чисел набора и значение его красоты.

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	30	$2 \leqslant n \leqslant 3,\ \ 1 \leqslant a_i \leqslant 100$	
2	30	$2 \leqslant n \leqslant 10$	1
3	40	$2 \leqslant n \leqslant 10^3$	1, 2

Примеры

стандартный ввод	стандартный вывод
2 15 30	15 4
3 20 30 20	30 4

Задача 5. Бермудский треугольник (9-11 классы)

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

Где-то в Атлантическом океане находится таинственный Бермудский треугольник, в котором происходят мистические исчезновения морских и воздушных судов и другие аномальные явления. Два корабля располагаются в различных точках A и B океана и с помощью специальных приборов пытаются установить координаты вершин этого треугольника. Для каждого корабля удалось определить точное расстояние до прямых, на которых лежат стороны Бермудского треугольника. И теперь вам необходимо обработать полученный массив данных и выяснить координаты его вершин.

Морские корабли могут располагаться как внутри, так и вне треугольника, но не могут располагаться на его сторонах. Расстояния от кораблей внутри треугольника считаются положительными, а для кораблей вне треугольника — отрицательными.

Формат входных данных

В двух первых строках записаны координаты корабля A и расстояния от него до прямых, на которых лежат стороны XY, YZ и ZX Бермудского треугольника XYZ. В следующих двух строках — координаты корабля B и расстояния до тех же прямых XY, YZ и ZX. Координаты A и B — целые числа, не превосходящие по абсолютной величине 10^4. Все расстояния — целые ненулевые числа, не превосходящие по абсолютной величине 10^4. Гарантируется, что треугольник XYZ существует и координаты его вершин не превосходят по абсолютной величине 10^9.

Формат выходных данных

В трёх строках запишите через пробел координаты вершин X, Y и Z Бермудского треугольника. Верным считается любой ответ, где для точек A и B расстояния до сторон найденного треугольника отличаются от требуемых не более, чем на 10^{-5}.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Обратите внимание: в подзадачах указаны ограничения на одно из возможных решений задачи. В тестах этих подзадач возможны также решения с другими значениями углов треугольника.

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	20	$\angle X = 90°$	
2	20	$\angle Y = 90°$	1
3	20	$\angle Z = 90°$	1, 2
4	40	нет	1, 2, 3

Примеры

стандартный ввод	стандартный вывод
1 5 1 5 5 5 2 5 5 2	0.00000 0.00000 0.00000 12.0000 16.0000 0.00000
5 5 5 5 5 5 -5 -5 -13 -5	0.00000 0.00000 20.0000 0.00000 0.00000 15.0000

Региональный этап, 2020-2021

Региональный этап Всероссийской олимпиады школьников состоялся 16-18 января 2021 года. Соревнование проходило в два тура с перерывом (17 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 Николай Будин, Ильдар Гайнуллин, Дмитрий Гнатюк, Арсений Кириллов, Даниил Орешников, Михаил Путилин, Андрей Станкевич.

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. Два станка

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

На производстве имеется два станка. Необходимо изотовить как можно больше деталей за сегодняшнюю смену, продложительность которой k минут.

Станки находятся в законсервированном состоянии. Для того, чтобы ввести в строй первый станок, требуется a минут, после чего он будет производить x деталей в минуту. Для того, чтобы ввести в строй второй станок, требуется b минут, после чего он будет производить y деталей в минуту.

Для введения в строй станка требуется присутствие инженера, поэтому нельзя вводить в строй два станка одновременно. При этом введение станка в строй и изготовление деталей на другом станке, а также одновременное изготовление деталей на двух станках разрешается.

Требуется выяснить, какое максимальное количество деталей удастся изготовить за k минут.

Формат входных данных

В первой строке ввода дано единственное целое неотрицательное число k — количество минут в смене $(0 \leqslant k \leqslant 10^9)$.

Во второй строке ввода даны целые неотрицательные числа a и x – время введения первого станка в строй и количество деталей, которое он изготавливает за одну минуту $(0 \leqslant a, x \leqslant 10^9)$.

В третьей строке ввода даны целые неотрицательные числа b и y — время введения второго станка в строй и количество деталей, которое он изготавливает за одну минуту $(0 \leqslant b, y \leqslant 10^9)$.

Формат выходных данных

Выведите единственное число — максимальное количество деталей, которое удастся изготовить за смену.

Примечание

Обратите внимание, что ответ в этой задаче может быть довольно большим и не помещаться в 32-битные типы данных.

Рекомендуется использовать 64-битный тип данных, например «`long long`» в C++ или «`int64`» в Паскале.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Для подзадач 1 и 2 предоставляется полная информация о прохождении тестов, для остальных подзадач — информация «Первая ошибка».

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	17	$a = 0,\ x = 0$	
2	14	$a = 0,\ b = 0$	
3	20	$a = b$	2
4	20	$x = y$	
5	29	нет	1, 2, 3, 4

Пример

стандартный ввод	стандартный вывод
20 10 4 5 3	65

Пояснение к примеру

В примере выгодно сначала ввести в строй второй станок и за оставшиеся 15 минут изготовить 45 деталей, а затем ввести в строй первый и за оставшиеся 5 минут изготовить на нём еще 20 деталей.

Если сначала ввести в строй первый станок и изготовить на нем в оставшиеся 10 минут 40 деталей, то после введения в строй второго на нем удастся изготовить лишь 15 деталей, суммарно 55, что меньше, чем 65.

Задача 2. Разбиение таблицы

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

Рассмотрим таблицу из n строк и m столбцов, в клетки которой по строкам записаны числа от 1 до $n \cdot m$. Сначала заполняется первая строка слева направо, затем вторая, и так далее. Другими словами в клетку (r, c) записано число $(r-1) \cdot m + c$.

Приведём пример такой таблицы для $n = 3$, $m = 5$.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

Требуется разделить таблицу одним вертикальным или горизонтальным разрезом, проходящим по сторонам клеток, так чтобы сумма чисел в получившихся частях таблицы отличалась как можно меньше. В этой задаче в одном тесте вам придётся ответить на несколько запросов об оптимальном разрезании таблицы.

Формат входных данных

В первой строке ввода задано целое число t — количество запросов ($1 \leqslant t \leqslant 10^5$).

В следующих t строках заданы по два целых числа n и m ($1 \leqslant n, m \leqslant 10^9$, $2 \leqslant n \cdot m \leqslant 10^9$).

Формат выходных данных

В t строках выведите ответы на запросы, по одному на строке.

Ответ на каждый запрос должен быть выведен в формате «`D` x», где `D` — это «`V`», если нужно резать по вертикали, «`H`» — если по горизонтали, а x — номер столбца или строки, перед которым надо сделать разрез. Строки пронумерованы от 1 до n, столбцы пронумерованы от 1 до m.

Если правильных ответов несколько, то надо вывести вариант с вертикальным разрезом, если он есть, а если и после этого вариантов несколько, то из вариантов с различными x следует выбрать тот, в котором x меньше.

Пример

стандартный ввод	стандартный вывод
5 1 3 4 7 1 10 3 3 3 5	V 3 V 5 V 8 H 3 V 4

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Для подзадачи 1 предоставляется полная информация о прохождении тестов, для остальных подзадач — информация «Первая ошибка».

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	20	$t = 1,\ 1 \leqslant n, m \leqslant 100.$	
2	14	$t = 1,\ 1 \leqslant n, m \leqslant 2\,000.$	1
3	15	$t = 1,\ 1 \leqslant n, m \leqslant 10^7.$	1, 2
4	16	$1 \leqslant t \leqslant 1\,000,$ $1 \leqslant n \cdot m \leqslant 10\,000.$	1
5	15	$1 \leqslant t \leqslant 100\,000,$ $n = 1,\ 1 \leqslant m \leqslant 10^9.$	
6	20	$1 \leqslant t \leqslant 100\,000,$ $1 \leqslant n, m \leqslant 10^9.$	1–5

Задача 3. Изменённая ДНК

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

Биологи обнаружили новый живой организм и решили изучить его ДНК. ДНК кодируется последовательностью символов «A», «G», «C» и «T».

Так как строка, кодирующая ДНК, часто очень длинная, для её хранения применяют RLE-кодирование. А именно, каждый блок, состоящий из двух или более идущих подряд одинаковых символов, заменяется на число, равное длине этого блока, после которого записывается соответствующий символ. Например, последовательность «AAAGGTCCA» в закодированной форме имеет вид «3A2GT2CA».

В результате экспериментов, проводимых в лаборатории, ДНК может мутировать. Каждая мутация — это либо удаление одного символа из последовательности, либо добавление одного символа, либо замена одного символа на другой.

Уходя вечером из лаборатории, учёный записал ДНК в закодированной форме. Когда он вернулся на работу утром, он обнаружил, что в ДНК произошла ровно одна мутация. Теперь ученых интересует, какая минимальная и максимальная длина может получиться у новой ДНК в закодированной форме.

Требуется по заданной ДНК в закодированной форме определить, какая мутация может привести к тому, что у новой ДНК будет закодированная форма минимальной возможной длины, а какая — к тому, что у новой ДНК будет закодированная форма максимальной возможной длины.

Формат входных данных

В единственной строке входа находится строка s, состоящая из цифр и букв «A», «G», «C» и «T» — закодированная ДНК.

Гарантируется, что это строка является корректной закодированной записью некоторой строки из символов «A», «G», «C» и «T».

Формат выходных данных

В первой строке выведите мутацию, после которой закодированная строка имеет минимальную длину. Выведите:

- `1` x `Z`, если надо вставить символ `Z` так, чтобы слева от него было ровно x старых символов. Символ `Z` должен быть из множества {`A`, `C`, `G`, `T`}.

- `2` x, если надо удалить символ с номером x из последовательности.
- `3` x `Z`, если надо заменить символ с номером x заменить на символ `Z`. Символ `Z` должен быть из множества {`A`, `C`, `G`, `T`}. При этом на этом месте до мутации обязательно должен был находиться символ, не равный `Z`.

В следующей строке выведите мутацию, после которой закодированная строка имеет максимальную длину, в таком же формате.

Если подходящих ответов несколько, можно вывести любой из них.

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Для подзадачи 1 предоставляется полная информация о прохождении тестов, для остальных подзадач — информация «Первая ошибка».

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	9	$1 \leqslant n \leqslant L \leqslant 10$	
2	17	$1 \leqslant n \leqslant 100,\ \ 1 \leqslant L \leqslant 10^4$	1
3	21	$1 \leqslant n \leqslant 1000,\ 1 \leqslant L \leqslant 10^5$	1-2
4	11	$1 \leqslant n \leqslant 10^5,\ 1 \leqslant L \leqslant 10^7$	1–3
5	42	$1 \leqslant n \leqslant 10^5,\ 1 \leqslant L \leqslant 10^9$	1–4

Пример

стандартный ввод	стандартный вывод
5AC5A2C	3 6 A 1 2 C

Пояснение к примеру

Исходная последовательность имела вид «AAAAACAAAAACC».

Первая операция превращает её в последовательность

«AAAAAAAAAAACC»,

которая кодируется как «11A2C». Эта закодированная последовательность имеет минимальную возможную для этого теста длину, равную 5.

Вторая операция превращает её в последовательность

«AACAAACAAAAACC»,

которая кодируется как «2AC3AC5A2C». Эта закодированная последовательность имеет максимальную возможную для этого теста длину, равную 10.

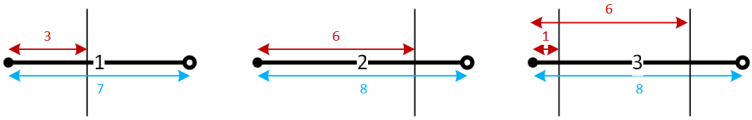
Задача 4. Антенна

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

Для связи с Землёй членам экспедиции на Марс необходимо собрать антенну. Антенна в разобранном состоянии представляет собой n фрагментов, i-й фрагмент представляет собой штангу длиной s_i сантиметров, на которой закреплены m_i перекладин. Каждый фрагмент содержит хотя бы одну перекладину.

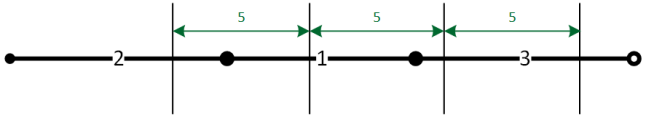
У каждой штанги есть начало, в котором расположен штекер, и конец, в котором расположено гнездо. Любые две штанги можно последовательно соединить, присоединив начало одной к концу другой. Для каждой перекладины известно расстояние от начала её штанги в сантиметрах. Для i-го фрагмента это расстояние может быть от 0 до s_i, значение 0 означает, что перекладина находится непосредственно в начале штанги, значение s_i — что она находится непосредственно в конце штанги. Толщиной перекладин и размерами штекера и гнезда следует пренебречь.

На рисунке показаны три фрагмента антенны из первого примера и отмечены расстояния от начала штанги до перекладины.



Чтобы корректно собрать антенну, необходимо соединить в некотором порядке все n фрагментов, при этом расстояние между любыми двумя соседними перекладинами должно быть одинаковым.

На рисунке показан корректный способ соединить фрагменты в первом примере.



К сожалению, члены экспедиции забыли инструкцию по сборке антенны на Земле, а передать её на Марс не представляется возможным — ведь антенна ещё не собрана. Помогите исследователям!

Требуется определить, в каком порядке необходимо соединить фрагменты антенны, чтобы установить связь с Землей.

Формат входных данных

В первой строке дано одно число n — количество фрагментов ($1 \leqslant n \leqslant 100\,000$).

Далее дано описание n фрагментов. В первой строке описания фрагмента даны два целых числа m_i и s_i — количество перекладин и длина штанги в i-м фрагменте ($1 \leqslant m_i \leqslant 100\,000$, $0 \leqslant s_i \leqslant 10^9$). В следующей строке даны m_i целых чисел $p_{i,j}$ — позиции перекладин, $p_{i,j}$ равно расстоянию в сантиметрах от начала штанги до j-й перекладины на ней ($0 \leqslant p_{i,1} < p_{i,2} < \ldots < p_{i,m_i} \leqslant s_i$).

