Задача A. Плохое хеширование

Разработчик:	Иван Казменко
Имя входного файла:	bad-hashing.in
Имя выходного файла:	bad-hashing.out
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Маленький мальчик Петя решает задачу про строки. Не будем останавливаться подробно на её условии. Для нас важно только то, что задача сводится к следующей: дано несколько строк, состоящих исключительно из маленьких букв английского алфавита, и нужно брать пары строк и проверять, равны ли они.

Петя придумал, как делать это быстро: сначала он посчитал *хеш-функцию* от каждой строки, а затем, когда нужно сравнить две строки, просто сравнивает значения хеш-функции от этих строк. Конечно, если значения хеш-функции различны, строки тоже различны. А вот если значения одинаковы, это ещё не гарантирует равенства самих строк.

Мы хотим *взломать* Петино решение, то есть придумать такие две различные строки, что значения хеш-функции от них одинаковы. Чтобы сделать это, разберёмся, что за функцию Петя использует для хеширования.

При ближайшем рассмотрении оказалось, что Петя реализовал *полиномиальный хеш* от строки. Полиномиальный хеш задаётся множителем p и модулем q. Для пустой строки ε значение хеш-функции $h(\varepsilon) = 0$, а для любой строки S и любого символа c хеш-функция рекуррентно определяется как $h(S + c) = (h(S) \cdot p + \text{code}(c)) \bmod q$. Здесь $\text{code}(c)$ — это ASCII-код символа c. Как известно, коды маленьких букв английского алфавита идут подряд: code('a') = 97, code('b') = 98, ..., code('z') = 122. Можно выписать и нерекуррентную формулу: если строка $S = s_1 s_2 \dots s_n$, то $h(S) = (\text{code}(s_1) \cdot p^{n-1} + \text{code}(s_2) \cdot p^{n-2} + \dots + \text{code}(s_n) \cdot p^0) \bmod q$.

Это достаточно распространённый метод хеширования, однако Петя не подумал о том, что его решение могут взломать, и допустил две существенные ошибки при выборе p и q. Во-первых, модуль q слишком мал, он равен всего лишь 2^{32} (Петя просто считает значение хеш-функции в 32-битном целом беззнаковом типе данных и не обращает внимания на переполнение). Во-вторых, при этом множитель p чётный.

Зная множитель p, взломайте решение Пети.

Формат входных данных

Первая строка ввода содержит целое число p — множитель полиномиального хеширования ($0 < p < 2^{32}$). Гарантируется, что p чётно.

Формат выходных данных

В первых двух строках выведите две различные строки S и T, для которых $h(S) = h(T)$. Строки должны состоять исключительно из маленьких букв английского алфавита (ASCII-коды 97–122) и иметь длину от 1 до 100 000 символов. Заметим, что длины строк не обязательно должны совпадать. Если возможных ответов несколько, разрешается вывести любой из них.

Примеры

bad-hashing.in	bad-hashing.out
4	ae ba
1000	vabavydw budqhmng

Пояснение к примерам

В первом примере $h(S) = (97 \cdot 4 + 101) \bmod 2^{32} = 489$ и

$h(T) = (98 \cdot 4 + 97) \bmod 2^{32} = 489$.

Во втором примере $h(S) = 118\,097\,098\,097\,118\,121\,100\,119 \bmod 2^{32} = 834\,470\,743$ и

$h(T) = 98\,117\,100\,113\,104\,109\,110\,103 \bmod 2^{32} = 834\,470\,743$.

Задача B. Мосты: последняя битва (Division 1 Only!)

Разработчик:	Сергей Копелиович
Имя входного файла:	`bridges3.in`
Имя выходного файла:	`bridges3.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

А у вашей дипломной работы есть практическое применение?

Популярный вопрос

Легенда

Мальчик Серёжа уже почти закончил работу над дипломной работой. Тема диплома с декабря не изменилась, это всё ещё «Dynamic 2-Edge-Connectivity Problem». Алгоритм уже придуман, протестирован, доказана оценка $O(K \log K)$... Осталось только написать главу про «практическое применение».

Ну какое может быть практическое применение у задачи «Dynamic 2-Edge-Connectivity Problem»? Вопрос непростой. Он оказался чуть ли не сложнее исходной задачи. Но диплом нужно скоро защитить, так что срочно нужно что-то делать.

