У некоторых задач есть написанные в скобках алгоритмы, это означает, что решать их нужно именно так, как сказано.

Задача A. Сортировка различных чисел

Имя входного файла:	`sortuniq.in`
Имя выходного файла:	`sortuniq.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 Мебибайт

В этой задаче нужно отсортировать последовательность натуральных чисел, заданных во входном файле, при условии, что числа в последовательности не повторяются, и максимальное из этих чисел не превосходит длины последовательности.

Формат входных данных

В первой строке входного файла задано натуральное число n ($1 \leqslant n \leqslant 1\,000\,000$). Во второй строке заданы через пробел n натуральных чисел a_1, a_2, …, a_n ($1 \leqslant a_i \leqslant n$).

Формат выходных данных

В первой строке выходного файла выведите через пробел n чисел — числа a_i в неубывающем порядке.

Пример

sortuniq.in	sortuniq.out
2 2 1	1 2

Задача B. Сортировка большой последовательности (HeapSort)

Имя входного файла:	`sort.in`
Имя выходного файла:	`sort.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 Мебибайт

В этой задаче нужно отсортировать числа, заданные во входном файле.

Формат входных данных

В первой строке входного файла задано целое число n ($1 \leqslant n \leqslant 300\,000$). Во второй строке заданы через пробел n целых чисел a_1, a_2, …, a_n ($1 \leqslant a_i \leqslant 10^9$).

Формат выходных данных

В первой строке выходного файла выведите через пробел n чисел — числа a_i в неубывающем порядке.

Примеры

sort.in	sort.out
3 1 2 3	1 2 3
4 3 2 2 1	1 2 2 3
5 10 100 10 1000 10	10 10 10 100 1000

Задача C. Сумма (Дерево отрезков)

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

Дан массив из N элементов, нужно научиться находить сумму чисел на отрезке.

Формат входных данных

Первая строка входного файла содержит два целых числа N и K — число чисел в массиве и количество запросов. $(1 \leqslant N \leqslant 100\,000)$, $(0 \leqslant K \leqslant 100\,000)$. Следующие K строк содержат запросы

1. A i x — присвоить i-му элементу массива значение x $(1 \leqslant i \leqslant n,\ 0 \leqslant x \leqslant 10^9)$
2. Q l r — найти сумму чисел в массиве на позициях от l до r. $(1 \leqslant l \leqslant r \leqslant n)$

Изначально в массиве живут нули.

Формат выходных данных

На каждый запрос вида Q l r нужно вывести единственное число — сумму на отрезке.

Примеры

sum.in	sum.out
5 9 A 2 2 A 3 1 A 4 2 Q 1 1 Q 2 2 Q 3 3 Q 4 4 Q 5 5 Q 1 5	0 2 1 2 0 5

Задача D. Сумма (Дерево Фенвика)

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

Дан массив из N элементов, нужно научиться находить сумму чисел на отрезке.

Формат входных данных

Первая строка входного файла содержит два целых числа N и K — число чисел в массиве и количество запросов. $(1 \leqslant N \leqslant 100\,000)$, $(0 \leqslant K \leqslant 100\,000)$. Следующие K строк содержат запросы

1. A i x — присвоить i-му элементу массива значение x $(1 \leqslant i \leqslant n,\ 0 \leqslant x \leqslant 10^9)$
2. Q l r — найти сумму чисел в массиве на позициях от l до r. $(1 \leqslant l \leqslant r \leqslant n)$

Изначально в массиве живут нули.

Формат выходных данных

На каждый запрос вида Q l r нужно вывести единственное число — сумму на отрезке.