Сумма всех m_i не превышает 100 000.

Формат выходных данных

Если собрать антенну указанным образом возможно, в первой строке выведите «`Yes`», а во второй строке выведите перестановку чисел от 1 до n — номера фрагментов в порядке, в котором

их следует соединить, начало каждого следующего фрагмента в этом порядке присоединяется к концу предыдущего фрагмента. Если существует несколько подходящих ответов, можно вывести любой из них.

Если собрать антенну невозможно, в единственной строке выведите «`No`».

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	8	$n \leqslant 8$, $m_i = 1$, $s_i \leqslant 100$	
2	8	$n \leqslant 8$, $s_i \leqslant 100$	1
3	21	$n \leqslant 1\,000$	1-2
4	21	$\sum m_i > n$	
5	21	$s_i \leqslant 100$	1–2
6	21	нет	1–5

Примеры

стандартный ввод	стандартный вывод
3 1 7 3 1 8 6 2 8 1 6	Yes 2 1 3
1 1 7 5	Yes 1
1 3 10 2 5 9	No
3 1 5 3 1 3 3 1 6 3	No
4 1 5 0 1 0 0 1 3 3 1 0 0	Yes 3 2 4 1

Задача 5. Календарь на Альфе Центавра

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

На планете в системе Альфы Центавра год состоит из m месяцев, пронумерованных от 1 до m, а каждый месяц из d дней, пронумерованных от 1 до d. В свою очередь неделя у поселенцев на этой планете состоит из w дней, проиндексированных строчными английскими буквами, от «`a`» до w-й буквы английского алфавита.

Первый день первого месяца первого года соответствует букве «`a`».

Требуется определить, какой букве будет соответствовать i-й день j-го месяца k-го года.

Формат входных данных

Первая строка содержит три целых числа d, m и w ($1 \leqslant d, m \leqslant 100$, $1 \leqslant w \leqslant 26$).

Вторая строка содержит три целых числа i, j и k ($1 \leqslant i \leqslant d$, $1 \leqslant j \leqslant m$, $1 \leqslant k \leqslant 10^9$).

Формат выходных данных

Выведите одну строчную букву английского алфавита — какой букве соответствует i-й день j-го месяца k-го года.

Пример

стандартный ввод	стандартный вывод
30 12 7 18 1 2021	b

Замечание

Обратите внимание, при решении этой задачи рекомендуется использовать 64-битные типы данных, например «`long long`» в `C++`, «`int64`» в Паскале.

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	16	$d = 1, m = 1$	
2	16	$m = 1,\ k \leqslant 10^7$	1
3	17	$i = 1,\ j = 1$	
4	17	$k = 1$	
5	17	$k \leqslant 100$	4
6	17	нет	1–5

Задача 6. Числа

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

Аня любит, когда числа состоят из одинаковых цифр. Поэтому ей нравятся числа 777 или 5555, а вот число 1234 ей совсем не нравится.

Иногда у Ани бывает хорошее настроение, тогда ей по прежнему нравятся все числа, состоящие из одинаковых цифр, но также нравятся числа, в которых все цифры кроме одной одинаковые, как, например, в числе 77727.

У Ани есть число x. Аня хочет найти минимальное целое число $y \geqslant x$, которое ей понравится.

Требуется написать программу, которая по заданному целому числу x и информации, хорошее ли настроение у Ани, находит минимальное целое число $y \geqslant x$, которое нравится Ане.

Формат входных данных

Первая строка ввода содержит целое число x ($1 \leqslant x \leqslant 10^{17}$, обратите внимание, что число x не может быть сохранено в стандартном 32-битном типе данных, необходимо использовать 64-битный тип данных, например «`long long`» в C++, «`int64`» в Паскале).

Вторая строка ввода содержит число k, равное 0 или 1. Значение $k = 1$ означает, что у Ани хорошее настроение, а значение $k = 0$ — что это не так.

Формат выходных данных

Следует вывести одно целое число y, для которого должны выполняться следующие свойства:

- $y \geqslant x$;
- если $k = 0$, то все цифры в десятичной записи числа y должны совпадать;
- если $k = 1$, то все цифры в десятичной записи числа y, кроме, может быть, одной, должны совпадать.

Примеры

стандартный ввод	стандартный вывод
700 0	777
700 1	700

Система оценивания

Баллы за каждую подзадачу начисляются только в случае, если все тесты для этой подзадачи и необходимых подзадач успешно пройдены. Для подзадач 1 и 3 предоставляется полная информация о прохождении тестов, для остальных подзадач — информация «Первая ошибка».

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	15	$1 \leqslant x \leqslant 10^5,\ k = 0$	
2	20	$1 \leqslant x \leqslant 10^{17},\ k = 0$	1
3	21	$1 \leqslant x \leqslant 10^5,\ 0 \leqslant k \leqslant 1$	1
4	44	$1 \leqslant x \leqslant 10^{17},\ 0 \leqslant k \leqslant 1$	1–3

Задача 7. Хорошие раскраски

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

Назовем раскраску клеток таблицы $n \times m$ *хорошей*, если никакие четыре клетки, центры которых образуют вершины прямоугольника со сторонами, параллельными осям координат, не покрашены в один цвет.

Иначе говоря, для раскраски не должно быть четверки целых чисел x_1, x_2, y_1, y_2 таких, что $1 \leqslant x_1 < x_2 \leqslant n$, $1 \leqslant y_1 < y_2 \leqslant m$, и клетки (x_1, y_1), (x_2, y_1), (x_1, y_2) и (x_2, y_2) покрашены в одинаковый цвет.

Требуется написать программу, которая по заданным целым числам n, m и c находит любую хорошую раскраску таблицы $n \times m$ в c цветов.

Формат входных данных

В первой строке записаны три целых числа n, m, c ($2 \leqslant n, m \leqslant 10$, $2 \leqslant c \leqslant 3$).

Гарантируется, что для заданных входных данных существует хотя бы одна хорошая раскраска.

Формат выходных данных

Выведите n строк по m чисел в каждой.

В качестве j-го числа i-й строки выведите $a_{i,j}$ — цвет клетки (i, j) $(1 \leqslant a_{i,j} \leqslant c)$.

Если есть несколько хороших раскрасок, можно вывести любую из них.

Система оценивания

Кроме теста из примера в этой задаче 20 тестов, каждый независимо оценивается в 5 баллов. В пяти тестах значение $c = 2$ и в пятнадцати тестах значение $c = 3$.

Для каждого теста сообщается результат проверки на этом тесте.

Пример

стандартный ввод	стандартный вывод
2 2 2	1 2 2 2

Задача 8. A + B

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

Рассмотрим a, b и c — целые неотрицательные числа, записанные в десятичной системе счисления. Пусть они имеют одинаковую длину n, при этом запись может начинаться с нуля. Числа записаны одно под другим, цифры расположены в три строки и n столбцов. Рассмотрим пример такой записи:

```
01211
12099
23300
```

Требуется переставить столбцы в этой записи таким образом, чтобы выполнялось равенство $a + b = c$. В полученной записи ведущие нули уже запрещены. Сколько существует различных способов это сделать?

Перестановки столбцов считаются различными, даже если полученные записи совпадают. Например, если в записи выше переставить два последних столбца, получится другая перестановка, хотя цифры в этих колонках совпадают.

Поскольку ответ может быть довольно большим, требуется посчитать для него остаток по модулю $10^9 + 7$.

Формат входных данных

Во входных данных записаны целые неотрицательные числа a, b и c по одному в строке. Каждое число состоит из n десятичных цифр и может начинаться с нуля ($2 \leqslant n \leqslant 2 \cdot 10^5$).

Формат выходных данных

Выведите количество подходящих перестановок столбцов по модулю $10^9 + 7$.

Примеры

стандартный ввод	стандартный вывод
123 123 246	6
01 02 03	1
01211 12099 23300	4
121 214 999	0

Пояснение к примерам

В первом примере подходят все перестановки столбцов.

Во втором примере единственная подходящая перестановка — $10 + 20 = 30$. $01 + 02 = 03$ не считается из-за наличия ведущих нулей.

В третьем примере возможны варианты 10121+21909 = 32030 и 12101 + 20919 = 33020, причём каждый из них может быть получен двумя разными перестановками.

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	7	$2 \leqslant n \leqslant 6$	
2	14	$2 \leqslant n \leqslant 18$	1
3	15	$2 \leqslant n \leqslant 200$, нет цифры 0	
4	5	$2 \leqslant n \leqslant 200$	1–3
5	17	$2 \leqslant n \leqslant 750$, нет цифры 0	3
6	5	$2 \leqslant n \leqslant 750$	1–5
7	20	$2 \leqslant n \leqslant 10^5$, нет цифры 0	3, 5
8	17	$2 \leqslant n \leqslant 10^5$	1–7

Задача 9. Сумма разностей (7-8 классы)

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

У вас есть n целых чисел a_1, a_2, $\ldots a_n$. Для каждой пары чисел a_i и a_j $(j > i)$ вычислим их разность, вычитая из большего числа меньшее.

Вам необходимо подсчитать сумму всех получившихся разностей. Другими словами, найдите сумму

$$\sum_{i=1}^{n-1}\sum_{j>i}^{n} |\, a_i - a_j \,|.$$

Формат входных данных

В первой строке записано целое число n, равное количеству чисел a_i ($2 \leqslant n \leqslant 2 \cdot 10^5$). Во второй строке записаны n целых чисел $a_1, a_2, \ldots a_n$ ($-10^8 \leqslant a_i \leqslant 10^8$).

Формат выходных данных

Запишите одно число — требуемую сумму.

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	30	$2 \leqslant n \leqslant 10^2$, $\lvert a_i \rvert \leqslant 10^3$	
2	30	$2 \leqslant n \leqslant 10^4$, $\lvert a_i \rvert \leqslant 10^8$	1
3	40	$2 \leqslant n \leqslant 10^5$, $\lvert a_i \rvert \leqslant 10^8$	1–2

Пример

стандартный ввод	стандартный вывод
3 5 -1 2	12

Пояснение к примеру

Значение суммы вычисляется так: $|5-(-1)|+|5-2|+|(-1)-2| = 12$.

Задача 10. Полупростые делители (7-8 классы)

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

Целое положительное число называется *полупростым*, если оно является произведением двух простых чисел (возможно равных между собой). Например, число $10 = 2 \cdot 5$ — полупростое, а число $12 = 2 \cdot 2 \cdot 3$ — нет. Последовательность полупростых чисел начинается так: 4, 6, 9, 10, 14, 15, 21, 22, 25, ...

Для заданного целого числа m найдите *наименьшее* натуральное число, у которого ровно m полупростых делителей.

Формат входных данных

В единственной строке записано число m ($0 \leqslant m \leqslant 10^6$).

Формат выходных данных

Запишите искомое число по модулю $(10^9 + 7)$.

Система оценивания

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

Подзадача	Баллы	Ограничения	Необходимые подзадачи
1	5	$0 \leqslant m \leqslant 7$	
2	10	$0 \leqslant m \leqslant 10^2$	1
3	20	$0 \leqslant m \leqslant 10^3$	1–2
4	25	$0 \leqslant m \leqslant 10^4$	1–3
5	40	$0 \leqslant m \leqslant 10^6$	1–4

Пример

стандартный ввод	стандартный вывод
2	12

Пояснение к примеру

Число 12 — это наименьшее число, у которого ровно 2 полупростых делителя $4 = 2 \cdot 2$ и $6 = 2 \cdot 3$.

Решения задач

Задача 1. Хамелеоны

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

Основные темы этой простой задачи — логические операции, целочисленное деление.

Если хамелеонов какого-нибудь цвета, скажем, белого, — чётное количество, то, разбивая их на пары, добьёмся, что все они превратятся в чёрных хамелеонов. Таким образом, все хамелеоны острова будут чёрного цвета.

Если же хамелеонов каждого цвета — нечётное число, то все хамелеоны не смогут окраситься в один цвет. Действительно, после каждой встречи число хамелеонов белого или чёрного цвета либо не меняется, либо изменяется на 2, то есть остаётся *нечётным* числом, и поэтому никогда не станет равным 0.

Таким образом, в задаче нужно проверить чётность чисел b и w. Если они оба нечётные, выводим `no`; если хотя бы одно чётное, выводим `yes`. Кроме того, для корректной работы алгоритма нужно учесть, что b и w должны быть переменными типа long long.

Основной фрагмент кода на языке C++:

```
long long w, b;
cin >> w >> b;

if ((w % 2 == 1) && (b % 2 == 1)) cout "no" << endl;
else cout << "yes" << endl;
```

Задача 2. Два альбома

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

Основные темы задачи — работа с массивами, сортировки.

Для каждого альбома определим массив, в котором будем хранить уникальные номера марок этого альбома. Пусть $a[n]$ и $b[m]$ — массивы из этих номеров. В задаче требуется найти количество общих элементов массивов $a[n]$, $b[m]$ и вывести эти элементы.

Подзадача 1. Простым перебором элементов двух массивов можно набрать 30 баллов.

Подзадача 2. Для нахождения общих элементов можно сравнить каждое число массива $a[n]$ с каждым числом массива $b[m]$. Алгоритмическая сложность такого решения — $O(n \cdot m)$, оно набирает 60 баллов.

Подзадача 3. Объединим массивы $a[n]$ и $b[m]$ в один массив $c[n+m]$ и отсортируем его по возрастанию. Тогда общие элементы исходных массивов будут встречаться в массиве $c[n+m]$ ровно два раза. Осталось пройти по всему массиву и подсчитать количество *соседних* повторяющихся элементов и сами эти элементы.

Другая возможность полного решения — использование функции `set_intersection` из библиотеки шаблонов <algorithm> языка C++. Для этого нужно сначала отсортировать массивы $a[n]$ и $b[m]$, а затем к полученным после сортировки массивам применить указанную функцию.