Итак, первое практическое применение: можно предложить на соревнование задачу на эту тему!

Задача

Дан неориентированный граф из не более чем 10^5 вершин. Изначально граф не содержит рёбер. Вам нужно обрабатывать запросы вида `ADD x y` и `DEL x y` — добавить или удалить ребро из x в y.

В каждый момент времени нужно знать, **сколько в графе мостов**.

Кратных рёбер и петель ни в какой момент времени нет.

Если поступает команда «удалить ребро», оно в графе точно присутствует.

Решение задачи

Диплом на то и диплом, что за 5 часов его так просто не напишешь. Поэтому вам даны целых 5 подсказок.

1. Добавить ребро и удалить ребро — сделать ребро «живым» на некотором отрезке времени.
2. Используйте метод «Разделяй и Властвуй».
3. Сжимайте компоненты двусвязности.
4. Даже когда компоненты двусвязности уже сжаты, если запросов мало, граф всё ещё можно сильно уменьшить.
5. Существует решение за $O(K \log K)$.

Формат входных данных

В первой строке записаны целые числа N и K: количество вершин и количество запросов, соответственно. $1 \leqslant N \leqslant 10^5$, $1 \leqslant K \leqslant 10^5$.

В следующих K строках по одному в строке записаны запросы. Каждый запрос начинается словом «`ADD`» или «`DEL`» для запроса на добавление или удаление ребра, соответственно. После этого слова записаны два целых числа a и b, задающих ребро. $1 \leqslant a, b \leqslant N$, $a \neq b$.

Формат выходных данных

После каждого запроса нужно вывести одно число — сколько в текущем графе мостов.

Пример

bridges3.in	bridges3.out
4 8 ADD 1 2 ADD 2 3 ADD 1 3 DEL 2 3 DEL 1 2 ADD 2 4 ADD 1 4 ADD 2 3	1 2 0 2 1 2 3 0

Задача C. Ненависть к литературе

Разработчик:	Сергей Копелиович
Имя входного файла:	`covertexts.in`
Имя выходного файла:	`covertexts.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Страх ведёт к гневу, гнев к ненависти,
ненависть — залог страданий

Мастер Йода

Коля с детства не любил классическую литературу. Особой нелюбовью у него всегда пользовались K самых запомнившихся ещё со школьной скамьи произведений.

Коля вырос. Теперь он занимает руководящий пост в министерстве образования. Пришло время осуществиться детской мечте. Но нельзя просто так сказать общественности: «Война и мир» ужасна! Коллеги не поддержат Колю в таких радикальных высказываниях. Поэтому был разработан хитрый план. На общей волне борьбы со сквернословием Коля может провести закон о том, что некоторые слова признаются «нехорошими». Любые тексты, содержащие «нехорошие» слова, конечно, в школе проходить никак нельзя. Да и публиковать не стоит. Осталось только выбрать слова, которые необходимо запретить. Талант убеждения Николая настолько велик, что он может вписать в закон абсолютно любое слово. Но добавление каждого нового слова в закон требует немалых сил и финансовых затрат, поэтому Коля просит Вас о помощи.

Напишите программу, которая по заданным текстам выберет такой набор слов, что:

1. В каждый из текстов входит хотя бы одно выбранное слово
2. Количество слов в наборе минимально

Формат входных данных

Количество текстов K ($1 \leqslant K \leqslant 12$). Далее K строк длины не более 10^5 — литературные произведения, так нелюбимые Колей. Все тексты состоят только из пробелов и маленьких латинских букв. Словом называется любая подстрока текста, состоящая только из букв и обрамлённая и справа, и слева пробелом или концом строки. В каждом тексте содержится хотя бы одно слово. Будьте внимательны, пробелы могут быть везде и в большом количестве.

Формат выходных данных

Минимальное количество слов, которые нужно признать «нехорошими». Далее сами слова, каждое слово на отдельной строке. Слова можно выводить в произвольном порядке. Если ответов с минимальным количеством слов несколько, разрешается вывести любой из них.

Пример

covertexts.in	covertexts.out
4 a b c sideli na trube c kakogo perepuga na nana la lala vot takaya pesnya etot salata iz kakogo to m	2 kakogo na

Замечание

В тестах жюри, конечно, присутствуют не только литературные тексты.