Примеры

sum.in	sum.out
5 9 A 2 2 A 3 1 A 4 2 Q 1 1 Q 2 2 Q 3 3 Q 4 4 Q 5 5 Q 1 5	0 2 1 2 0 5

Задача E. Среднее на отрезках

Имя входного файла:	`middle.in`
Имя выходного файла:	`middle.out`
Ограничение по времени:	0.5 секунд
Ограничение по памяти:	64 мегабайта

Дана последовательность из N чисел и M запросов. Каждый запрос представляет собой пару чисел i, j. После каждого запроса необходимо заменить все элементы последовательности от i до j на их среднее арифметическое. Если сумма элементов текущей последовательности меньше либо равна сумме элементов первоначальной последовательности, среднее округляется вверх, иначе — вниз. Ваша задача — найти последовательность после проведенных преобразований.

Формат входных данных

Первая строка входных данных содержит два натуральных числа N и M ($1 \leqslant M, N \leqslant 30\,000$). Во второй строке записаны N чисел, каждое из которых не превосходит 10^9 по модулю — элементы последовательности. Затем идут M строчек по два числа в каждой — запросы, которые необходимо выполнить.

Формат выходных данных

Выведите N чисел — элементы результирующей последовательности.

Примеры

middle.in	middle.out
6 4 1 2 3 4 5 6 1 2 2 5 5 6 4 6	2 3 3 5 5 5

Задача F. Художник

Имя входного файла:	`painter.in`
Имя выходного файла:	`painter.out`
Ограничение по времени:	4 секунды
Ограничение по памяти:	256 мебибайт

Итальянский художник-абстракционист Ф. Мандарино увлекся рисованием одномерных черно-белых картин. Он пытается найти оптимальное местоположение и количество черных участков картины. Для этого он проводит на прямой белые и черные отрезки, и после каждой из таких операций хочет знать количество черных отрезков на получившейся картине и их суммарную длину.

Изначально прямая — белая. Ваша задача — написать программу, которая после каждой из таких операций выводит в выходной файл интересующие художника данные.

Формат входных данных

В первой строке входного файла содержится общее количество нарисованных отрезков ($1 \leqslant N \leqslant 100\,000$). В последующих N строках содержится описание операций. Каждая операция описывается строкой вида $c\ x\ l$, где c — цвет отрезка (`W` для белых отрезков, `B` для черных), а сам отрезок имеет вид $[x; x+l]$, причем координаты обоих концов — целые числа, не превосходящие по модулю 500 000. Длина задается положительным целым числом.

Формат выходных данных

После выполнения каждой из операций необходимо вывести в выходной файл на отдельной строке количество черных отрезков на картине и их суммарную длину, разделенные одним пробелом.

Пример

painter.in	painter.out
7 W 2 3 B 2 2 B 4 2 B 3 2 B 7 2 W 3 1 W 0 10	0 0 1 2 1 4 1 4 2 6 3 5 0 0

Задача G. K-инверсии

Имя входного файла:	`kinverse.in`
Имя выходного файла:	`kinverse.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Рассмотрим перестановку $a_1, a_2, \ldots, a_n$ (здесь a_i — различные целые числа в пределах от 1 до n включительно). Назовём *k-инверсией* последовательность чисел $i_1, i_2, \ldots, i_k$ такую, что $1 \leqslant i_1 < i_2 < \ldots < i_k \leqslant n$ и $a_{i_1} > a_{i_2} > \ldots > a_{i_k}$. Ваша задача — найти количество различных k-инверсий в заданной перестановке.

Формат входных данных

Первая строка ввода содержит два целых числа n и k ($1 \leqslant n \leqslant 20\,000$, $2 \leqslant k \leqslant 10$). Во второй строке заданы через пробел n чисел a_i.

Формат выходных данных

Выведите одно число — количество k-инверсий в заданной перестановке. Поскольку это число может быть очень велико, выведите его по модулю 10^9.