Все такие решения набирают 100 баллов, их алгоритмическая сложность — $O(n \log n + m \log m)$.

Приведём основной фрагмент кода на языке C++:

```
cin >> n >> m;
vector <int> c(n + m);
for (int i = 0; i < n + m; ++i)
    cin >> c[i];
sort(c.begin(), c.end());
```

```
vector <int> v;
for (int i = 1; i < n + m; ++i)
     if (a[i] == a[i - 1]) v.push_back(a[i]);
cout << v.size() << endl;
for (int i = 0; i < v.size(); ++i)
     cout << v[i] << ' ';
return 0;
```

Задача 3. Факториал

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

Основные темы задачи — теория чисел, делимость, строки.

Подзадача 1. Если длина записи числа $n!$ не больше 3, решить задачу можно перебором значений $n!$, где $1 \leqslant n \leqslant 6$.

Подзадача 2. Предварительно вычислим значения факториалов $n!$ для всех чисел n от 1 до 20. Затем сравним заданную строку с записью каждого вычисленного факториала. Если эти строки отличаются только одним символом, то находим цифру, соответствующую символу #.

Подзадача 3. Если запись числа $n!$ содержит более 20 символов, то для вычисления значений $n!$ можно воспользоваться длинной арифметикой, а затем опять сравнить заданную строку с записью $n!$, как это описано в решении подзадачи 2.

Подзадача 4. Полное решение задачи достаточно простое и не требует дополнительных идей, связанных с предварительным подсчётом значений $n!$ или длинной арифметики.

Пусть $n!$ — записанное на доске число. Отметим, что если $n \geqslant 6$, число $n! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot \ldots \cdot n$ обязательно делится на 9. По признаку делимости сумма цифр записанного на доске числа $n!$ также делится на 9. Отсюда вытекает следующий алгоритм нахождения стёртой цифры.

Подсчитаем сумму всех цифр в строке s (за исключением, разумеется, «#») и найдём остаток r при делении этой суммы на 9. Тогда ненулевая цифра, заменённая символом #, равна $9 - r$.

Если же $n < 6$, запись числа $n!$ содержит не более трёх цифр, из которых одна цифра заменена символом #. Все эти случаи легко решаются простым перебором:

— при $n \leqslant 3$ запись $n!$ состоит из одной цифры, и подойдёт любая из цифр 1, 2 или 6;

— при $n = 4$ ответом будет цифра 2, если заменили первую цифру, или 4, если заменили вторую цифру в записи $n!$;

— при $n = 5$ ответом будет цифра 1, если заменили первую цифру, или 2, если заменили вторую цифру.

Приведём основной фрагмент кода на языке C++:

```
string s;
cin >> s;

for (int i = 0; i < s.length(); ++i) {
   if (s[i] != '#') sum += s[i] - '0';
}
if (s.length() == 1) cout << 2;
if (s.length() == 2) cout << 6 - sum;
if (s.length() == 3)
   if (s[0] == '#') cout << 1; else cout << 2;
if (s.length() > 3) cout << 9 - sum % 9;
return 0;
```

Задача 4. Сортировка мусора

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

Задание представляет собой усложнённый вариант фольклорной задачи Всероссийской олимпиады школьников 2000-2001 г. Основные темы задачи — комбинаторика, генерация перестановок.

Пусть $a[1][1]$, $a[2][1]$, ..., $a[n][1]$ — количество отходов вида 1 в контейнерах с номерами 1, 2, ..., n соответственно. Общее

количество отходов вида 1, очевидно, равно

$$s[1] = a[1][1] + a[2][1] + \ldots + a[n][1] = \sum_{k=1}^{n} a[k][1].$$

Предположим, что после сортировки все отходы вида 1 окажутся в контейнере с номером i_1. Тогда количество операций для перемещения отходов вида 1 в этот контейнер будет равно $s[1] - a[i_1][1]$. Аналогичным образом, подсчитаем количество операций для перемещения отходов вида 2, 3, …, n в контейнеры с номерами i_2, i_3, …, i_n соответственно. Тогда общее число операций, необходимое для сортировки всего мусора, равно

$$s = (s[1] - a[i_1][1]) + (s[2] - a[i_2][2]) + \ldots + (s[n] - a[i_n][n]) = \\ = \sum_{i=1}^{n}\sum_{k=1}^{n} a[i][k] - \sum_{k=1}^{n} a[i_k][k].$$

Двойная сумма в правой части этого равенства — это сумма всех чисел исходного массива $a[n][n]$, и она не зависит от выбора контейнеров i_1, i_2, …, i_n. Число операций s будет наименьшим только, если сумма $\sum_{k=1}^{n} a[i_k][k]$ принимает наибольшее значение. Другими словами, в двумерной таблице-массиве $a[n][n]$ необходимо выбрать n элементов $a[i_1][1]$, $a[i_2][2]$, …, $a[i_n][n]$ — по одному из каждой строки и из каждого столбца — так, чтобы сумма выбранных чисел была *наибольшей*.

Для оценки сложности алгоритма подсчитаем количество вариантов выбора таких n элементов в массиве $a[n][n]$. Это количество совпадает с количеством вариантов расстановки n ладей на шахматной доске $n \times n$, в которых ни одна из них не угрожает другой, то есть равно $n! = 1 \cdot 2 \cdot \ldots \cdot n$.

Подзадача 1. При $n = 2$ количество вариантов в переборном алгоритме равно $2! = 2$. Другими словами, достаточно выбрать наибольшее число из двух сумм $a[1][1] + a[2][2]$ и $a[1][2] + a[2][1]$. Это решение оценивается в 20 баллов.

Подзадача 2. При $n = 3$ количество вариантов в переборном алгоритме равно $3! = 6$ и необходимо выбрать наибольшую из шести сумм, каждая из которых состоит из трёх слагаемых. Это решение оценивается ещё в 20 баллов.

Подзадача 3. При $n = 4$ задача также решается аналогичным перебором.

Подзадача 4. Для реализации переборного алгоритма в общей ситуации сгенерируем все $n!$ перестановок n-элементного множества $\{1, 2, \ldots, n\}$. Затем для каждой перестановки $(i_1, i_2, \ldots, i_n)$ вычисляем сумму вида $\sum_{k=1}^{n} a[i_k][k]$. Теперь осталось выбрать наибольшую среди $n!$ таких сумм.

При $n \leqslant 10$ количество вариантов не превосходит $10! < 4 \cdot 10^6$.

Такое переборное решение оценивается в 100 баллов.

Задача 5. Кенгурёнок и Тигра

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

Основные темы задачи — геометрия, кратчайшие пути между точками.

Прежде всего, отметим, что наименьший по длине маршрут перемещения кенгурёнка из точки A в точку B будет наименьшим и по количеству прыжков.

Подзадача 1. Пусть длина d прыжка кенгурёнка больше расстояния AB. Тогда кенгурёнок сможет попасть из точки A в точку B за два прыжка — сначала он прыгает в точку C, отстоящую от точек A и B на расстоянии d, а затем уже из точки C прыгает в точку B. Для вычисления координат точки C сначала находим середину M отрезка AB, составляем вектор $\overrightarrow{MT}$, перпендикулярный вектору $\overrightarrow{AB}$. Затем в направлении вектора $\overrightarrow{MT}$ от точки M откладываем отрезок MT' (вектор) длины $\sqrt{d^2 - \frac{1}{4}AB^2}$. Осталось к координатам точки M прибавить координаты смещения MT'.

Реализация этой идеи позволяет решить задачу на 25 баллов.

Подзадачи 2 и 3. Пусть точки A и B находятся на горизонтальной или вертикальной прямой. Тогда вычислительные формулы для нахождения промежуточных точек, в которые нужно прыгать кенгурёнку, достаточно простые.

Сначала находим расстояние AB и проверяем, делится ли AB без остатка на длину прыжка d. Если это так, то количество прыжков равно частному от деления AB на d, а координаты промежуточных точек-прыжков получаются из координат точки A смещением по горизонтали (подзадача 2) или вертикали (подзадача 3) на величину $\frac{i}{q} \cdot AB$, где i — номер прыжка, q — частное от деления AB на d.

Если же AB не делится нацело на d, то общее число прыжков равно 2, если $AB < 2 \cdot d$, и равно $q = \lceil \frac{AB}{d} \rceil$, если $AB > 2 \cdot d$. (Здесь $\lceil x \rceil$ — наименьшее целое, которое не меньше чем x.) Для того чтобы попасть из точки A в точку B кенгурёнок сначала делает $q - 2$ прыжков по прямой AB, приближаясь к точке B, а затем, когда оставшееся расстояние до точки B будет уже меньше $2d$, ещё двумя прыжками, как в описано в подзадаче 1, попадает в точку B.

Реализация этих идей позволяет решить задачу на 75 баллов.

Подзадача 4. Идея решения задачи в общей ситуации такая же, как и в подзадачах 2 и 3. Формулы для вычисления координат промежуточных прыжков незначительно усложняются, поскольку приходится учитывать угол наклона прямой AB.

Реализация этих идей в общем случае позволяет решить задачу на 100 баллов.

Задача 6. Простые суммы

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

Основные темы задачи — паросочетания, венгерский алгоритм Куна, задача о назначениях.

Пусть n — количество участников соревнований. Перенумеруем участников соревнований числами от 1 до n и составим граф G, в котором вершины соответствуют номерам участников соревнований. Две вершины i и j графа G соединены ребром, если сумма исходных номеров $a[i]$ и $a[j]$ является простым числом.

Каждое разбиение игроков на пары соответствует некоторо-

му выбору рёбер графа G. Поскольку каждый спортсмен может принимать участие не более чем в одной паре, соответствующая ему вершина может входить только в одно ребро графа G. Другими словами, любые два ребра, входящие в требуемое разбиение, не имеют общих вершин. В теории графов такие рёбра называют *попарно несмежными*, а набор из попарно несмежных рёбер — *паросочетанием* (или независимым множеством рёбер графа).

Наибольшее паросочетание (или максимальное по размеру паросочетание) — это такое паросочетание, которое содержит максимальное количество рёбер. Число паросочетания $\nu(G)$ графа G — это число рёбер в наибольшем паросочетании. (У графа может быть несколько наибольших паросочетаний.)

Таким образом, в задаче требуется найти число паросочетания и само наибольшее паросочетание графа G.

Подзадача 1. Ясно, что число паросочетания $\nu(G)$ не превосходит $\frac{n}{2}$. При небольших ограничениях на количество вершин n задачу можно решить несложным перебором.

Подзадача 2. Для нахождения наибольшего паросочетания в произвольном графе можно воспользоваться алгоритмом Эдмондса или алгоритмом поиска максимального потока. Асимптотическая сложность таких решений $O(n^4)$, и они проходят тесты для всех $n \leqslant 100$.

Подзадачи 3 и 4. Разобьём множество всех вершин графа G на два подмножества (доли): в одну часть включим все *чётные* числа, во вторую — все *нечётные* числа. Очевидно, сумма любых двух чётных или двух нечётных чисел является чётной, то есть не может быть простым числом. Отсюда следует, что каждое ребро графа G соединяет вершины только из разных долей. Такой граф называется *двудольным*. В этом случае можно воспользоваться более эффективным алгоритмом Куна поиска наибольшего паросочетания, его сложность — $O(n^3)$.

В зависимости от технической реализации этого алгоритма получим решения, которые проходят тесты для всех $n \leqslant 1500$ (подзадача 3) или $n \leqslant 2500$ (подзадача 4). Для решения подзадачи 4 достаточно указать явное описание долей графа G сразу на этапе считывания данных.

Региональный этап, 2019-2020

Задача 1. Разность квадратов

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

Во-первых заметим, что числа 1, 2 и 4 невозможно представить в виде разности двух квадратов.

Рассмотрим решения для первых двух подзадач, время их работы $O(n)$.

Переберём все возможные числа x из интервала $(1; 2^{10})$. Зная уменьшаемое и разность, найдем вычитаемое. Если число является квадратом, то ответ найден. Если ни одно число из вычитаемых не является квадратом, то ответа нет.

Для решения подзадачи 3 заметим, что $x^2 - y^2 = (x-y)(x+y)$. Переберём делители n, и для каждого делителя $p \leqslant \sqrt{n}$ проверим, есть ли решение в целых числах у системы уравнений

$$\begin{cases} x - y = p, \\ x + y = n/p. \end{cases}$$

Если такое решение есть, ответ получен.

Интересно, что это решение легко модифицируется до $O(1)$. Если x и y одинаковой чётности, то их разность квадратов делится на 4, иначе – не делится даже на 2. Поэтому для чётных n, которые не делятся на 4, ответ «`No`».

При *нечётном* n исследуем такую систему уравнений:

$$\begin{cases} x - y = 1, \\ x + y = n \end{cases} \iff \begin{cases} x = \frac{1}{2}(n+1), \\ y = \frac{1}{2}(n-1). \end{cases}$$

Другими словами, для любого нечётного $n \geqslant 3$ решение всегда существует.

Если n делится на 4, решим систему:

$$\begin{cases} x - y = 2, \\ x + y = n/2 \end{cases} \iff \begin{cases} x = \frac{1}{4}(n+4), \\ y = \frac{1}{4}(n-4), \end{cases}$$

то есть решение всегда существует при любом натуральном $n \geqslant 4$, кратном 4.

Заметим, что решение с перебором делителей автоматически найдёт указанные решения, если n не даёт остаток 2 при делении на 4. Таким образом, для его доработки достаточно рассмотреть указанный случай.

Наконец, не забудем, что 0 можно получить, например, с помощью пары $x = 1$, $y = 1$.