Задача D. Словарь

Разработчик:	Наталья Гинзбург
Имя входного файла:	`dictionary.in`
Имя выходного файла:	`dictionary.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Герцог Бэкингемский в затруднительном положении.

Еще неделю назад, отдыхая в своей библиотеке, он любовался щедро украшенным двенадцатитомным словарем, драгоценностью французской королевской фамилии. Словарь был тайно пожалован ему королевой Франции Анной Австрийской в знак особого расположения. Герцог успел бегло пролистать несколько томов, и пришел к выводу, что словарь скорее декоративный: записи, которые там содержались, казались бессмысленными, как будто кто-то незнакомый с английским языком пытался изобразить его слова, случайно расставляя буквы. Да и кроме «слов» там ничего не было — ни перевода, ни толкований. Зато каждая буква была выведена с особой любовью и старательностью, каждое слово находилось на отдельной строке, никакое из слов не повторялась и все они были аккуратно упорядочены в обычном словарном порядке. Кроме того, все тома были пронумерованы, и все они содержали одинаковое количество слов.

Вчера, однако, случилась большая неприятность: после бала в Бэкингемском дворце один из томов бесценного словаря исчез (заметим, что это был не первый том, и не последний).

А сегодня к герцогу Бэкингемскому ворвались гонцы от королевы Франции со срочной просьбой вернуть королеве все двенадцать томов — и без малейшего промедления! Как оказалось, враги королевы выследили ее, узнали о ее подарке герцогу, и, желая скомпрометировать несчастную Анну Австрийскую перед ее супругом, королем Франции, подсказали ему устроить через пару недель костюмированный вечер в парижской ратуше — и непременно с гаданием по злосчастному словарю.

Словарь обязательно надо вернуть королеве как можно скорее, но найти похищенный том вряд ли удастся, поэтому герцог готов изготовить точную его копию — по крайней мере, такую, чтобы никто не догадался о подмене.

Иначе честь королевы будет запятнана, а этого герцог допустить не может.

К счастью, к услугам герцога лучший переплетчик Англии, чтобы переплести новый том так же, как переплетены остальные одиннадцать, лучший ювелир, чтобы его украсить, лучший писарь, чтобы вписать слова, гонцы королевы, чтобы отправить словарь во Францию, — и вы, чтобы оценить, сколько времени на все это понадобится. Оценивать нужно то время, которое потратит писарь, — все остальные со своими оценками уже определились.

Герцог сообщил вам с писарем количество слов в каждом томе, последнее слово предыдущего тома, и первое слово следующего (оба слова состояли только из английских букв, и писарю тоже придется ограничиться английским адфавитом). А в выборе слов, которые придется вписать, предоставил некоторую свободу — все равно их не восстановить точно, поэтому важен только словарный порядок, которому должны подчиняться все вписанные слова. Кроме того, он напомнил, что слова во всем словаре не должны повторяться (слова, которые различаются только регистром, герцог велел считать одинаковыми — иначе говоря, Вы можете рассматривать только слова, состоящие из строчных английских букв).

Копию нужно изготовить как можно быстрее, поэтому писарь будет стремиться ограничиться минимальным количеством букв. Выведите минимальное количество букв, которое нужно вписать, или -1, если задание герцога выполнить невозможно.

Формат входных данных

В первой строке ввода задано одно целое число n ($1 \leqslant n \leqslant 10^9$) — количество слов в любом томе словаря. Во второй строке задано последнее слово предыдущего тома. В третьей строке задано первое слово следующего тома. Оба слова состоят только из строчных букв английского алфавита, и имеют длину от 1 до 100 000 символов.

Формат выходных данных

Выведите единственное целое число — ответ на вопрос герцога, или -1, если его задание выполнить невозможно.

Примеры

dictionary.in	dictionary.out
2 ab ba	3
3 ba ab	-1
5 abcdabcd abcdabcz	40

Пояснение к примерам

В первом примере одно из оптимальных решений — вписать слова «ac» и «b».

Во втором примере не существует ни одного слова, которое может идти следом за последним словом предыдущего тома и перед первым словом следующего тома (герцог явно где-то ошибся и его задание выполнить невозможно).

