Задача A. Декомпозиция строки

Имя входного файла:	decomp.in
Имя выходного файла:	decomp.out
Ограничение по времени:	2 секунд
Ограничение по памяти:	64 мегабайт

Для строки α и целого числа n определим n-ю степень строки α^n как конкатенацию n копий строки α. Например, $\mathtt{aab}^4 = \mathtt{aabaabaabaab}$.

Любая строка S может быть представлена в виде разложения $S = S_1^{d_1} S_2^{d_2} \ldots S_k^{d_k}$. Вообще говоря, такое разложение может быть не единственным. Весом разложения строки S в указанном виде назовем сумму $|S_1| + |S_2| + \ldots + |S_k|$, где $|Z|$ означает длину строки Z.

По заданной строке S найдите ее разложение с минимальным весом.

Формат входного файла

Входной файл содержит строку S. S состоит из заглавных латинских букв и имеет длину не более $5\,000$.

Формат выходного файла

Первая строка выходного файла должна содержать w — минимальный возможный вес разложения строки S. Пусть k — число элементов в таком разложении. Тогда следующие k строк должны содержать элементы разложения: строку S_i и степень d_i, разделенные ровно одним пробелом.

Если существует несколько оптимальных решений, выведите любое из них.

Пример

decomp.in	decomp.out
ABABAAABABA	5 AB 2 A 3 BA 2

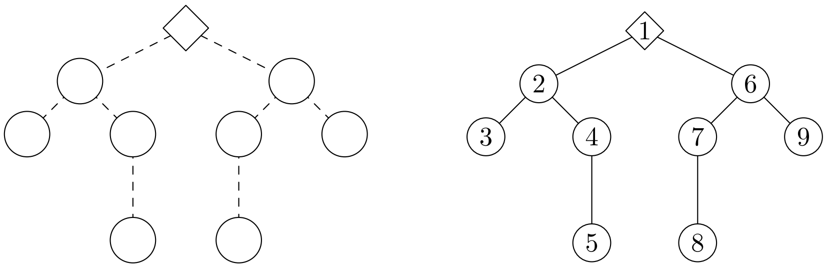
Задача B. Собери сам

Имя входного файла:	`doityourself.in`
Имя выходного файла:	`doityourself.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

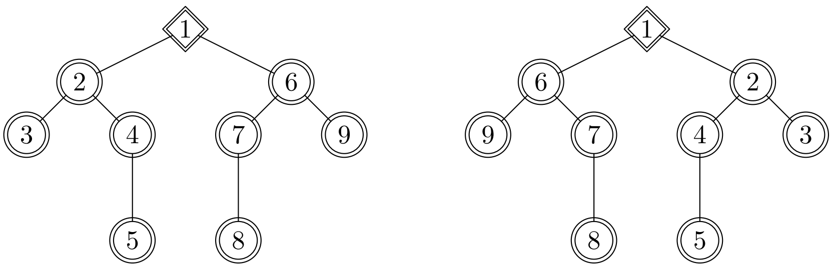
Компания «Ёжики-Хрюшечки» готовит новый конструктор «Собери сам» для детей младшего школьного возраста. Одна из главных составных частей конструктора — электронное устройство, для работы которого надо соединить набор клемм на специальной доске проводами.

Провода, которыми требуется выполнить соединение, имеют топологическую структуру дерева, в вершинах которого расположены контакты, подключаемые к клеммам. В свою очередь, доска имеет изображенную на ней схему укладки проводов, поэтому на первый взгляд кажется, что подключить устройство очень просто. Однако, к сожалению, все контакты на проводах, кроме одного выделенного, подключаемого к «корню» дерева, одинаковые, поэтому понять какой контакт куда подключить непросто.

Например, рассмотрим изображенную на рисунке схему.



На этой схеме возможны два способа подключить провода, они показаны на следующем рисунке.

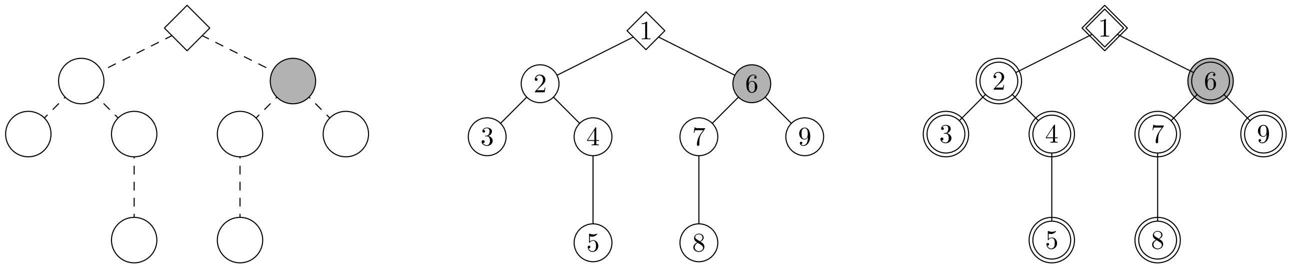


Разработчики из компании «Ёжики-Хрюшечки» не хотят, чтобы у схемы было несколько возможных подключений. Конечно, можно было бы пометить каждый контакт и каждую клемму уникальным кодом, чтобы ясно было что куда надо подключить, но тогда игра получилась бы не слишком интересной. Поэтому они решили раскрасить клеммы и контакты в разные цвета, так, чтобы были выполнены следующие условия:

1. контакт и клемма, которые необходимо соединить, раскрашены в один цвет;
2. существует ровно один способ соединить контакты и клеммы, чтобы не каждый контакт был соединен с клеммой того же цвета, провода проходили вдоль соответствующих линий на схеме и выделенный контакт соединялся с выделенной клеммой в корне дерева.

Разумеется, разработчики хотели бы использовать по возможности меньшее число цветов.

Например, для схемы, изображенной на рисунке выше, достаточно двух цветов:



По заданному дереву проводов определите, в какое минимальное число цветов его можно раскрасить, чтобы выполнить приведенные условия.

Формат входного файла

Первая строка входного файла содержит число n — количество вершин дерева ($1 \le n \le 500$). Пусть вершины дерева пронумерованы числами от 1 до n, так что номер родителя вершины меньше ее номера. Корень дерева имеет номер 1.

Вторая строка входного файла содержит $n-1$ число, для каждой вершины, начиная со второй, указан номер ее родителя.

Формат выходного файла

Первая строка выходного файла должна содержать число k — минимальное необходимое количество цветов. Следующая строка должна содержать n целых чисел — цвета вершин.

Пример

doityourself.in	doityourself.out
9 1 2 2 4 1 6 7 6	2 1 1 1 1 1 2 1 1 1

Задача C. Неправильный RSA

Имя входного файла:	`false.in`
Имя выходного файла:	`false.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайт

Рома, Сережа и Андрюша решили улучшить знаменитый алгоритм шифрования RSA. Они думаю, что ограничение, что в RSA в качестве модуля можно использовать только число n, являющееся произведением двух различных простых чисел — неудачное. Вместо него они предлагаю ввести новое ограничение — n должно быть произведением k-х степеней двух различных простых чисел: $n = p^k q^k$.

Однако Коля указал, что помимо различных математических трудностей, новая схема может оказаться менее устойчивой к взлому. А именно, большое число, равное произведению двух различных простых, тяжело разложить на множители, в частности, поскольку у него существует ровно одно нетривиальное разложение. А у числа вида $n = p^k q^k$ их может быть больше. Например, у числа $100 = 2^2 \cdot 5^2$ есть целых восемь нетривиальных разложений на множители: $100 = 2 \cdot 50$, $100 = 2 \cdot 2 \cdot 25$, $100 = 2 \cdot 2 \cdot 5 \cdot 5$, $100 = 2 \cdot 5 \cdot 10$, $100 = 4 \cdot 25$, $100 = 4 \cdot 5 \cdot 5$, $100 = 5 \cdot 20$ и $100 = 10 \cdot 10$.

Теперь Рома, Сережа и Андрюша думают — сколько же различных нетривиальных разложений на множители есть у числа $n = p^k q^k$?

Формат входного файла

Входной файл содержит число n ($6 \le n \le 10^{18}$, гарантируется, что $n = p^k q^k$ для различных простых p и q и целого $k > 0$).

Формат выходного файла

Выведите в выходной файл одно число — количество нетривиальных разложений на множители числа n.

Пример

false.in	false.out
6	1
100	8

Задача D. Игра на графе

Имя входного файла:	game.in
Имя выходного файла:	game.out
Ограничение по времени:	3 секунды
Ограничение по памяти:	64 мегабайта

Наташа и Петя любят играть в следующую игру на лекциях по теории сложности. Они рисуют неориентированный двудольный граф G на листе бумаги и ставят фишку в одну из его вершин. После этого они делают ходы по очереди, Наташа ходит первой.

Ход в игре заключается в том, что фишка перемещается по графу вдоль одного из ребер. После хода вершина, в которой фишка находилась перед ходом, а также все инцидентные ей ребра, удаляются из графа. Игрок, который не может сделать ход, проигрывает.

Вам задан граф, который нарисовали Наташа и Петя. Для каждой вершины графа определите, кто выиграет при оптимальной игре обоих игроков, если фишка будет исходно размещена в этой вершине.