Задача 2. Превышение скорости

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

Предположим, максимальное превышение скорости автомобиля на дороге не превышает d. Это значит, что на i-м участке автомобиль ехал со скоростью не выше $v_i + d$ и проехал его за время не меньше $\frac{l_i}{v_i+d}$. Общее время, за которое автомобиль проедет дорогу, не меньше

$$\sum_i \frac{l_i}{v_i + d}.$$

Таким образом, превышение не больше чем на d возможно, если

$$s + \sum_i \frac{l_i}{v_i + d} \geqslant t.$$

Решение за $O(nmq)$. Для каждой верхней границы отрезков штрафов проверим, возможно ли, чтобы превышение скорости автомобиля было не более чем d. Заметим, что если для некоторого i невозможно, чтобы превышение было не больше a_{i-1}, то гарантировано можно назначить автомобилю штраф f_i. Из всех таких значений надо выбрать максимальное возможное.

Заметим, что проверку можно делать с использованием вещественной арифметики, не опасаясь проблем с точностью, благодаря ограничению в условии, что небольшое изменение времени въезда или выезда во входных данных не может изменить штраф.

Это решение проходит все подзадачи, кроме подзадачи 7.

Чтобы решить подзадачу 7, заметим, что свойство «можно проехать с превышением не больше d» является *монотонным*: если можно проехать с превышением не больше d, то можно проехать с превышением не больше d' для всех $d' \geqslant d$. Поэтому для определения максимального возможного штрафа можно применить двоичный поиск. Полученное решение работает за время $O(nq \log m)$.

Отметим также частичные решения для некоторых подзадач.

Для решения подзадач с $n = 1$ можно воспользоваться тем, что в этом случае легко вычисляется превышение, а именно: $d = \max(0, (t - s)/l_1)$.

В подзадачах с $m = 1$ требуется лишь проверить, есть ли превышение. Для этого можно вычислить, за какое время может проехать дорогу автомобиль, соблюдающий ограничения скорости.

В подзадаче 5 можно применить линейный поиск вместо двоичного, перебрав значения d превышения от 0 до 10.

Задача 3. Борьба с рутиной

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

Для решения первых двух подзадач можно непосредственно реализовать определение из условия. Будем перебирать значения d, и для каждого значения d переберём все отрезки длины d. Для каждого отрезка переберём все числа на нём и посчитаем число различных.

В подзадаче 1 для этого можно использовать массив элементов типа `bool`, отмечая встретившиеся значения. В подзадаче 2 встретившиеся значения можно сложить, например, в `std::set`, либо в массив, отсортировать и найти число различных значений. Наконец, отметим, что поскольку $n \leqslant 50$, можно обойтись даже без сортировки, для каждого элемента проверяя, не встречался ли он раньше.

Решение с `std::set` при аккуратной реализации может также пройти тесты подзадачи 3. Другая возможность — отсортировать

значения на отрезке с помощью `std::sort` и посчитать количество различных значений.

Улучшим это решение.

Зафиксируем длину отрезка d и определим для каждого числа c в массиве, в скольких отрезках оно не встречается. Рассмотрим два соседних вхождения числа; если расстояние между ними $x > d$, то существует ровно $(d - x + 1)$ отрезков, лежащих между ними. Просуммировав эту величину по всем парам соседних вхождений числа, а также между началом массива и первым числом и между последним числом и концом массива, получим количество отрезков, которые не включают в себя ни одно вхождение данного числа.

Обозначим эту величину через $D(d, c)$. Тогда ответ для заданной длины отрезка — это $\sum_c((n - d + 1) - D(d, c))$. Если посчитать эту сумму наивно, получим решение за $O(n^2)$, которое проходит подзадачи 3–5. Заметим, что для решения подзадачи 4 этим методом не требуется перебор значений, которые встречаются в массиве, можно перебирать значения от 1 до 5000.

Для получения полного решения этим методом нужно избавиться от суммирования по c для каждого d. Для этого запишем все значения расстояний между соседними вхождениями c в один массив a для всех c. Тогда

$$\sum_c((n - d + 1) - D(d, c)) = \sum_i \max(a[i] - d + 1, 0).$$

Для подсчёта последней суммы можно использовать префиксные суммы массива a_i и двоичным поиском находить позицию, где $\max(a[i] - d + 1, 0)$ начинает равняться 0 для данного d. Другой способ — использовать факт, что эта позиция только растёт при увеличении d.

Задача 4. Олимпиада для роботов

Автор задачи :	*Дроздова А., Рахматуллин Р.*
Разработчик :	*Жюри ЦПМК.*
Разбор задачи :	*Жюри ЦПМК.*

Отметим три важных свойства операций `and` и `or`: сохранение 0, сохранение 1 и монотонность:

- 0 `and` $0 = 0$ и 0 `or` $0 = 0$, поэтому если все входные переменные равны 0, то значение любой функции, которую можно задать БМЛП, равно 0;
- 1 `and` $1 = 1$ и 1 `or` $1 = 1$, поэтому если все входные переменные равны 1, то значение любой функции, которую можно задать БМЛП, равно 1;
- Если увеличить значение аргумента, то значение `and` и `or` не уменьшается. Значит если входное значение изменяется с 0 на 1, то значение функции, которую вычисляет БМЛП, либо не изменяется, либо изменяется с 0 на 1.

Изначально положим $z_i = 0$. Поскольку $x < 0$ ложно для всех неотрицательных x, все начальные значения val будут равны 0, а значит, в силу сохранения 0 все значения функций равны 0.

Изменим одно из значений $z_i < m$ на $z_i + 1$. Заметим, что в этом столбце ровно одно значение $x_{j,i}$ равно z_i. Только у j-й булевой функции изменятся входные переменные, причем 0 заменится на 1, а значит, в силу монотонности количество программ, возвращающих единицу, либо не изменится, либо увеличится на единицу.

Наконец, если все $z_i = m$, то все входные переменные равны 1 и значение всех функций равно 1.

Сделав эти наблюдения, получаем первое решение задачи. Будем последовательно выбирать любое значение $z_i < m$ и увеличивать его на единицу. После этого будем пересчитывать количество программ, возвращающих единицу, тогда в какой-то момент мы обязательно получим s программ. Такая наивная реализация работает за $O((nm)^2)$ и решает подзадачи 1 и 3.

Заметим, что поскольку входные данные изменяются только для одной функции, то можно пересчитывать значение только для неё. Время работы усовершенствованного решения равно $O(mn^2) = O((nm)n)$, поэтому оно решает подзадачи с небольшим значениями n: 1, 2, 3 и 6.

Поскольку порядок изменений z_i не важен, можно, например, сначала увеличивать z_1, пока оно не станет равно m, потом z_2, и

так далее. Теперь для фиксированного i можно сделать двоичный поиск по количеству изменений. Полученное решение работает за $O((nm)\log(nm))$ и при достаточно оптимальной реализации может пройти тесты для всех подзадач.

Рассмотрим теперь решение без двоичного поиска. Для каждой инструкции нас интересует первый момент, когда она станет равна единице. Заметим, что результат каждой инструкция, кроме последней, в каждой программе является входом ровно одной другой инструкции. Поэтому, когда мы увеличиваем z_i, мы изменяем ровно одно выражение $(x_{j,i} < z_i)$ с нуля на единицу, которое в свою очередь может изменить выход той инструкции, в которой это выражение было входом, и так далее. Следуя этой логике, будем рекурсивно подниматься по дереву разбора программы и изменять выходы инструкций с нуля на единицу, пока не встретим единицу или пока не прекратим изменения.

Так как каждая инструкция будет изменена не более одного раза, итоговое время работы составит $O(nm)$. В этом решении используется бесповторность программы: факт, что каждая ячейка массива *val* используется как вход ровно один раз.

Задача 5. Максимальное произведение

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

В первой подзадаче можно просто перебрать все возможные точки разреза, для каждой найти сумму элементов, перемножить и выбрать оптимальное значение. При этом сумму можно каждый раз вычислять заново, получив время работы $O(n^2)$. Благодаря тому, что сумма элементов всего массива не превышает 10^9, все действия можно безопасно выполнить в 64-битном типе данных.

Для решения четвертой подзадачи можно заметить, что вместо суммирования элементов заново, каждый раз при перемещении границы вправо на один элемент можно пересчитывать сумму за $O(1)$. Следовательно решение работает за $O(n)$.

Для решения остальных подзадач метод «просуммировать

элементы и перемножить» не подходит — произведение получается слишком большое и не влезает в 64-битный тип данных. Заметим, что решения на языке `Python` или в случае доступности 128-битного типа данных, в принципе, лишены этого недостатка. Но есть решение, которое не выполняет операций умножения.

Решим сначала подзадачи, где все значения равны. Пусть массив поделен на части размера k и l. Заметим, что произведение сумм равно $(k \cdot a) \cdot (l \cdot a)$, где a — значение элементов массива, а $k + l = n$. Так как $(ka)(la) = (kl)a^2 = k(n-k)a^2$, достаточно максимизировать величину $k(n-k)$. Эта величина максимальна, когда k примерно равно $n/2$, а именно равно $n/2$ при чётном n и $(n \pm 1)/2$ при нечётном n. Таким образом, получаем решение за $O(1)$ — нужно всего лишь вывести $\lfloor n/2 \rfloor$. Это решение подходит для второй и пятой подзадачи, также можно не находить математически максимум $k(n-k)$, а перебрать значения k. Во второй подзадаче для вычисления $k(n-k)$ при этом достаточно 32-битного типа данных.

Наконец, для получения полного решения снова используем идею о том, что значение $k(n-k)$ максимально при k в районе $n/2$. Пусть s — сумма всех элементов массива, а t_k — сумма первых k элементов. Тогда произведение $t_k(s - t_k)$ максимально, когда t_k максимально близко к $s/2$. Далее, можно перебирать k и вычислять t_k за $O(n)$ в первых трёх подзадачах, либо вычислять t_k за $O(1)$ с помощью префиксных сумм, или пересчитывать за $O(1)$ при увеличении k на 1, получая полное решение.

Задача 6. Планировка участка

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

Заметим, что если $a > n$, то $ab - cd > ab - ad = a(b-d) \geqslant a > n$ и никакие значения b, c и d не подходят. Следовательно $a \leqslant n$. Аналогично доказывается, что $b \leqslant n$.

Переберём a и b от 2 до n, а также c от 1 до $a-1$ и d от 1 до $b-1$. Проверив для каждого варианта, что $ab - cd = n$,

получим решение первой подзадачи за $O(n^4)$, которое несложно модифицируется проверкой неравенств $a \neq x$ и $b \neq x$, тем самым решая подзадачу 2.

Пусть мы зафиксировали a, b и c. Тогда заметим, что d определяется единственным образом из уравнения $ab - cd = n$: если $ab - n$ кратно c, то $d = (ab - n)/c$, иначе никакие значения d не подходят. Следует также проверить, что $1 \leqslant d < b$. Таким образом получается решение за $O(n^3)$, оно подходит для подзадачи 3, а добавляя проверку неравенств $a \neq x$ и $b \neq x$, также и для подзадачи 4.

Для решения подзадачи 5 можно применить забавный трюк. Давайте подсчитаем предыдущим алгоритмом за $O(n^3)$ решения для всех возможных значений n. Решение за $O(n^4)$ работает очень долго для n порядка 3000, но у нас почти все время тура. Все полученные значения поместим в константный массив. Такой метод известен как *предподсчёт*. К сожалению, сделать предподсчёт для шестой подзадачи не получается — это и слишком долго, и занимает слишком много места, такой большой исходный файл не удастся отправить на проверку.

Наконец, приведём решение за $O(n^2 \log n)$, которое решает задачу полностью.

Вместо c будем перебирать значения $a - c$. Заметим, что $ab - cd = ab - bc + bc - cd = (a - c)b + c(b - d)$. Поскольку $c > 0$ и $b > d$, второе слагаемое положительно, и следовательно, $(a - c)b < n$. Таким образом, достаточно перебирать только $O(n/b)$ значений. Значит, для каждого a необходимо проверить

$$n + n/2 + n/3 + \ldots + n/n = n(1 + 1/2 + 1/3 + \ldots + 1/n)$$

пар значений. Сумма $1 + 1/2 + 1/3 + \ldots + 1/n$ хорошо известна — это частичная сумма гармонического ряда, она равна $O(\log n)$. Следовательно, для фиксированного a суммарно перебирается $O(n \log n)$ значений, а общее время работы алгоритма равно $O(n^2 \log n)$. Финальное замечание: поскольку a и b перебираются явно, нет никакой сложности в том, чтобы проверить $a \neq x$ и $b \neq x$.

Задача 7. Банкомат

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

Рассмотрим сначала решение первых четырёх подзадач, в которых $q \leqslant 5$. В этом случае каждый запрос можно обрабатывать независимо от других, поэтому будем решать задачу для одного значения b.

В первой подзадаче с ограничениями $b \leqslant 500$ и $n \leqslant 500$ можно применить динамическое программирование. Обозначим как $dp[i]$ количество купюр, которое будет выдано, если запросить сумму i. Тогда $dp[0] = 0$, и если $m(i)$ — максимальный номинал купюры, которая не превосходит i, то $dp[i] = dp[i - m(i)] + 1$. Ответом будет максимальное значение среди $dp[1], \ldots, dp[b]$. Решение работает за время $O(nb)$.

Чтобы распространить описанное решение на третью подзадачу, избавимся от множителя n в оценке времени работы. Для этого заметим, что $m(i)$ только возрастает при увеличении i, поэтому в процессе вычисления $dp[i]$ можно поддерживать текущее значение $m(i) = a_j$ и при переходе к следующему значению i проверять, нельзя ли перейти к следующему номиналу a_{j+1}. Время работы будет $O(b + n)$, и третья задача будет решена.

Во второй подзадаче номиналы купюр являются последовательными степенями двойки. Заметим, для выдачи суммы c банкомат выдаст купюры с номиналами, соответствующими позициям единиц в двоичной записи числа c. Значит, задача свелась к поиску числа, не превосходящего b, содержащего максимальное количество единиц в двоичной записи. Пусть двоичная запись числа b содержит k разрядов. Тогда число единиц в ответе не превышает k, и число $2^{k-1} - 1$ заведомо меньше b и содержит ровно $k - 1$ двоичную единицу. таким образом, ответом будет либо b, либо $2^{k-1} - 1$; получившееся решение работает за $O(\log b)$.