Задача E. Игра на графе

Разработчик:	Сергей Копелиович
Имя входного файла:	`graphgame.in`
Имя выходного файла:	`graphgame.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Есть ориентированный граф. В одной из вершин графа стоит фишка. Двое — Саша и Александра — играют в игру. Нужно за один ход (его делают оба игрока сообща) сдвинуть фишку по одному из рёбер. Ход делается так: из вершины, в которой сейчас стоит фишка, выходят K рёбер. На доске записываются числа от 1 до K и игроки их по очереди вычёркивают. Первой всегда вычёркивает Саша. Последнее вычеркнутое число — номер ребра, по которому делается ход.

В этом графе есть особые вершины типа 1 — выигрышные для Саши. А ещё есть особые вершины типа 2 — выигрышные для Александры.

Как только фишка попадает в одну из таких вершин, игра заканчивается победой одного из двух игроков. Если игра продолжается бесконечно, её называют ничейной.

Вам дан граф и начальное положение фишки. Кто выиграет при оптимальной игре обеих сторон?

Формат входных данных

В первой строке ввода заданы числа N, M, S, K_1, K_2 — число вершин и рёбер в графе, начальное положение фишки, число вершин первого типа и число вершин второго типа. Вторая строка содержит K_1 чисел от 1 до N — вершины первого типа. В третьей строке находится K_2 чисел от 1 до N — вершины второго типа. Все $K_1 + K_2$ чисел различны.

Следующие M строк содержат рёбра. Ребро из вершины a в вершину b задаётся парой чисел (a, b).

Ограничения:

- $1 \leqslant n, m \leqslant 100\,000$.
- Из всех вершин, кроме выигрышных, выходит хотя бы по одному ребру.

Формат выходных данных

Исход игры — `Sasha wins`, или `Alexandra wins`, или `Draw`.

Примеры

graphgame.in	graphgame.out
2 4 1 1 0 2 1 1 1 2 2 1 2 2	Sasha wins
3 9 1 1 1 2 3 1 1 1 2 1 3 2 1 2 2 2 3 3 1 3 2 3 3	Draw

Задача F. Митинг «За честные выборы» (Division 1 Only!)

Разработчик:	Олег Давыдов
Имя входного файла:	`meeting.in`
Имя выходного файла:	`meeting.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Вот я бы на их месте, я бы... Я бы!!!

Один знакомый программист.

21-е декабря 2012 года. В разгаре очередной (54-й по счёту) митинг оппозиции. Тысячи людей пришли на Болотную площадь. Звучат лозунги, которые мы не будем приводить в условии этой задачи (по политическим причинам).

В какой-то момент начались беспорядки. Руководство полиции приняло решение беспорядки прекратить, задержав часть участников митинга. Известно, что в толпе кроме оппозиционеров присутствуют участники движения «Наши». Капитан подразделения ОМОНа поручил Вам, как главному программисту, быстро спланировать задержание таким образом, чтобы все оппозиционеры оказались задержаны, а среди участников движения «Наши» суммарная ценность задержанных была минимальна.

Оперативный расклад следующий: на площади находится n оппозиционеров и m «нашистов». Оппозиционеры отмечены на оперативной схеме красными точками, «нашисты» — синими. Благодаря грамотной работе сотрудников полиции известны координаты (для удобства проведения митингов за 2012-й год на Болотной площади успели ввести декартову систему координат) каждого, находящегося на площади. Можно считать, что каждый, кто там есть — либо «нашист», либо оппозиционер: люди, не входящие в движение «Наши», автоматически причисляются к оппозиции и подлежат задержанию.

Предполагаемый план действий следующий: из бойцов ОМОНа выстроить живую цепь, которая будет представлять из себя прямую, и задержать всех без исключений, кто находится с одной стороны от этой прямой. Все люди, находящиеся на самой прямой, задерживаются или не задерживаются по усмотрению ОМОНа (каждого из находящихся на прямой, не зависимо от других, можно задержать или не задержать).

В результате должны быть задержаны все оппозиционеры. К сожалению не всегда получится отделить оппозиционеров и «нашистов». Для того, чтобы возможно было принять объективное решение, про каждого «нашиста» вам предоставили секретную информацию — его Партийную Ценность (или, более коротко, ценность). Требуется спланировать ход операции так, чтобы суммарная ценность задержанных «нашистов» была минимальна.