Примеры

kinverse.in	kinverse.out
3 2 3 1 2	2
5 3 5 4 3 2 1	10

Задача H. Переворачивания

Имя входного файла:	`reverse.in`
Имя выходного файла:	`reverse.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Учитель физкультуры школы с углубленным изучением предметов уже давно научился считать суммарный рост всех учеников, находящихся в ряду на позициях от l до r. Но дети играют с ним злую шутку. В некоторый момент дети на позициях с l по r меняются местами. Учитель заметил, что у детей не очень богатая фантазия, поэтому они всегда «переворачивают» этот отрезок, т. е. l меняется с r, $l+1$ меняется с $r-1$ и так далее. Но учитель решил не ругать детей за их хулиганство, а все равно посчитать суммарный рост на всех запланированных отрезках.

Формат входных данных

В первой строке записано два числа n и m ($1 \leqslant n, m \leqslant 200\,000$) — количество детей в ряду и количество событий, произошедших за все время. Во второй строке задано n натуральных чисел — рост каждого школьника в порядке следования в ряду. Рост детей не превосходит $2 \cdot 10^5$. Далее в m строках задано описание событий: три числа q, l, r в каждой строке ($0 \leqslant q \leqslant 1$, $1 \leqslant l \leqslant r \leqslant n$). Число q показывает тип события: 0 показывает необходимость посчитать и вывести суммарный рост школьников на отрезке $[l, r]$; 1 показывает то, что дети на отрезке $[l, r]$ «перевернули» свой отрезок. Все числа во входном файле целые.

Формат выходных данных

Для каждого события типа 0 выведите единственное число на отдельной строке — ответ на этот запрос.

Пример

reverse.in	reverse.out
5 6 1 2 3 4 5 0 1 5 0 2 4 1 2 4 0 1 3 0 4 5 0 3 5	15 9 8 7 10

Задача I. Компоненты связности

Имя входного файла:	`connect.in`
Имя выходного файла:	`connect.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Вам задан неориентированный граф с N вершинами и M ребрами ($1 \leqslant N \leqslant 20\,000$, $1 \leqslant M \leqslant 200\,000$). В графе отсутствуют петли и кратные ребра.

Определите компоненты связности заданного графа.

Формат входных данных

Граф задан во входном файле следующим образом: первая строка содержит числа N и M. Каждая из следующих M строк содержит описание ребра — два целых числа из диапазона от 1 до N — номера концов ребра.

Формат выходных данных

На первой строке выходного файла выведите число L — количество компонент связности заданного графа. На следующей строке выведите N чисел из диапазона от 1 до L — номера компонент связности, которым принадлежат соответствующие вершины. Компоненты связности следует занумеровать от 1 до L произвольным образом.

Пример

connect.in	connect.out
4 2 1 2 3 4	2 1 1 2 2

Задача J. Топологическая сортировка

Имя входного файла:	`topsort.in`
Имя выходного файла:	`topsort.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Дан ориентированный невзвешенный граф. Необходимо его топологически отсортировать.

Формат входных данных

В первой строке входного файла даны два натуральных числа N и M ($1 \leqslant N \leqslant 100\,000, M \leqslant 100\,000$) — количество вершин и рёбер в графе соответственно. Далее в M строках перечислены рёбра графа. Каждое ребро задаётся парой чисел — номерами начальной и конечной вершин соответственно.

Формат выходных данных

Вывести любую топологическую сортировку графа в виде последовательности номеров вершин. Если граф невозможно топологически отсортировать, вывести -1.

Пример

topsort.in	topsort.out
6 6 1 2 3 2 4 2 2 5 6 5 4 6	4 6 3 1 2 5
3 3 1 2 2 3 3 1	-1

Задача K. Поиск цикла

Имя входного файла:	`cycle.in`
Имя выходного файла:	`cycle.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Дан ориентированный невзвешенный граф. Необходимо определить есть ли в нём циклы, и если есть, то вывести любой из них.

Формат входных данных

В первой строке входного файла находятся два натуральных числа N и M ($1 \leqslant N \leqslant 100\,000$, $M \leqslant 100\,000$) — количество вершин и рёбер в графе соответственно. Далее в M строках перечислены рёбра графа. Каждое ребро задаётся парой чисел — номерами начальной и конечной вершин соответственно.