Решение четвертой подзадачи приближает нас к полному решению: ограничения на n, a_i и b_i в ней максимальны, и только $q \leqslant 5$. Научимся решать задачу для одного запроса b за $O(n)$. Правильное решение основывается на жадном алгоритме.

Сделаем инверсию задачи. Пусть mn_x — минимальное число c такое, что банкомат выдаст x купюр для суммы c.

Будем по-прежнему обозначать через $m(v)$ максимальный номинал купюры, не превышающий v. Докажем, что

$$mn_x = mn_{x-1} + m(mn_x).$$

С одной стороны, $mn_{x-1} \leqslant mn_x - m(mn_x)$, поскольку из набора купюр, дающего сумму mn_x, всегда можно убрать самую большую купюру (равную $m(mn_x)$), при этом получим корректный набор из $x-1$ купюр, дающий в сумме $mn_x - m(mn_x)$.

С другой стороны, предположим, что мы зафиксировали x и $mn_{x-1} < mn_x - m(mn_x)$. Можно заметить, что в этом случае величина $mn_{x-1}+m(mn_x)$ меньше, чем mn_x, и банкомат вернёт при запросе такой величины ровно x купюр — одну купюру $m(mn_x)$ (она максимальная, не превышающая этой суммы) и еще $x-1$ купюр, на которые раскладывается сумма mn_{x-1}. Противоречие, исходное утверждение доказано.

Используя этот факт, можно посчитать все значения mn_x за время $O(n)$. Заметим, что для двух последовательных чисел a_i, a_{i+1} и для всех значений x таких, что $m(mn_x) = a_i$ (то есть если банкомат выдаёт x купюр, самая большая купюра равна a_i), значения mn_x образуют арифметическую прогрессию с разностью a_i и значениями в промежутке $[a_i; a_{i+1})$. Посчитав такую арифметическую прогрессию для промежутка $[a_i; a_{i+1})$, можно перейти к промежутку $[a_{i+1}; a_{i+2})$: если последнее значение в прогрессии равно y ($y < a_{i+1}$), то следующая прогрессия начнётся с y и будет состоять из чисел $y + k \cdot a_{i+1}$, меньших a_{i+2}. Чтобы найти все эти числа и их количество, можно просто поделить длины соответствующих отрезков нацело на a_{i+1}. Таким образом, последнее число mn_x, не превышающее b, и соответствующее значение x являются ответом на задачу.

Задача 8. Плакаты

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

В первой подзадаче можно просто перебрать, кто из друзей поднимет плакаты, для каждого варианта проверить, что нет

трёх поднятых подряд плакатов, и посчитать суммарную красочность поднятых плакатов. Время работы решения $O(2^n \cdot n)$.

Во второй подзадаче можно реализовать тот же алгоритм, но повторить его q раз, после каждого изменения. Время работы $O(2^n \cdot nq)$.

Третья и четвертая подзадача решаются с помощью динамического программирования. Заметим, что друзья 1, 2, 3 и 4 не могут одновременно поднять плакат. Проверим, кто из них не поднял плакат, и переставим его и друзей, которые идут до него, в конец нумерации. Теперь стоящий последним друг не поднял плакат, поэтому из задачи на окружности мы получили задачу на прямой. Используем для её решения динамическое программирование: обозначим через $dp[i][j]$ максимальную суммарную красочность плакатов, поднятых первыми i друзьями, причём последние j из них подняли плакат ($0 \leqslant j \leqslant 3$). Тогда

$$dp[i][0] = \max(dp[i-1][0],\ dp[i-1][1],\ dp[i-1][2],\ dp[i-1][3]),$$

а для $j > 0$ выполнено $dp[i][j] = dp[i-1][j-1] + a_i$.

В решении третьей подзадачи запустим этот алгоритм $q+1$ раз, для начальной позиции и после каждого изменения, получив решение за $O(nq)$. В четвертой подзадаче один запуск этого алгоритма работает за $O(n)$.

Наконец, для полного решения задачи необходимо научиться изменять величину красочность плакатов и пересчитывать значения $dp[i][j]$. Потребуется чуть более сложная конструкция: построим дерево отрезков на индексах от 1 до n. В узле, соответствующем отрезку от l до r, будем хранить матрицу 4×4: $best[a][b]$ равно максимальной суммарной красочности плакатов, поднятых друзьями, с номерами от l до r, в начале a стоящих подряд друзей подняли плакат, а в конце — b подряд стоящих друзей. Тогда объединение двух соседних отрезков происходит по алгоритму, аналогичному пересчёту значений динамического программирования в одной из предыдущих задач. Изменяя значение красочности i-го плаката, нам необходимо пересчитать $O(\log n)$ значений в узлах на пути к листу, соответствующему отрезку от i до i.

Время работы получившегося решения — $O(n + q \log n)$. Наконец, отметим, что, несмотря на небольшие по меркам получившейся асимптотики ограничения, пересчёт значения в узле дере-

ва отрезков получился довольно трудоёмким, поэтому константа, «спрятанная» в O, довольно велика, и для того, чтобы уложиться в ограничение по времени, необходима достаточно аккуратная реализация.

Задача 9. Шпион, выйди вон!

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

Основные темы задачи — битовые операции, поиск элемента в массиве.

Решение $O(n)$ для $k = 2, 3, 4$. [Битовые операции.]

Подзадача 1. При $k = 2$ ответом в задаче будет результат побитовой операции `XOR` всех чисел. В самом деле, если какой-то бит в искомом числе равен нулю (или единице), то во всей последовательности он будет равен 1 в чётном числе элементов, и его значение в `XOR` также будет равно нулю (или единице). Проще говоря, одинаковые элементы при суммировании `XOR` взаимо уничтожаются. Такое решение оценивается в 20 *баллов*.

Подзадачи 2 и 3. При $k = 3$ и $k = 4$ можно посчитать сумму битов в каждом двоичном разряде, и если она делилась на 3 (в случае $k = 3$) или на 4 (в случае $k = 4$), то в искомом числе соответствующий бит равнялся нулю. Иначе, этот бит в искомом числе равен единице. Такое решение оценивается в 60 *баллов*.

Решение $O(n \log n)$. [Сортировка.] Отсортируем исходный массив чисел и будем сравнивать соседние числа массива. В случае их совпадения будем увеличивать значение счётчика на единицу, подсчитывая, таким образом, количество вхождений в массив данного числа. Если значение счётчика оказывается меньше k, номер шпиона найден. В противном случае, переходим к просмотру следующего числа. Поскольку сортировка требует $n \log n$ операций, и затем еще n операций для просмотра элементов отсортированного массива, в итоге получаем алгоритм за $O(n \log n)$ операций. Это решение проходит все тесты (100 *баллов*).

Решение $O(n)$. [Побитовые суммы.] Как и при решении подзадач 2 и 3, посчитаем сумму битов в *каждом* разряде для всех чисел исходного массива. Если эта сумма делится на k, то в искомом числе соответствующий бит равен 0. Иначе, этот бит в искомом числе равен единице. Такое решение оценивается в 100 *баллов*.

Задача 10. Сломанный калькулятор

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

Подзадача 1. [Перебор.] Для чисел $n \leqslant 100$ задачу можно решить несложным перебором операций умножения и извлечения корня. Сначала раскладываем число n на простые множители. Заметим, что указанные в условии операции не позволяют избавиться от простых множителей, которые входят в разложение числа n. Поэтому *наименьшее* число, которое можно получить из n, должно содержать те же простые делители, что и в числе n, но с меньшими показателями, желательно со степенями 1.

Итак, рассматриваем все простые множители числа n. Некоторые из них будут входить в разложение числа n с нечётными степенями, и значит, операция извлечения корня неприменима к n. Будем умножать n на эти же *простые* числа, добиваясь, чтобы у полученного произведения эти делители имели чётную степень. После этого можно применить операцию извлечения корня, а затем к полученному результату снова применить аналогичные рассуждения. Таким образом, комбинируя операции умножения на простые делители n и операции извлечения корня, можем придти к наименьшему числу за минимальное число операций.

Подзадачи 2 и 3. [Факторизация.] Разложим исходное число $n > 1$ на простые множители, представив его в виде $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdot \ldots \cdot p_m^{\alpha_m}$, где $p_1, p_2, \ldots, p_m$ — различные простые делители n. Как уже отмечалось, при любых допустимых операциях с числом n показатели $\alpha_1, \alpha_2, \ldots, \alpha_m$ будут всегда не меньше 1, поэтому наименьшее число, которое можно получить из n, будет не меньше, чем $n = p_1 p_2 \cdot \ldots \cdot p_m$.

Рассмотрим сначала случай $m = 1$, то есть $n = p^\alpha$. Если показатель α является степенью двойки, $\alpha = 2^s$, то с помощью s операций извлечения корня получим число p:

$$p^\alpha \to p^{\alpha/2} \to p^{\alpha/2^2} \to \ldots \to p^{\alpha/2^s} = p.$$

Если же α не является степенью двойки и $2^{s-1} < \alpha < 2^s$, то с помощью одной операции умножения на число $p^{2^s-\alpha}$ увеличиваем этот показатель до значения 2^s, а затем с помощью s операций извлечения корня получим число p. Всего потребуется $1 + s$ операций:

$$p^\alpha \to p^\alpha \times p^{2^s-\alpha} = p^{2^s} \to \sqrt{p^{2^s}} =$$
$$= p^{2^{s-1}} \to \sqrt{p^{2^{s-1}}} = p^{2^{s-2}} \to \ldots \to p^{2^0} = p.$$

Пусть теперь $m > 1$, то есть у числа n более одного простого множителя. В этом случае среди всех показателей $\alpha_1, \alpha_2, \ldots, \alpha_m$ выбираем *наибольший*, пусть это будет α_k. Если $2^{s-1} < \alpha_k < 2^s$, то умножаем n на такое число, после которого все показатели α_i будут равны 2^s, а затем с помощью ещё s операций извлечения корня получаем наименьшее число. Как и выше, наименьшее число операций равно $1 + s$.

Если же все показатели α_i совпадают с $\alpha_k = 2^s$, то наименьшее число операций будет s.

Это решение имеет сложность $O(\sqrt{n})$; в зависимости от технической реализации процедуры разложения на простые множители оно получает 60 *баллов* или 100 *баллов*.

Муниципальный этап, 2020-2021

Задача 1. Подарки Деда Мороза

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

Основные темы задачи — сортировка, логика, разбор случаев.

Упорядочим исходные числа по возрастанию, для удобства будем считать, что $a \leqslant b \leqslant c$. Рассмотрим два случая.

1 случай. Пусть $a + b \leqslant c$. Тогда Дед Мороз сможет приготовить $a + b$ подарков. Для этого он сначала приготовит a подарков из a пряников и a мандаринов, затем из b конфет и b мандаринов он может сделать ещё b подарков, при этом оставшиеся $c - a - b$ мандаринов будут неиспользованными. Поскольку первые два множества содержат всего $a + b$ элементов, сделать больше чем $a + b$ подарков не удастся.

2 случай. Пусть $a + b > c$. Тогда Дед Мороз сможет приготовить $(a + b + c)/2$ подарков.

Операция / здесь означает целочисленное деление, то есть ответом будет целое значение без остатка. (Доказать это можно, например, так. Дед Мороз уравнивает количества угощений, приготовив вначале $c - b$ подарков из пряников и мандаринов, после чего остаётся $x = a - (c - b)$ пряников и $c - (c - b) = b$ мандаринов. Затем из конфет и мандаринов он делает еще $(b - x)$ подарков, и значит, останется $b - (b - x) = x$ конфет и столько же мандаринов. Теперь у Деда Мороза x пряников, x конфет и x мандаринов, из которых получаются $3x/2$ подарков. Всего приготовлено $(c-b)+(b-x)+\frac{3x}{2} = \frac{1}{2}(a+b+c)$ подарков. Поскольку все три множества содержат $a + b + c$ элементов, очевидно, сделать больше чем $\frac{1}{2}(a + b + c)$ подарков не удастся.)

Задача 2. Больше-меньше

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

Основные темы задачи — жадный алгоритм, сортировка.

Будем расставлять карточки последовательно слева направо с помощью «жадного» алгоритма.

Если сразу после текущей искомой карточки идет знак `<`, то пишем в искомую карточку самое *маленькое* из оставшихся чисел массива a`[]`; иначе — самое *большое* из оставшихся чисел. При таком «жадном» алгоритме заполнения все неравенства в итоге окажутся верными.

В самом деле, если на текущем шаге написано число b, а до этого было написано число a, то при знаке $a < b$ неравенство верно, так как по алгоритму a меньше всех стоящих правее чисел (в частности, меньше b); и при $a > b$ неравенство верно, так как по построению a больше всех стоящих правее чисел (в частности, больше b).

Из приведённого «жадного» алгоритма следует, что расстановка карточек возможна для *произвольной* строки символов `<` и `>`.

Задача 3. Отмерь и отрежь

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

Основные темы задачи — динамическое программирование, индукция, рекурсия.

Отсортируем массив длин в порядке неубывания; будем считать, что $l[1] \leqslant l[2] \leqslant \ldots \leqslant l[n]$.

Обозначим через $L[k]$ *наименьшую* длину провода в задаче, где нужно получить k кусков длиной $l[1]$, $l[2], \ldots, l[k]$. Сначала убедимся, что $L[1] = m \cdot l[1]$.

В самом деле, если длина провода меньше $m \cdot l[1]$, то после

разрезании его на m равных частей каждая часть будет меньше $l[1]$, и мы не сможем вырезать кусок длиной $l[1]$. Если же $L[1] \geqslant m{\cdot}l[1]$, при любом способе разрезания провода на m частей одна из них будет не меньше, чем $l[1]$ (иначе их суммарная длина меньше $L[1]$), и из этой части всегда можно вырезать кусок длиной $l[1]$.