Строго говоря, вам необходимо найти прямую такую, что:

- Все оппозиционеры окажутся (нестрого) с одной стороны прямой.
- Суммарная партийная ценность «нашистов», оказавшихся (строго) с той же стороны прямой, минимальна.

Формат входных данных

В первой строке входного файла даны два целых неотрицательных числа: n и m. В следующих n строках даны координаты оппозиционеров (по два целых числа на строку). Затем идут m строк с описанием «нашистов», по три целых числа на строку: координаты и партийная ценность.

Ограничения:

- По оперативной информации, на площади суммарно не более 200 000 человек.
- Координаты указаны в специальных единицах, так что все они целые и не превосходят 10^6 по абсолютному значению.

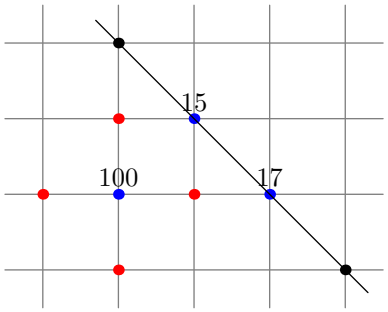

Рис. 1: Оперативная схема для площади из первого примера

- Партийная ценность каждого «наиста» — целое число от 0 до 1 000.

Формат выходных данных

На первой строке выходного файла выведите одно целое число: суммарную ценность **не задержанных** «наистов». Оно понадобится для отчётности. На второй строке выведите четыре целых числа: координаты двух различных точек, через которые нужно провести прямую. Выведенные числа не должны превосходить 10^9 по абсолютной величине.

Пример

meeting.in	meeting.out
4 3 1 0 2 1 0 1 1 2 2 2 15 1 1 100 3 1 17	32 1 3 4 0

Замечание

Если ваши политические взгляды не позволяют вам писать программу, предназначенную для задержания оппозиционеров, помните, что программа универсальна и красные с синими точками всегда можно поменять местами :-)

Задача G. Многоугольник (Division 1 Only!)

Разработчик:	Ольга Бурсиан
Имя входного файла:	`polygon.in`
Имя выходного файла:	`polygon.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Некоторое время назад у Васи был многоугольник из картона. Вася положил его на клетчатый листок бумаги, обвёл границу и получил первый многоугольник на плоскости. Потом он снова взял свой многоугольник из картона и положил на листок, возможно, в другом месте и, возможно, перевернув при этом многоугольник. Вася снова обвёл границу и получил второй многоугольник на плоскости. Таким образом, второй многоугольник получен из первого некоторым изометрическим преобразованием, то есть преобразованием плоскости, сохраняющим расстояния между точками.

Сейчас у него остался рисунок и уже нет картонного многоугольника. Он хочет опять получить из первого многоугольника второй, но теперь он может сделать только несколько преобразований следующего вида: начертить некоторую прямую, отразить первый многоугольник относительно неё (согнув по ней листок), получить снова некоторый многоугольник, начертить ещё одну прямую, отразить полученный многоугольник относительно новой прямой и так далее. В результате последовательности этих преобразований Вася надеется получить второй многоугольник из первого. Вася не хочет сильно помять рисунок, поэтому он готов сделать не более трёх преобразований. Помогите ему их сделать.

Каждая ось отражения задаётся уравнением прямой $ax + by + c = 0$. Выведите коэффициенты осей отражения a, b и c, относительно которых нужно отражать первый многоугольник, чтобы получить второй.

Формат входных данных

В первой строке входного файла дано число n — количество точек в многоугольнике ($3 \leqslant n \leqslant 100\,000$). Далее во второй строке даны $2n$ вещественных чисел $x_1^{(1)}$, $y_1^{(1)}$, $x_2^{(1)}$, $y_2^{(1)}$, …, $x_n^{(1)}$, $y_n^{(1)}$ — координаты первого многоугольника в порядке обхода против часовой стрелки. В третьей строке также даны $2n$ вещественных чисел $x_1^{(2)}$, $y_1^{(2)}$, $x_2^{(2)}$, $y_2^{(2)}$, …, $x_n^{(2)}$, $y_n^{(2)}$ — координаты второго многоугольника в порядке обхода против часовой стрелки. Все координаты не превосходят 100 по абсолютной величине и заданы не более чем с десятью знаками после десятичной точки.