Формат входного файла

Первая строка входного файла содержит три целых числа: n_1, n_2 и m — количество вершин в первой и второй доле, соответственно, а также количество ребер в графе ($1 \le n_1, n_2 \le 500$, $0 \le m \le 50\,000$). Следующие m строк описывают ребра — каждая строка содержит по два числа — номера вершин, соединенных соответствующим ребром. Вершины в каждой доле независимо пронумерованы, начиная с 1.

Формат выходного файла

Выведите две строки. Первая строка должна содержать n_1 символов, i-й символ должен быть 'N', если при исходом расположении фишки в i-й вершине первой доли, выигрывает Наташа и 'P', если выигрывает Петя. Вторая строка должна описывать вершины второй доли аналогичным образом.

Пример

game.in	game.out
3 3 5 1 1 1 2 1 3 2 1 3 1	NPP NPP

Задача E. Money, Money, Money

Имя входного файла:	`money.in`
Имя выходного файла:	`money.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Правительство Флатландии решило провести реформу денежной системы. Цель реформы — радикально уменьшить число банкнот в обращении, в результате их должно остаться ровно две. Таким образом, после реформу во Флатландии будут циркулировать банкноты достоинством a тупиков[1] и b тупиков, где a и b поручено выбрать министерству финансов.

Одна из проблем заключается в том, что президент Флатландии ненавидит число x. Поэтому министр финансов решил, что выберет такие a и b, что нельзя будет заплатить ровно x тупиков без сдачи. С другой стороны, для любой суммы большей x должна быть возможность заплатить ее без сдачи.

Итак, вам поручено выбрать соответствующие a и b.

Формат входного файла

Входной файл содержит число x ($1 \le x \le 10^{12}$).

Формат выходного файла

Выведите два целых числа a и b, такие что сумму в x тупиков нельзя заплатить банкнотами в a и b тупиков без сдачи, а любую бóльшую сумму — можно.

Если решения не существует, выведите в выходной файл два нуля.

Пример

money.in	money.out
3	2 5
4	0 0
5	3 4

[1]тýпик — денежная единица Флатландии, равная примерно 3.14 Российским рублям

Задача F. Военный полигон

Имя входного файла:	`polygon.in`
Имя выходного файла:	`polygon.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	64 мегабайта

Министерство обороны Флатландии планирует построить новый военный полигон. Полигон должен иметь форму круга.

Поскольку генералы в министерстве волнуются о секретности проводимых на полигоне испытаний, он должен быть надежно защищен. Флатландия защищена сверху несколькими специальными силовыми щитами, каждый из них имеет форму прямоугольника со сторонами, параллельными осям координат. Генералы хотят выбрать такое место для полигона, где он был бы полностью защищен хотя бы двумя конкретными силовыми щитами (недостаточно, чтобы каждая точка просто была защищена хотя бы двумя щитами, должно быть два щита, каждый из которых защищает полигон полностью).

Конечно, генералы хотят построить полигон максимального возможного размера. Помогите им выбрать такое место для полигона, чтобы он имел максимальный возможный радиус.

Формат входного файла

Первая строка входного файла содержит число n — количество силовых щитов. Каждая из следующих n строк описывает силовой щит ($1 \le n \le 200\,000$). Описание представляет собой четверку координат: x_{min}, y_{min}, x_{max}, y_{max}. Все координаты целые и не превышают $100\,000$ по абсолютной величине.

Формат выходного файла

Выведите три вещественных числа — координаты центра полигона и его радиус. Все числа следует выводить ровно с одним знаком после десятичной точки.

Если построить полигон невозможно, выведите “`Impossible`” на первой строке выходного файла.

Пример

polygon.in	**polygon.out**

`4`
`0 0 2 3`
`1 -1 4 1`
`1 1 4 4`
`2 0 5 5`
	`3.0 2.0 1.0`

`1` `0 0 1 1`	`Impossible`

`2`
`0 0 3 3`
`0 0 3 3`
	`1.5 1.5 1.5`

Задача G. Поразрядная сортировка

Имя входного файла:	`radix.in`
Имя выходного файла:	`radix.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Поразрядная сортировка является одним из видов сортировки, которые работают за линейное от размера сортируемого массива время. Такая скорость достигается за счет того, что эта сортировка использует внутреннюю структуру сортируемых объектов.

Изначально этот алгоритм использовался для сортировки перфокарт. Первая его компьютерная реализация была создана в университете MIT Гарольдом Сьюардом (Harold H. Seward).

Опишем алгоритм подробнее. Пусть задан массив строк $s_1, \ldots, s_n$, причем все строки имеют одинаковую длину m. Работа алгоритма состоит из m фаз. На i-ой фазе строки сортируются по i-ой с конца букве.