Несложно сообразить, что $L[2] = \max\{\, m \cdot l[2],\ m \cdot l[1] + l[2] \,\}$.

Действительно, первое из этих двух чисел не меньше второго при условии $(m-1) \cdot l[2] \geqslant m \cdot l[1]$, и в этом случае из провода длиной $L[2] = m{\cdot}l[2]$ можно вырезать куски с длинами $l[1]$ и $l[2]$. В самом деле, после разрезания провода на m частей произвольной длины одна из них будет не меньше, чем $l[2]$. Удалив из неё кусок длиной $l[2]$, мы получим m частей общей длиной $L[2]-l[2] = (m-1){\cdot}l[2]$, и поскольку выполнено условие $(m-1){\cdot}l[2] \geqslant m{\cdot}l[1]$ можно вырезать ещё кусок длиной $l[1]$, как это доказано в рассуждениях с $L[1] = m \cdot l[1]$.

Если же $(m-1) \cdot l[2] \leqslant m \cdot l[1]$, то в этом случае из провода длиной $L[2] = m \cdot l[1] + l[2]$ можно вырезать куски с длинами $l[1]$ и $l[2]$, причём эта длина наименьшая. В самом деле, из условия следует, что $L[2] \geqslant m \cdot l[2]$, и значит, после разрезания провода на m частей произвольной длины одна из них снова будет длиной не меньше $l[2]$. Снова удалив из неё кусок $l[2]$, мы получим m частей общей длиной $L[2]-l[2] = m \cdot l[1]$, и как доказано в рассуждениях для $L[1] = m \cdot l[1]$, из этих частей всегда можно вырезать кусок длиной $l[1]$.

Таким образом, доказано, что

$$L[2] = \max\{\, m \cdot l[2],\ m \cdot l[1] + l[2] \,\}.$$

Осталось заметить, что величина $m \cdot l[1]$ совпадает с $L[1]$; после такой замены в формуле приходим к формулировке утверждения в общем случае.

Лемма. $L[k] = \max\{\, m \cdot l[k],\ L[k-1] + l[k] \,\}$ для всех $k \geqslant 1$.

Доказательство леммы использует уже знакомые идеи, поэтому если это утверждение кажется очевидным, его обоснование можно пропустить.

Вновь сравним два числа из формулировки леммы. Первое число не меньше второго при условии $L[k-1] \leqslant (m-1) \cdot l[k]$. Докажем, что в этом случае из провода длиной $L[k] = m \cdot l[k]$

можно вырезать требуемые куски с длинами $l[1]$, $l[2], \ldots, l[k]$. В самом деле, при любом способе разрезания провода на m частей одна из них будет не меньше, чем $l[k]$, и из этой части всегда можно вырезать кусок длиной $l[k]$. Удалив из неё кусок длиной $l[k]$, мы получим m частей общей длиной $L[k]-l[k] = (m-1)\cdot l[k]$, и поскольку выполнено условие $(m-1)\cdot l[k] \geqslant L[k-1]$, из них можно вырезать ещё куски длиной $l[1], l[2], \ldots, l[k-1]$. (Последнее следует из определения числа $L[k-1]$.)

Таким образом, в первом случае доказано, что из провода длиной $L[k] = m \cdot l[k]$ можно вырезать *все* требуемые куски с длинами $l[1], l[2], \ldots, l[k]$, причем минимальность $L[k]$ фактически уже доказана.

Аналогично разбирается ситуация с неравенством $L[k-1] \geqslant$ $\geqslant (m-1)\cdot l[k]$.

Действительно, в этом случае из провода длиной

$$L[k] = L[k-1] + l[k]$$

можно вырезать кусок $l[k]$ — это следует из легко проверяемого неравенства $L[k] \geqslant m \cdot l[k]$. После удаления куска длиной $l[k]$ получим m частей суммарной длины $L[k] - l[k] = L[k-1]$. По определению числа $L[k-1]$ из них можно вырезать остальные требуемые куски длиной $l[1], l[2], \ldots, l[k-1]$.

Осталось доказать *точность* приведённой оценки длины провода. Другими словами, нужно доказать, что при уменьшении длины $L[k] = L[k-1]+l[k]$ на сколь угодно малое число вырезать требуемые куски не удастся при некотором специально подобранном способе разрезания провода на m частей.

Рассмотрим последовательность чисел $L[1], L[2], \ldots, L[k]$. Будем просматривать её *справа налево*, и пусть $L[s]$ — первое *справа* число из этого набора, где нарушается неравенства вида $L[i-1] \geqslant$ $(m-1)\cdot l[i]$, то есть $L[s] \leqslant (m-1)\cdot l[s+1]$ и для всех чисел $L[i]$ с номерами $i > s$ выполняется $L[i] \geqslant (m-1)\cdot l[i+1]$. Тогда по предположению индукции $L[s+1] = m\cdot l[s+1]$ и $L[i+1] = L[i]+l[i+1]$ для всех $i \geqslant s+1$. Имеем

$$\begin{aligned} L[k] &= L[k-1] + l[k] = L[k-2] + l[k-1] + l[k] = \\ &= \ldots = L[s+1] + l[s+2] + \ldots + l[k] = \\ &= m \cdot l[s+1] + l[s+2] + \ldots + l[k]. \end{aligned}$$

Рассмотрим провод меньшей длины $L'[k] = L[k] - m \cdot \varepsilon$, где ε — сколь угодно малое положительное число. Разрежем его на m частей, где первые $(m-1)$ частей длиной $a = l[s+1] - \varepsilon$, а последняя m-я часть — оставшейся длины $L'[k] - (m-1) \cdot a$. Из первых $(m-1)$ частей невозможно вырезать куски длиной $l[s+1], \ldots, l[k]$, и значит, их придётся вырезать из последней m-й части, а это сделать невозможно, потому что её длина меньше $l[s+1] + \ldots + l[k]$:

$$L'[k] - (m-1) \cdot a =$$
$$= m \cdot l[s+1] + l[s+2] + \ldots + l[k] - (m-1) \cdot l[s+1] - \varepsilon =$$
$$= l[s+1] + l[s+2] + \ldots + l[k] - \varepsilon.$$

Таким образом, при уменьшении длины $L[k] = L[k-1] + l[k]$ можно подобрать такой разрез провода на m частей, при котором не удастся получить куски требуемой длины. Лемма доказана.

Сложность алгоритма — $O(n)$, техническая реализация — простая.

Задача 4. Самое красивое число

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

Основные темы задачи — теория чисел, факторизация, подсчёт делителей числа.

Пусть a — произвольное число из данного массива чисел, его запись в форме красивой суммы имеет вид:

$$a = (m+1) + (m+2) + \ldots + n =$$
$$= \frac{1}{2}n(n+1) - \frac{1}{2}m(m+1) = \frac{1}{2}(n-m)(n+m+1).$$

Сомножители $n-m$ и $n+m+1$ имеют разную чётность, ровно один из них *нечётный*. Значит, красивое разложение числа a обязательно имеет нечётный множитель. Легко доказать обратное утверждение — каждому нечётному множителю соответствует разложение a с нечётным множителем. Итак, *количество*

нечётных делителей у числа a равно количеству красивых разложений числа a. Например, число 15 имеет четыре нечётных делителя 1, 3, 5 и 15, поэтому у него четыре красивых разложения.

Таким образом, алгоритм решения следующий. Каждое числа a исходного массива раскладываем на простые множители с учетом их кратности:

$$a = p_1^{\alpha_1} p_2^{\alpha_2} \cdot \ldots \cdot p_s^{\alpha_s},$$

отбрасываем все двойки, если они входят в разложение a, и для оставшихся нечётных простых делителей числа a вычисляем общее количество нечётных делителей числа a. Например, если все $p_i \neq 2$, то красота числа a будет равна $(\alpha_1+1)(\alpha_2+1)\cdot\ldots\cdot(\alpha_s+1)$.

Осталось среди всех чисел a_i массива найти то, у которого красота принимает наибольшее возможное значение. Если таких чисел несколько, в ответе записываем наименьшее среди них.

Если использовать стандартный алгоритм факторизации, то общая сложность представленного метода — $O(n \cdot \sqrt{\max a_i})$.

Задача 5. Бермудский треугольник

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

Основные темы задачи — геометрия, общие касательные к двум окружностям. Задание сводится к решению следующей математической задачи:

Построить треугольник по известным расстояниям двух точек до сторон этого треугольника. Точки могут располагаться как внутри, так и вне треугольника. Расстояния для точек внутри треугольника считаются положительными, а для точек вне треугольника — отрицательными.

Проведём окружность с центром в точке A радиусом, равным расстоянию до прямой XY, на которой лежит сторона Бермудского треугольника. Тогда прямая XY является *касательной* к

этой окружности. Аналогичным образом, XY будет также касательной к окружности с центром в точке B и радиусом, равным расстоянию до XY. Таким образом, искомая прямая XY является *общей касательной* двух окружностей с центрами A и B.

Алгоритм поиска общих касательных к двум окружностям подробно описан на сайте

`http://e-maxx.ru/algo/circle_tangents`.

В большинстве случаев две окружности имеют четыре общих касательных: две из них будут *внешними*, две — внутренними касательными. Если оба расстояния от точек A и B положительные или отрицательные, необходимо оставить только внешние касательные; если же расстояния от этих точек разных знаков — нужно оставить только внутренние касательные. В последнем случае центры окружностей располагаются по разные стороны от общей касательной, что соответствует исходным условиям задачи.

После того как общие касательные X_iY_i построены, то есть найдены коэффициенты a_i, b_i, c_i в уравнениях прямых

$$a_i x + b_i y + c_i = 0,$$

задающих эти касательные, находим общие касательные Y_iZ_i и X_iZ_i к окружностям с центрами в тех же точках A и B и соответствующими радиусами, равными расстояниям до этих прямых.

Следующим шагом находим точки пересечения всех пар построенных общих касательных. Эта несложная задача сводится к решению соответствующих систем линейных уравнений. Найденные из этих уравнений координаты задают тройки возможных вершин $V[0]$, $V[1]$, $V[2]$ Бермудского треугольника.

Теперь осталось проверить, что найденные вершины действительно удовлетворяют условиям задачи, и в частности, расстояния от точек A и B имеют требуемые знаки. Для этого нужно установить, что A и B находятся «внутри» треугольника, если все эти расстояния положительные, или «вне» треугольника, если все эти расстояния отрицательные.

Региональный этап, 2020-2021

Задача 1. Два станка

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

В первых четырех подзадачах достаточно легко найти конкретную формулу, которая позволяет найти ответ. Обозначим искомое количество деталей как r.

Подзадача 1. Если $a = 0$ и $x = 0$, то первый станок вообще нет смысла использовать. Введем в строй второй и получим ответ:

$$r = b \cdot \max(k - b, 0).$$

Важно не забыть, что время на запуск может оказаться больше k, и тогда количество деталей будет равно 0, для чего мы и берем max в формуле.

Подзадача 2. Если a и b равны нулю, то можно сразу запустить оба станка в работу и получить ответ:

$$r = (x + y) \cdot k.$$

Подзадача 3. Когда $a = b$, выгодно сначала ввести в строй наиболее производительный станок, и сразу после этого начать вводить в строй второй. Ответ тогда будет равен:

$$r = \max(x, y) \cdot \max(k - a, 0) + \min(x, y) \cdot \max(k - 2a, 0).$$

Тут снова надо не забыть о случаях, когда время на введение в строй больше доступного нам.

Подзадача 4. Если станки производят детали с одинаковой скоростью, то требуется ввести в строй сначала тот, на запуск которого уйдет меньше времени, чтобы детали раньше начали производиться. Получаем:

$$r = a \cdot \max(k - \min(a, b), 0) + a \cdot \max(k - a - b, 0).$$

Подзадача 5. Теперь посмотрим как решать общий случай. Переберем два варианта: какой станок будем вводить в строй первым. Если мы сначала введем в строй первый станок, ответ будет:

$$r_1 = x \cdot \max(k - a, 0) + y \cdot \max(k - a - b, 0).$$

Наоборот, если сначала ввести в строй второй, то получим:

$$r_2 = y \cdot \max(k - b, 0) + x \cdot \max(k - a - b, 0).$$

Чтобы получить ответ на задачу, достаточно взять максимум из двух этих значений:

$$r = \max(r_1, r_2).$$

Заметим, что, разумеется, решение пятой подзадачи решает также и первые четыре подзадачи.

Задача 2. Разбиение таблицы

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

Подзадача 1. Будем перебирать линию разделения таблицы. Затем за $n \cdot m$ операций посчитаем сумму в разделенной таблице. Выберем лучший разрез. В сумме решение работает за $n \cdot m \cdot (n + m)$ операций.

Подзадача 2. Заметим, что при перемещении линии разреза на 1 в каждой из половин добавляется или удаляется только $O(n)$ или $O(m)$ клеток. Значит можно перебрать все варианты за $O(n \cdot m)$.

Подзадачи 3 и 4. Зафиксируем какой-нибудь разрез по вертикали перед столбцом k. Теперь посчитаем сумму в полученном прямоугольнике, используя формулу арифметической прогрессии:

$$a_1 + a_2 + a_3 + \ldots + a_x = \frac{(a_1 + a_x)}{2} \cdot x$$

Тогда

$$1+2+\ldots+k+$$
$$+(m+1)+(m+2)+\ldots+(m+k)+\ldots+$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$+(m\cdot(n-1)+1)+(m\cdot(n-1)+2)+\ldots+(m\cdot(n-1)+k)=$$
$$=\frac{1+k}{2}\cdot k+\frac{(m+1)+(m+k)}{2}\cdot k+\ldots+$$
$$+\frac{(m\cdot(n-1)+1)+(m\cdot(n-1)+k)}{2}\cdot k.$$

Полученная сумма также является арифметической прогрессией с шагом $m\cdot k$, значит, она равна:

$$\frac{1}{2}\left(\frac{1+k}{2}\cdot k+\frac{((m\cdot(n-1)+1)+(m\cdot(n-1)+k)}{2}\cdot k\right)\cdot n$$

Сумму в горизонтальном разрезе найдем по более простой формуле:

$$1+2+\ldots+m+$$
$$+(m+1)+(m+2)+\ldots+(m+m)+\ldots+$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$+(m\cdot(k-1)+1)+(m\cdot(k-1)+2)+\ldots+(m\cdot(k-1)+m)=$$
$$=\frac{1+m\cdot k}{2}\cdot mk.$$

Таким образом, за $O(1)$ можно посчитать сумму при фиксированном разрезе, а значит, можно найти лучший разрез за $O(n+m)$.