Гарантируется, что многоугольники не имеют самопересечений и самокасаний, а кроме того, первый многоугольник переходит во второй в результате изометрического преобразования. Заметим, однако, что при этом преобразовании вершины первого многоугольника могут переходить в вершины второго с другими номерами.

Входные данные построены таким образом, чтобы избежать большей части проблем, связанных с точностью. Выполнены следующие формальные условия:

1. Стороны многоугольника и углы многоугольника в радианах считаются равными, если они отличаются не более чем на 10^{-8}.
2. Гарантируется, что стороны и углы, не считающиеся равными, отличаются как минимум на 10^{-6}.
3. Любые две вершины одного многоугольника различны; более точно, попарные расстояния между вершинами не меньше 10^{-6}.
4. Каждое ребро имеет длину не меньше 10^{-1}.
5. Никакие три идущие подряд вершины не лежат на одной прямой; более точно, каждый угол каждого многоугольника в радианах отличается от углов, кратных π, как минимум на 10^{-3}.

Формат выходных данных

В первой строке выходного файла выведите целое число m — количество осей отражения ($0 \leqslant m \leqslant 3$; заметим, что m не нужно минимизировать). В следующих m строках выведите по три вещественных числа в каждой — коэффициенты a, b и c осей отражения в том порядке, в котором Васе необходимо их применить. Необходимо, чтобы было выполнено неравенство $a^2 + b^2 > 0$. Выводите вещественные числа как можно более точно! При проверке совпадения многоугольника,

полученного из первого в результате всех отражений, со вторым многоугольником сравнение координат вершин производится с точностью 10^{-4}. Если возможных ответов несколько, можно вывести любой из них. Гарантируется, что для любого возможного теста существует хотя бы один ответ.

Пример

polygon.in
3 1.0 5.0 1.0 1.0 3.0 1.0 4.0 1.0 8.0 1.0 8.0 3.0
polygon.out
2 -7.000000000000 0.000000000000 31.500000000000 -2.000000000000 -2.000000000000 18.000000000000
polygon.in
5 1.0 1.0 4.5 1.0 3.0 5.5 1.0 4.0 2.0 3.0 8.0 2.0 9.0 1.0 10.5 3.0 6.0 4.5 6.0 1.0
polygon.out
3 -8.000000000000 3.000000000000 32.500000000000 -0.410958904110 -1.095890410959 4.794520547945 1.000000000000 1.000000000000 -10.000000000000

Задача H. Распознай-ка

Разработчик:	Сергей Копелиович
Имя входного файла:	`segments.in`
Имя выходного файла:	`segments.out`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мебибайт

Тебе еще многому нужно учиться, юный падаван

Оби-Ван Кеноби

На плоскости были нарисованы N отрезков с целыми координатами концов. Длины отрезков были не меньше 50. Для любой пары отрезков угол между прямыми, образованными отрезками, был больше $\frac{\pi}{8} - 10^{-6}$. Из этого, например, следует, что $N \leqslant 8$.

Теперь есть координатная сетка, столбцы которой пронумерованы от 0 до $W-1$, а строки — от 0 до $H-1$. Клеткам сетки соответствуют точки на плоскости с такими же координатами. Изначально все клетки сетки белые.

N отрезков нарисованы на сетке чёрным цветом. Клетка покрашена в чёрный цвет, если существует хотя бы один отрезок, расстояние от которого до клетки не превосходит единицы.

Отрезки целиком поместились на сетку (если бы сетка содержала все целые точки плоскости от $-\infty$ до ∞, никакие точки, кроме уже покрашенных, не пришлось бы красить).

Формат входных данных

В первой строке ввода даны два числа H и W — высота и ширина сетки ($1 \leqslant H \leqslant 300$, $1 \leqslant W \leqslant 300$). Далее идут H строк по W символов — сама координатная сетка. Символ «*» обозначает чёрную краску, а символ «.» — белую краску.

Формат выходных данных

Одно число — количество отрезков в исходном наборе.