Формат выходных данных

Если в графе нет цикла, то вывести «`NO`», иначе — «`YES`» и затем перечислить все вершины в порядке обхода цикла.

Примеры

cycle.in	cycle.out
2 2 1 2 2 1	YES 1 2
2 2 1 2 1 2	NO

Задача L. Точки сочленения

Имя входного файла:	`points.in`
Имя выходного файла:	`points.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Дан неориентированный граф. Требуется найти все точки сочленения в нем.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количество вершин и ребер графа соответственно ($n \leqslant 20\,000$, $m \leqslant 200\,000$).

Следующие m строк содержат описание ребер по одному на строке. Ребро номер i описывается двумя натуральными числами b_i, e_i — номерами концов ребра ($1 \leqslant b_i, e_i \leqslant n$).

Формат выходных данных

Первая строка выходного файла должна содержать одно натуральное число b — количество точек сочленения в заданном графе. На следующей строке выведите b целых чисел — номера вершин, которые являются точками сочленения, в возрастающем порядке.

Пример

points.in	points.out
9 12 1 2 2 3 4 5 2 6 2 7 8 9 1 3 1 4 1 5 6 7 3 8 3 9	3 1 2 3

Задача M. Конденсация графа

Имя входного файла:	`condense2.in`
Имя выходного файла:	`condense2.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Требуется найти количество ребер в конденсации ориентированного графа. Примечание: конденсация графа не содержит кратных ребер.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количество вершин и ребер графа соответственно ($n \leqslant 10\,000$, $m \leqslant 100\,000$). Следующие m строк содержат описание ребер, по одному на строке. Ребро номер i описывается двумя натуральными числами b_i, e_i — началом и концом ребра соответственно ($1 \leqslant b_i, e_i \leqslant n$). В графе могут присутствовать кратные ребра и петли.

Формат выходных данных

Первая строка выходного файла должна содержать одно число — количество ребер в конденсации графа.

Пример

condense2.in	condense2.out
4 4 2 1 3 2 2 3 4 3	2

Задача N. Предок

Имя входного файла:	`ancestor.in`
Имя выходного файла:	`ancestor.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Напишите программу, которая для двух вершин дерева определяет, является ли одна из них предком другой.

Формат входных данных

Первая строка входного файла содержит натуральное число n ($1 \leqslant n \leqslant 100\,000$) — количество вершин в дереве. Во второй строке находится n чисел. При этом i-ое число второй строки определяет непосредственного родителя вершины с номером i. Если номер родителя равен нулю, то вершина является корнем дерева.

В третьей строке находится число m ($1 \leqslant m \leqslant 100\,000$) — количество запросов. Каждая из следующих m строк содержит два различных числа a и b ($1 \leqslant a, b \leqslant n$).

Формат выходных данных

Для каждого из m запросов выведите на отдельной строке число `1`, если вершина a является одним из предков вершины b, и `0` в противном случае.

Пример

ancestor.in	ancestor.out
6 0 1 1 2 3 3 5 4 1 1 4 3 6 2 6 6 5	0 1 1 0 0

Задача O. Chip Installation

Имя входного файла:	`chip.in`
Имя выходного файла:	`chip.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Новый ЧИП скоро установят в новый летательный апарат, недавно выпущенной компанией `Airtram`. ЧИП имеет форму диска. Есть n проводов, которые нужно подсоединить к ЧИПу.

Каждый провод можно подсоединить в один из двух разъемов, допустимых для этого провода. Все $2n$ разъемов расположены на границе диска. По кругу. Каждый провод имеет свой цвет. Для повышения безопасности два провода одного цвета не могут быть подсоединены к соседним разъемам.

Дана конфигурация разъемов на ЧИПе, найдите способ подсоединить все провода, не нарушающий условия про цвета.