Подзадача 5. Здесь арифметическая прогрессия одномерная и надо найти такое k, чтобы

$$\frac{1+k}{2}\cdot k\approx\frac{1+m}{4}\cdot m$$

Решение 1. Решая это квадратное уравнение относительно k, найдем лучшее k за $O(1)$. Из-за округления величин возможна погрешность, поэтому нужно перебрать ответ в диапазоне ±1.

Решение 2. Можно найти лучшее k, используя двоичный поиск (сумма первых k значений возрастает при увеличении k).

Подзадача 6. Совместим идеи в подзадачах 4 и 5. Посчитаем сумму при фиксированной границе раздела:

$$\frac{\frac{1+k}{2}\cdot k+\frac{(m\cdot(n-1)+1)+(m\cdot(n-1)+k)}{2}\cdot k}{2}\cdot n\approx\frac{1+n\cdot m}{4}\cdot nm.$$

Решение 1. [за $O(1)$ на запрос.] Упрощая, получим квадратное уравнение на k, и найдем лучшее k с точностью до ±1, так как знак приблизительно равно.

Решение 2. [за $\log_2 \max(n,m)$ на запрос.] Можно найти лучшее k, используя двоичный поиск, так как значение слева возрастает при увеличении k.

Задача 3. Изменённая ДНК

Автор задачи : *Христенко О.*
Разработчик : *Жюри ЦПМК.*
Разбор задачи: *Жюри ЦПМК.*

Общая идея. Для того, чтобы получить минимальную длину, всегда выгодно удалить какой-то элемент, а чтобы получить максимальную длину, нужно добавить какой-то элемент. Докажем это.

Для минимума. Пусть мы заменили какой-то символ. Вместо этого удалим этот символ. Длина каждого из соседних блоков не изменится, либо они сольются в один. В любом случае длина закодированной строки получится меньше или равна, чем при замене. Аналогично, вместо добавления символа всегда выгоднее удалить символ перед ним.

Для максимума. Пусть мы заменили какой-то символ. Тогда вместо этого добавим в эту позицию тот символ, на который мы заменили. Длина каждого из соседних блоков не уменьшится, значит длина закодированной версии полученной строки будет не меньше. Аналогично вместо удаления символа всегда выгоднее добавить символ перед ним.

Перейдем теперь к рассмотрению того, как решить, какую именно операцию сделать.

Подзадачи 1 и 2. Для того, чтобы решить первые две подзадачи, достаточно перебрать место, в которое нужно вставить новый элемент, или удалить существующий. После фиксирования изменения, можно посчитать длину закодированной строки полным проходом по ней. Асимптотика этого решения $O(L^2)$.

Подзадача 3. Для того, чтобы решить третью подзадачу, нужно заметить, что каждый блок одинаковых символов, изменился не сильно. Поэтому после изменения символа достаточно для каждого исходного блока посчитать новую длину этого блока или новых блоков, которые появились после изменения. Асимптотика решения $O(n \cdot L)$.

Подзадача 4. Для того, чтобы решить четвертую подзадачу, нужно заметить, что после изменения меняется только блок, в котором произошло изменение, или соседние с ним блоки. Для всех остальных блоков длина не меняется, и её можно не пересчитывать. Асимптотика решения $O(L)$.

Подзадача 5. Для того, чтобы решить пятую подзадачу, нужно научиться понимать, как правильно удалять и добавлять символы.

Если в строке есть блок длины 1, соседние блоки которого содержат одинаковые символы, то при удалении этого символа эти блоки объединятся в один, и длина закодированной строки уменьшится. Если в строке есть блок длины 1, то при его удалении длина строки уменьшится на 1. Если в строке есть блок длины 2, то при удалении символа из него, в его кодировке больше не нужно будет писать число, и длина строки уменьшится на 1. Если в строке есть блок длины 10^x, то при удалении символа из него длина числа уменьшится на 1. Это все возможные способы уменьшения длины при удалении символа.

При добавлении символа можно поставить его в начало для создания нового блока длины 1, чтобы длина строки увеличилась на 1. Также можно какой-то блок разделить на два, если добавить другой символ внутри блока. Для блока длины до 20, достаточно перебрать, где именно будет новый символ. Если длина блока от

$2 \cdot 10^x$ до 10^{x+1}, то из него можно выделить блок размера 10^x, и увеличить длину строки на $x+3$. Если длина блока от 10^x+10^{x-1} до $2 \cdot 10^x$, то из него можно выделить блок длины 10^x, и увеличить длину строки на $x+2$. Если длина блока от 10^x до 10^x+10^{x-1}, то из него можно выделить блок длины $8 \cdot 10^{x-1}$, и увеличить длину строки на $x+1$. Это все возможные способы увеличить длину строки при добавлении символа.

Для каждого блока проверим все оптимальные способы добавления и удаления символа и выберем оптимальный способ среди всех. Так как для каждого блока оптимальных способов всего $O(1)$, то асимптотика решения $O(n)$.

Задача 4. Антенна

Автор задачи :	*Орешников Д., Короткевич Г.*
Разработчик :	*Жюри ЦПМК.*
Разбор задачи :	*Жюри ЦПМК.*

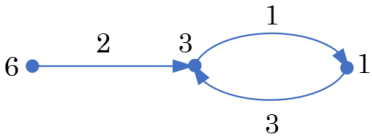
Подзадача 1. Для того, чтобы решить первую подзадачу, можно перебрать, в каком порядке нужно соединить фрагменты. Всего существует $n!$ таких вариантов. Для каждого варианта можно вычислить позиции всех перекладин и проверить, подходит ли такой вариант.

Подзадача 2. Для того, чтобы решить вторую подзадачу, нужно заметить, что если на одном из фрагментов присутствует более одной перекладины, то, если существует способ правильно собрать антенну, расстояние между соседними перекладинами должно быть равно расстоянию между перекладинами на этом фрагменте. Поэтому можно заранее проверить, что расстояния между соседними перекладинами в каждом отдельно взятом фрагменте равны. После чего можно оставить в каждом фрагменте только первую и последнюю перекладины. Затем, как в предыдущей подзадаче, перебрать порядок фрагментов и проверить, подходит ли он.

Общая идея. Как во второй подзадаче, оставим на каждом фрагменте только первую и последнюю перекладины. Пусть на

i-м фрагменте первая перекладина находится на расстоянии l_i от начала, а последняя — на расстоянии r_i от конца. Научимся проверять, существует ли порядок фрагментов, при котором расстояние между соседними перекладинами будет равно d. Пусть такой порядок существует, обозначим его q_1, q_2, $\dots q_n$. Тогда должно выполняться $r_{q_i} + l_{q_{i+1}} = d$ $(1 \leqslant i \leqslant n-1)$, значит $l_{q_{i+1}} = d - r_{q_i}$. Рассмотрим граф, вершинами которого являются все целые числа. Проведем ориентированное ребро из l_i в $d - r_i$ для всех i. Заметим, что искомый порядок существует тогда и только тогда, когда в построенном графе существует эйлеров путь. Поиск эйлерова пути (и проверка существования) является стандартным алгоритмом.

Например, в первом тесте из примеров, $l_1 = 3$, $r_1 = 4$, $l_2 = 6$, $r_2 = 2$, $l_3 = 1$, $r_3 = 2$. В ответе на тест, $d = 5$. Построим граф по этим данным:



Рядом с вершинами написаны соответствующие им числа, а рядом с ребрами — номера соответствующих им фрагментов. Видно, что последовательность ребер 2, 1, 3 является эйлеровым путем.

Для решения оставшихся подзадач, мы будем находить некоторое множество значений d, каждое из которых будем проверять за линейное время алгоритмом из предыдущего абзаца.

Подзадача 4. Чтобы решить четвертую подзадачу, нужно заметить, что по принципу Дирихле всегда будет существовать хотя бы один фрагмент, содержащий хотя бы две перекладины. Поэтому, если искомый порядок существует, расстояние d между соседними перекладинами должно быть равно расстоянию между соседними перекладинами на этом фрагменте. Воспользуемся алгоритмом и проверим, подходит ли данное d.

Подзадача 3. Чтобы решить третью подзадачу, можно перебрать два варианта:

- Если фрагмент номер 1 не будет стоять самым последним,

то можно перебрать номер фрагмента i, который будет стоять сразу после фрагмента 1, и проверить $d = r_1 + l_i$.

- Если фрагмент номер 1 будет стоять самым последним, то фрагмент номер 2 не будет стоять последним. Можно перебрать номер фрагмента i, который будет стоять после фрагмента 2, и проверить $d = r_2 + l_i$.

Таким образом, мы проверим $O(n)$ вариантов, каждый за время $O(n)$. Итого, решение работает за $O(n^2)$.

Подзадача 5. Чтобы решить пятую подзадачу, можно проверить все значения d от 0 до 200. Дополнительно ускорить решение можно, проверяя только значения d от 0 до $\lfloor \frac{100n}{n-1} \rfloor$.

Подзадача 6. Наконец, для полного решения задачи нужно было научиться проверять только $O(1)$ различных вариантов значений d. Рассмотрим пары $(l_{q_i}, r_{q_{i+1}})$ и отсортируем их в порядке возрастания l. Заметим, что так как суммы значений в каждой паре равны, значения r будут убывать. Среди всех значений l в рассмотренных парах не присутствует ровно одно, соответствующее первому фрагменту. Аналогично, среди всех значений r отсутствует значение, соответствующее последнему фрагменту. Таким образом, можно отсортировать все значения l в порядке возрастания, все значения r в порядке убывания и проверить 4 варианта значения d: сумма одного из двух минимальных значений l и одного из двух максимальных значений r.

Задача 5. Календарь на Альфе Центавра

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

Основная идея. Пусть мы нашли x — номер дня от начала летосчисления до заданного во вводе дня. С первого дня первого месяца первого года (который, как мы знаем, имел обозначение «a», прошло $x - 1$ дней. Каждые w дней повторялся день с обозначением «a», поэтому на самом деле нас интересует величина $y = (x - 1) \bmod w$.

Заметим, что если $y = 0$, то день имеет обозначение «a», если $y = 1$, то «b», и так далее. Чтобы получить по сдвигу относительно «a» букву в английском алфавите, можно воспользоваться возможностями языка программирования. В C++ это сделать проще всего:

cout << 'a' + y << **endl**;

В других языках могут потребоваться функции приведения типа. Например, в Паскале:

writeln(chr(ord('a') + y));

В языке Python:

print(**chr**(**ord**('a') + y));

Осталось разобраться, как найти x.

Подзадача 1. В первой подзадаче год состоит из одного дня. Поэтому $i = 1$ и $j = 1$, и значит, $x = k$.

Подзадача 2. Во второй подзадаче $m = 1$, то есть месяц всего один. $x = (k-1) \cdot d + i$. Заметим, что в этой подзадаче значение k небольшое, поэтому достаточно 32-битного типа данных.

Подзадача 3. В третьей подзадаче речь все время идет о первом дне первого месяца, $x = (k-1) \cdot d \cdot m + 1$.

Подзадача 4. В четвертой подзадаче $k = 1$. Год учитывать не надо, надо учесть только день и месяц, $x = (j-1) \cdot d + i$.

Подзадачи 5 и 6. Наконец получим общую формулу: $x = (k-1) \cdot m \cdot d + (j-1) \cdot d + i$. Отличие пятой подзадачи в том, что в ней значение k невелико, и поэтому можно использовать 32-битный тип данных.

В заключение отметим, что с помощью различных способов итераций перебирать дни могли проходить некоторые подзадачи, например подзадачи 1, 2, 3, 5.

Задача 6. Числа

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

Подзадача 1. Заметим, что при $k = 0$ число должно состоять из всех одинаковых цифр. В этой подзадаче можно просто увеличивать x, пока все цифры числа не будут одинаковы. Число 111 111 обладает этим свойством и не меньше всех возможных чисел во вводе, значит будет сделано не больше чем такое количество шагов.

Подзадача 2. В этой подзадаче увеличение шагами по одному уже не приводит к успеху. Для получения баллов за эту подзадачу нужно разобраться, как устроено минимальное число, большее заданного, состоящее из одинаковых цифр.

Первую цифру числа нельзя уменьшить, посмотрим, можно ли её оставить такой же. Пойдем по числу от старших цифр к младшим, пока они равны первой цифре числа f. Если в какой-то момент очередная цифра цифра отличается от первой, посмотрим меньше она или больше. Если она оказывается меньше первой цифры числа, то можно просто заменить все цифры на f, получив большее число из всех одинаковых цифр. Если же она оказывается больше, то придется увеличить первую цифру. Это можно сделать, если она не равна 9. В этом случае заполним число цифрами $f + 1$. Иначе надо взять число из всех единиц, имеющее на одну цифру в своей десятичной записи больше.

Подзадача 3. В этой задаче снова маленькое значение x, и можно увеличивать его, пока все его цифры, кроме, может быть, одной не станут равны. На этот раз верхним порогом выступает число 100 000. Единственная трудность этой подзадачи — техническая, проверить, что все цифры, кроме не более чем k, равны.