Пример

segments.in	segments.out
25 30 ....*.....*.. .***.....***. ..***.....**.. ..*****.....**... ...*****.....**... ...**.***.....**.... ...***.***.....***.... ....**..****.....****.... ....**....***.....******..... .....**....***..********..... .....**.....*******..**..... .....***...******...***..... .....**.*****.***..**..... .....******....*****..... ...******.....****..... ..****.**.....****..... ...*...***.....******..... .....**.....**..***..... .....**.....**....***.... .....**.....**.....****... .....**....**.....***.. .....***..***.....***. .....*....*.....*..	4

Замечание

Первый тест в тестирующей системе совпадает с тестом из примера.

Пример формально не соответствует условию задачи, так как длины нарисованных отрезков слишком малы. Тем не менее, мы надеемся, пример поможет правильному пониманию условия, тестированию решений, а правильным решениям не помешает получить Accepted.

Задача I. Очередная задача про подстроки (Division 1 Only!)

Разработчик:	Сергей Копелиович
Имя входного файла:	substr7.in
Имя выходного файла:	substr7.out
Ограничение по времени:	2.5 секунды
Ограничение по памяти:	256 мебибайт

Неужели это тоже баян?...

Разговор бывалых людей

Даны два набора строк. Строки состоят из маленьких символов латинского алфавита. Суммарная длина строк в каждом из наборов не более 10^5.

Нужно для каждой строки первого набора найти такую максимальную по длине подстроку, что она встречается, как подстрока, в одной из строк второго набора.

Формат входных данных

В первой строке ввода дано описание первого набора строк. Формат описания следующий: сперва N ($1 \leqslant N \leqslant 10^5$) — количество строк, затем сами строки, записанные через пробел.

Во второй строке ввода дано описание второго набора строк в таком же формате.

Формат выходных данных

Для каждой строки первого набора выведите четыре числа — длину найденной подстроки, позицию подстроки в строке, номер парной строки из второго набора, позицию подстроки в парной строке из второго набора.

Строки и позиции внутри строк нумеруются с единицы. Выводить ответы для строк первого набора нужно в том же порядке, в каком строки первого набора даны во входных данных.

Если оптимальных по длине подстрок несколько, можно выбрать любую.

Пример

substr7.in	substr7.out
6 aba mama abacaba x opa z 2 cabama xopa	3 1 1 2 3 2 1 4 4 4 1 1 1 1 2 1 3 1 2 2 0 1 1 1

Задача J. Level of Acidity (Division 2 Only!)

Разработчик:	*не указан*
Имя входного файла:	`acid.in`
Имя выходного файла:	`acid.out`
Ограничение по времени:	3 seconds
Ограничение по памяти:	256 mebibytes

В рамках экологической экспедиции вы проводите измерения уровня кислотности очень длинной реки. Для этого вы разместили N датчиков в реке. Каждый датчик выдаёт в качестве результата измерения целое число R. В рамках исследований вы хотите найти модуль разности между двумя наиболее часто встречающимися результатами измерений.

При этом если существует более, чем один результат, встречающийся наибольшее количество раз (обозначим такие результаты как A_i), искомое значение равно максимальному значению $|A_i - A_j|$. Если наибольшее количество раз встречается ровно один результат A, но следующих по частоте результатов B_i несколько, то требуемое значение равно максимуму $|A - B_i|$.

Формат входных данных

Первая строка входного файла содержит одно целое число N ($2 \leqslant N \leqslant 2 \cdot 10^6$) — количество датчиков. Каждая из последующих N строк содержит целое число R ($1 \leqslant R \leqslant 1000$) — значение измерения соответствующим датчиком. Гарантируется, что как минимум два из этих значений различны.

Формат выходных данных

Выведите одно целое число — модуль разности между двумя наиболее часто встречающимися результатами измерений, вычисленный в соответствии с требованиями задачи.

Примеры

acid.in	acid.out
5 1 1 1 4 3	3
4 10 6 1 8	9

Задача K. Roman and Arabic (Division 2 Only!)

Разработчик:	*не указан*
Имя входного файла:	`aromatic.in`
Имя выходного файла:	`aromatic.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 mebibytes

В этой задаче мы рассмотрим «ароматную» (от английских слов «arabic» и «roman») запись — способ записи целых чисел с использованием римских и арабских цифр.