Происходит это следующим образом. Будем, для простоты, в этой задаче рассматривать строки из цифр от 0 до 9. Для каждой цифры создается «корзина» («bucket»), после чего строки s_i распределяются по «корзинам» в соответствии с i-ой с конца цифрой. Строки, у которых i-ая с конца цифра равна j попадают в j-ую корзину (например, строка 123 на первой фазе попадет в третью корзину, на второй — во вторую, на третьей — в первую). После этого элементы извлекаются из корзин в порядке увеличения номера корзины. Таким образом, после первой фазы строки отсортированы по последней цифре, после двух фаз — по двум последним, ..., после m фаз — по всем.

При важно, чтобы элементы в корзинах сохраняли тот же порядок, что и в исходном массиве (до начала этой фазы). Например, если массив до первой фазы имеет вид: $111, 112, 211, 311$, то элементы по корзинам распределятся следующим образом: в первой корзине будет: $111, 211, 311$, а второй: 112.

Ваша задача состоит в написании программы, детально показывающей работу этого алгоритма на заданном массиве.

Формат входного файла

Первая строка входного файла содержит целое число n ($1 \le n \le 1000$). Последующие n строк содержат каждая по одной строке s_i. Длины всех s_i одинаковы и не превосходят 20. Все s_i состоят только из цифр от 0 до 9.

Формат выходного файла

В выходной файл выведите исходный массив строк s_i, состояние «корзин» после распределения элементов по ним для каждой фазы и отсортированный массив. Следуйте формату, приведенному в примере.

Примеры

radix.in
9 12 32 45 67 98 29 61 35 09
radix.out
Initial array: 12, 32, 45, 67, 98, 29, 61, 35, 09 ********** Phase 1 Bucket 0: empty Bucket 1: 61 Bucket 2: 12, 32 Bucket 3: empty Bucket 4: empty Bucket 5: 45, 35 Bucket 6: empty Bucket 7: 67 Bucket 8: 98 Bucket 9: 29, 09 ********** Phase 2 Bucket 0: 09 Bucket 1: 12 Bucket 2: 29 Bucket 3: 32, 35 Bucket 4: 45 Bucket 5: empty Bucket 6: 61, 67 Bucket 7: empty Bucket 8: empty Bucket 9: 98 ********** Sorted array: 09, 12, 29, 32, 35, 45, 61, 67, 98

Задача H. 2-3 Дерево

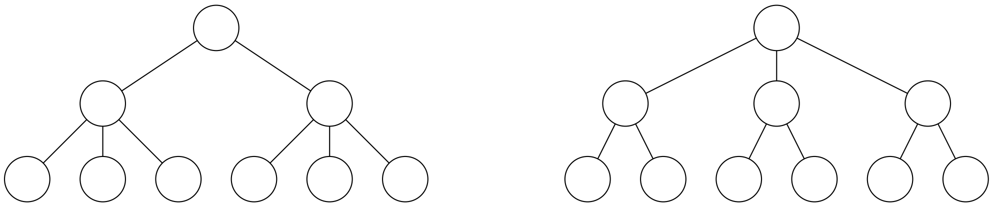
Имя входного файла:	`twothree.in`
Имя выходного файла:	`twothree.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

2-3 дерево — элегантная структура данных, изобретенная Джоном Хопкрофтом. Она предназначена для использования с той же целью, что и двоичное дерево поиска. 2-3 дерево представляет собой дерево с корнем, которое обладает следующими свойствами:

- корень и каждая внутренняя вершина имеет либо 2 либо 3 ребенка;
- глубина всех листьев одна и та же.

Единственное исключение — это когда дерево содержит ровно одну вершину. В этом случае корень дерева является и листом, и поэтому не имеет детей. Основная суть приведенных свойств в том, что дерево с l листьями имеет высоту $O(\log l)$.

Вообще говоря, может существовать несколько 2-3 деревьев с l листьями. Например, на следующем рисунке показаны два возможных дерева с 6 листьями.



По заданному числу l найдите количество различных 2-3 деревьев с l листьями. Так как ответ может быть довольно большим, выведите его по модулю r.

Формат входного файла

Входной файл содержит два целых числа: l и r ($1 \le l \le 5\,000$, $1 \le r \le 10^9$).

Формат выходного файла

Выведите одно число — количество различных 2-3 деревьев, имеющих ровно l листьев, взятое по модулю r.

Пример

twothree.in	twothree.out
6 1000000000	2
7 1000000000	3

Задача I. Различные слова

Имя входного файла:	`words.in`
Имя выходного файла:	`words.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	128 мегабайт

Дана строка S состоящая из N символов. Назовем ее подстрокой S_{ij} строку с i-го по j-й символ ($i \leq j$). Ваша задача — посчитать количество различных подстрок заданной строки.

Формат входного файла

Во входном файле находится одна непустая строка, состоящая из маленьких латинских букв, длиной не более чем 1024 символа.

Формат выходного файла

Выходной файл должен содержать одно число — количество различных подстрок строки S.

Примеры

words.in	**words.out**
abc	6
aaa	3