Подзадача 4. На самом деле полное решение задачи даже проще разбора частных случаев предыдущих подзадач. Заметим, что искомое число имеет не более 18 десятичных цифр. Значит всего подходящих чисел 18·9 для $k = 0$ и не больше 18·10·18·10 для $k = 1$ (во втором случае мы перебираем длину, основную цифру, позицию отличающейся и саму отличающуюся). Переберем все потенциальные ответы и из тех, которые не меньше x, выберем минимальный.

Код на C++, который строит число из цифр d длины len, на позиции pos ставится цифра $d2$:

```
long long num(int len, int d, int pos, int d2) {
    long long res = 0;
    for (int i = 0; i < len; i++) {
        if (i == pos) {
            res = res * 10 + d2;
        } else {
            res = res * 10 + d;
        }
    }
    return res;
}
```

Следующий код на C++ перебирает все подходящие числа для $k = 1$:

```
for (int i = 1; i <= 18; i++) {
    for (int d = 0; d < 10; d++) {
        for (int p = 0; p < i; p++) {
            for (int d2 = 0; d2 < 10; d2++) {
                long long y = num(i, d, p, d2);
                if (y >= x && (ans == -1 || y < ans)) {
                    ans = y;
                }
            }
        }
    }
}
```

Задача 7. Хорошие раскраски

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

Эта задача — довольно нетрадиционная для олимпиад по информатике. Вместо конкретного конструктивного или алгоритмического решения участникам предлагалось поэкспериментировать с эвристиками и перебором.

Перед разбором основных идей отметим, что конструктивного решения в этой задаче, скорее всего, нет. На это наводит следу-

ющая мысль: конструктивный паттерн, который решал бы задачу для приведенных ограничений, мог бы быть распространен на бесконечное поле. Можно доказать (это интересное олимпиадное упражнение по математике), что раскрасить таким образом в 2 или 3 цвета бесконечное поле невозможно.

Отметим, что в задаче очень маленькие ограничения и потестовая оценка. В принципе, можно локально запустить решение для всех возможных входов (их 200) и отправить в систему результаты предподсчета. Практически любое продвижение для больших полей вознаграждается баллами.

Перейдем теперь к рассмотрению основных идей переборных решений и решений с помощью локальных оптимизаций.

Перебор. Самое простое решение — это перебор за $O(c^{n \cdot m})$, можно перебирать все раскраски простой рекурсией: перебирать цвета клеток в порядке сортировки по строкам и при равенстве по столбцам дополнительно проверять, подходит ли итоговое поле. Чтобы оптимизировать это решение, в процессе перебора подходящих раскрасок, можно проверять, что среди клеток, которым уже проставлены значения, нет плохих четвёрок. Это решение уже может пройти все тесты, где $c = 2$, и значительное число тестов, где $c = 3$. Для дальнейшего улучшения решения можно перебирать цвета для клеток в случайном порядке.

Дополнительная оптимизация, которая сильно ускоряет перебор, — добавление «*мемоизации*». А именно, будем запоминать некоторую информацию о текущей раскраске, которая точно не приведёт к ответу (поскольку такая же комбинация перебиралась ранее, и ответ найден не был). Например, можно хранить множество троек (c, y_1, y_2), для которых среди покрашенных клеток найдется такое x, что $a_{x,y_1} = a_{x,y_2} = c$, и проверять, что такое множество не встречалось, когда мы начинаем красить первую клетку в определенной строке.

Предыдущее решение работает заметно лучше при $m < n$, в противном случае можно поменять местами n и m и транспонировать поле для ответа. С этими оптимизациями можно пройти почти все тесты.

Локальные оптимизации. Другим возможным решением является метод локальных оптимизаций.

Вначале присвоим каждой клетке случайный цвет $1 \dots c$. Затем будем выбирать случайную клетку и менять её цвет до тех пор, пока уменьшается значение некоторой функции, которую назовём *стоимостью* поля.

Стоимостью поля может быть почти любая функция, которая равна нулю для «хороших» полей. Например, в качестве таковой можно взять количество четвёрок x_1, x_2, y_1, y_2, для которых выполняются неравенства $1 \leqslant x_1 < x_2 \leqslant n$, $1 \leqslant y_1 < y_2 \leqslant m$, и при этом клетки (x_1, y_1), (x_2, y_1), (x_1, y_2) и (x_2, y_2) покрашены в одинаковый цвет. Когда дальнейшее изменение цвета не приводит к уменьшению ответа, можно завершить процесс.

Для большинства случайных раскрасок стоимость в конечном итоге будет равна маленькому числу. А если с исходной раскраской «*повезёт*», то стоимость будет равна нулю, и поле сойдется к «хорошему» полю. Отсюда получается следующее решение: будем случайно генерировать исходные поля и запускать описанный выше процесс до тех пор, пока функция стоимости уменьшается, проверяя при этом, что в итоге получается значение 0. Если это так, можно вывести ответ, иначе генерируем исходное поле заново.

Это решение способно пройти все тесты с ограничениями этой задачи меньше чем за секунду.

Задача 8. A + B

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

Подзадача 1. Первая подзадача решается перебором $n!$ перестановок.

Подзадача 2. Каждый столбец можно охарактеризовать тремя параметрами (обозначим цифры в нём через a_i, b_i, c_i):

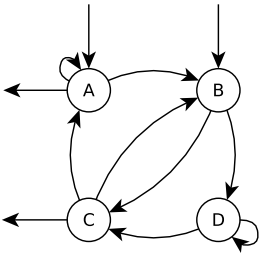
1. Есть ли в нём цифра ноль (это влияет на то, может ли он быть первым)?
2. Требуется ли ему перенос из столбца справа, в зависимости

от того, верно $(a_i + b_i)$ `mod` $10 = c_i$ или $(a_i + b_i + 1)$ `mod` $10 = c_i$? Если ни одно из этих равенств не выполнено, то ответ 0. Обозначим этот параметр $needCarry_i \in \{0, 1\}$.

3. Создаёт ли этот столбец перенос через разряд? Это так, когда $a_i + b_i + needCarry_i \geqslant 10$. Обозначим этот столбец $makeCarry_i \in \{0, 1\}$.

Будем перебирать перестановки столбцов, двигаясь слева направо. Первым может оказаться столбец без нулей и не создающий переноса. Если после столбца i идёт столбец j, то

$$needCarry_i = makeCarry_j.$$



Для последнего столбца $needCarry_i = 0$. Получается граф, в котором нужно найти число гамильтоновых путей. Это можно сделать при помощи динамического программирования за $O(2^n n^2)$, решив, таким образом, вторую подзадачу.

Подзадачи 3 и 4.

В полученном графе есть четыре вида вершин в зависимости от значений $needCarry_i$ и $makeCarry_i$. Обозначим эти виды следующим образом:

- Тип A: $needCarry_i = 0$, $makeCarry_i = 0$
- Тип B: $needCarry_i = 1$, $makeCarry_i = 0$
- Тип C: $needCarry_i = 0$, $makeCarry_i = 1$
- Тип D: $needCarry_i = 1$, $makeCarry_i = 1$

На рисунке показано, какие рёбра есть в графе, а также какие вершины могут быть начальными и конечными.

Посчитать гамильтоновы пути в таком графе можно динамикой за $O(n^4)$: $d_x[A][B][C][D]$ — число путей, которые заканчиваются в вершине вида x и проходящие через соответствующее число вершин каждого вида. Это решает подзадачу 3.

Чтобы решить подзадачу 4, нужно учесть, что нельзя начинать со столбца с нулём. Это влияет только на базу динамики:

в $d_A[1][0][0][0]$ и $d_B[0][1][0][0]$ нужно записать количество соответствующих столбцов, в которых нет нулей.

Подзадачи 5 и 6. На любом пути в таком графе, который начинается в A или B, количество вершин вида B не более чем на один превосходит количество вершин вида C. Это позволяет сократить число состояний в динамике до $O(n^3)$, решив, таким образом, подзадачи 5 и 6.

Подзадача 7. Обозначим через A, B, C, D количество вершин каждого вида. Любой интересующий нас путь выглядит следующим образом:

$$\_B\_C\_B\_C\_B\_C\ldots B\_C\_$$

При этом вершины типа A расположены в промежутках перед вершинам типа B, а также в последнем. Вершины типа D расположены в промежутках после вершин типа B. Кроме того, если $B \neq C$, то ответ 0.

Если $B = C = 0$ и $D > 0$, то ответ 0. Если $B = C = 0$ и $D = 0$, то ответ $A!$.

Если $B = C > 0$, то ответом будет произведение следующих чисел:

1. Количество способов расставить вершины B по местам: $B!$
2. Количество способов расставить вершины C по местам: $C!$
3. Количество способов расставить A вершин в $(B + 1)$ промежутков с учётом порядка: $\frac{(A+B)!}{B!}$.
4. Количество способов расставить D вершин в B промежутков с учётом порядка: $\frac{(D+B-1)!}{(B-1)!}$.

Ответ равен $(A + B)! \cdot (D + B - 1)!B$. Это решение за $O(n)$ проходит тесты подзадачи 7.

Подзадача 8. Осталось учесть, что среди вершин вида A и B может быть несколько вершин, с которых нельзя начинать (столбцы с нулями). Обозначим эти количества через A_0 и B_0.

Как и в прошлой подзадаче, если $B = C = 0$ и $D > 0$, то ответ 0. Если $B = C = 0$ и $D = 0$, то ответ $(A - 1)! \cdot (A - A_0)$.

Далее $B = C > 0$. Сначала отдельно посчитаем:

- ans_B — число путей, которые начинаются с вершины вида B. Это означает, что промежутков, в которые можно ставить вершины A, не $B+1$, а B. Тогда $ans_B = B! \cdot B! \cdot \frac{(A+B-1)!}{(B-1)!} \cdot \frac{(D+B-1)!}{(B-1)!} = (A+B-1)!(D+B-1)!B^2$
- ans_A — число путей, которые начинаются с вершины вида A. $ans_A = (A+B)!(D+B-1)!B - ans_B$

И тогда ответ равен $ans_A \cdot \frac{A-A_0}{A} + ans_B \cdot \frac{B-B_0}{B}$.

Задача 9. Сумма разностей

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

Подзадачи 1-2. Непосредственно вычислим требуемую сумму S. Поскольку всего можно составить $\frac{1}{2}n(n-1)$ различных пар разностей, то это решение имеет сложность $O(n^2)$. Кроме того, необходимо учесть, что сумма всех попарных разностей может превышать обычный целочисленный тип данных, и значит, будет числом типа `long long`. При правильной технической реализации решение набирает до 60 баллов.

Подзадача 3. Изменим форму записи для требуемой суммы попарных разностей. Заметим, что сумма S не зависит от расстановки чисел исходного массива, и значит, не меняется при любых перестановках чисел a_i. Отсюда следует, что можно предварительно отсортировать исходный массив чисел (после этой операции изменится лишь порядок расположения чисел), а затем подсчитать сумму S.

Итак, будем считать, что $a_1 \leqslant a_2 \leqslant a_3 \ldots \leqslant a_n$. Тогда все разности $a_i - a_j$ неположительны при всех $j > i$, и значит,

$$S = \sum_{i=1}^{n-1} \sum_{j>i}^{n} |\, a_i - a_j \,| = \sum_{i=1}^{n-1} \sum_{j>i}^{n} (a_j - a_i) =$$

$$= (n-1)a_n + (n-3)a_{n-1} + (n-5)a_{n-2} + \ldots = \sum_{i=1}^{n} (2i - 1 - n)a_i.$$

Это решение имеет сложность $O(n \log n)$, и оно проходит все тесты.

Задача 10. Полупростые делители

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

Сначала научимся подсчитывать количество полупростых делителей для произвольного натурального числа $n > 1$. Разложим число n на простые множители:

$$n = p_1^{\alpha_1} p_2^{\alpha_2} \cdot \ldots \cdot p_k^{\alpha_k}.$$

Каждый полупростой делитель n равен произведению либо двух различных простых множителей $p_i \cdot p_j$, либо является квадратом простого числа p_i^2. В первом случае таких делителей будет столько же, сколько существует способов выбрать два элемента из k-элементного множества $\{ p_1, p_2, \ldots, p_k \}$, то есть $\frac{1}{2}k(k-1)$. Во втором случае количество таких делителей совпадает с количеством простых чисел p_i, у которых показатели α_i не меньше 2. Итак, количество полупростых делителей n равно

$$\frac{1}{2}k(k-1) + \sum_{\alpha_i \geqslant 2} 1,$$

то есть находится в диапазоне между двумя соседними «треугольными» числами $\frac{1}{2}k(k-1)$ и $\frac{1}{2}k(k+1)$. Последнее число получается только в случае, когда все $\alpha_i \geqslant 2$ $(1 \leqslant i \leqslant k)$.

Поэтому алгоритм для нахождения наименьшего числа n следующий. Для заданного m сначала находим наименьшее число k, удовлетворяющее неравенствам

$$\frac{1}{2}k(k-1) \leqslant m < \frac{1}{2}k(k+1).$$

Если $m = \frac{1}{2}k(k-1)$, то искомое наименьшее число имет вид $n = p_1 p_2 \cdot \ldots \cdot p_k$. Действительно, все его полупростые делители — это числа вида $p_i \cdot p_j$, где $i \neq j$, их $C_k^2 = \frac{1}{2}k(k-1)$ штук.

Пусть $\frac{1}{2}k(k-1) < m < \frac{1}{2}k(k+1)$. Число $n = p_1 p_2 \cdot \ldots \cdot p_k$ имеет всего $\frac{1}{2}k(k-1)$ полупростых делителей. Недостающее количество полупростых делителей равно $l = m - \frac{1}{2}k(k-1)$. Включим в формулу произведения числа n первые l простых чисел с квадратами, то есть искомое число n составим в виде:

$$n = p_1^2 p_2^2 \cdot \ldots \cdot p_l^2 p_{l+1} \cdot \ldots \cdot p_k.$$

Осталось вычислить это произведение, взяв в качестве p_i наименьшие возможные простые числа, то есть $p_1 = 2$, $p_2 = 3$, $p_3 = 5$ и так далее.

Оглавление