«Ароматная» запись имеет вид $A_1R_1A_2R_2 \dots A_nR_n$, где A_i — арабские цифры, а R_i — римские. Каждая пара A_iR_i задаёт значение в соответствии с описанием, следующим ниже, итоговое значение вычисляется с помощью сложения или вычитания полученных результатов. Напомним, что A_i может принимать значения 0, 1, 2, 3, 4, 5, 6, 7, 8 9, а R_i — значения I, V, X, L, C, D, M (1, 5, 10, 50, 100, 500, 1000 соответственно).

Значение пары AR есть значение A, умноженное на значение R. Полученные произведения складываются, за исключением ситуации, когда в записи встречаются идущие подряд символы $ARA'R'$, при этом значение R' строго больше значения R. В этом случае значение пары AR вычитается из общей суммы, а не прибавляется к ней.

Например, запись $3M1D2C$ задаёт число $3 \cdot 1000 + 1 \cdot 500 + 2 \cdot 100 = 3700$, а запись $3X2I4X$ — число $3 \cdot 10 - 2 \cdot 1 + 4 \cdot 10 = 68$.

По заданной «ароматной» записи числа найдите его десятичную запись.

Формат входных данных

В первой строке входного файла содержится строка длиной не менее 2 и не более 20 символов — «ароматная» запись некоторого числа. Гарантируется, что запись корректна.

Формат выходных данных

Выведите десятичную запись числа, задаваемого «ароматной» записью из входного файла.

Примеры

aromatic.in	aromatic.out
3M1D2C	3700
2I3I2X9V1X	-16

Задача L. Giant Soccer (Division 2 Only!)

Разработчик:	*не указан*
Имя входного файла:	`giantsoccer.in`
Имя выходного файла:	`giantsoccer.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 mebibytes

В игре «гигантский футбол» правила отличаются от обычного футбола. Во-первых, каждая команда состоит из 99 игроков. Во-вторых, гол засчитывается только в случае, когда номера на футболках четырёх игроков, последними коснувшихся мяча, расположены строго по возрастанию (игрок с наименьшим из этих номеров начинает атаку, игрок с наибольшим из этих номеров забивает гол). Игроки имеют на своих футболках номера от 1 до 99, при этом никакие два номера не повторяются.

Вам известен номер на футболке игрока, забившего гол. Вычислите, сколько существует различных комбинаций, после которых гол будет засчитан.

Формат входных данных

В первой строке входного файла задано целое число J ($1 \leqslant J \leqslant 99$) — номер на футболке игрока, забившего гол.

Формат выходных данных

Выведите одно число — количество возможных комбинаций, после которых гол игрока с номером J будет засчитан.

Примеры

giantsoccer.in	giantsoccer.out
4	1
2	0
90	113564

Задача M. Sticks (Division 2 Only!)

Разработчик:	*не указан*
Имя входного файла:	`sticks.in`
Имя выходного файла:	`sticks.out`
Ограничение по времени:	2 seconds
Ограничение по памяти:	256 mebibytes

Пусть E —это множество, состоящее из элементов {/, \, -, |, .}, т.е. элементы этого множества — это диагонали единичного квадрата, отрезки, соединяющие середины противоположных сторон, а также точка — центр этого квадрата. Дан прямоугольник, состоящий из единичных квадратов, в каждом из которых отмечен ровно один из элементов множества E. Назовем сегментом такого прямоугольника набор единичных квадратов указанного прямоугольника, такой, что отмеченные элементы этого набора образуют отрезок, причем этот набор не может быть расширен до другого набора квадратов, отмеченные элементы которого также образуют отрезок.

Задача: для указанного во входном файле прямоугольника с отмеченными элементами из множества E найти число всех сегментов.

Формат входных данных

Первая строка входного файла содержит два целых числа n и m ($1 \leqslant n, m \leqslant 100$) — высота и ширина прямоугольника соответственно.

Следующие n строк содержат по m символов «/», «\», «-», «|» или «.», задающих прямоугольник.

Формат выходных данных

Выходной файл должен содержать одно целое число — количество сегментов.

Примеры

sticks.in	sticks.out
4 5 ..|/. ..|.. ./|.. -----	4