Формат входных данных

Первая строка содержит число n — количество проводов ($1 \leqslant n \leqslant 50\,000$). Вторая строка содержит n целых чисел от 1 до 10^9 — цвета проводов. Третья строка содержит $2n$ целых чисел от 1 до n описывающих разъемы. Число обозначает номер провода, который может быть подсоединен к данному разъему. Каждое число от 1 до n встречается ровно дважды. Разъемы перечислены порядке "по кругу". 1-й разъем является соседним со 2-м и так далее, не забудьте, что n-й является соседним с 1-м.

Формат выходных данных

Если не существует способа подключить все провода, выведите одно слово “`NO`”.

Иначе выведите “`YES`” и n целых чисел. Для каждого провода выведите номер разъема, к которому нужно подключить этот провод. Разъемы нумеруются числами от 1 до $2n$ в том порядке, в котором они даны во входном файле.

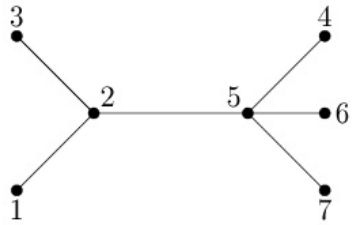
Примеры

chip.in	chip.out
2 1 1 1 1 2 2	YES 1 3
2 1 1 1 2 1 2	NO
2 1 2 1 2 1 2	YES 1 2

Задача P. Autotourism

Имя входного файла:	`autotourism.in`
Имя выходного файла:	`autotourism.out`
Ограничение по времени:	2 с
Ограничение по памяти:	256 Мб

В Байтландии существуют n городов, соединённых n - 1 дорогами с двусторонним движением таким образом, что из каждого города можно проехать в любой другой по сети дорог. Длина каждой дороги равна 1 километру.



Бензобак автомобиля позволяет проехать без заправки m километров. Требуется выбрать маршрут, позволяющий посетить наибольшее количество различных городов без дозаправки. При этом начинать и заканчивать маршрут можно в произвольных городах.

Формат входных данных

В первой строке входного файла заданы два целых числа n и m ($2 \leqslant n \leqslant 500\,000$, $1 \leqslant m \leqslant 200\,000\,000$) — количество городов в стране и количество километров, которое автомобиль может проехать без дозаправки. В последующих n - 1 строках описаны дороги. Каждая дорога задаётся двумя целыми числами a и b ($1 \leqslant a, b \leqslant n$) — номерами городов, которые она соединяет. Длина каждой дороги равна 1 км.

Формат выходных данных

Выведите одно число — максимальное количество городов, которое можно посетить без дозаправки.

Пример

autotourism.in	autotourism.out
7 6 1 2 2 3 2 5 5 6 5 7 5 4	5

Пояснение к примеру

5 городов можно посетить, например, по схеме $4 \to 5 \to 7 \to 5 \to 6 \to 5 \to 2$ или по схеме $3 \to 2 \to 1 \to 2 \to 5 \to 6 \to 5$.

Задача Q. Длиннейший путь

Имя входного файла:	`longpath.in`
Имя выходного файла:	`longpath.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Дан ориентированный граф без циклов. Требуется найти в нем длиннейший путь.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количество вершин и дуг графа соответственно. Следующие m строк содержат описания дуг по одной на строке. Ребро номер i описывается двумя натуральными числами b_i и e_i — началом и концом дуги соответственно ($1 \leqslant b_i, e_i \leqslant n$).

Входной граф не содержит циклов и петель.

$n \leqslant 10\,000$, $m \leqslant 100\,000$.

Формат выходных данных

Первая строка выходного файла должна содержать одно натуральное число — количество дуг в длиннейшем пути.

Пример

longpath.in	longpath.out
5 5 1 2 2 3 3 4 3 5 1 5	3

Задача R. Кратчайший путь двух коней (BFS)

Имя входного файла:	`knight2.in`
Имя выходного файла:	`knight2.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мегабайта

Переведите каждого из двух коней из одной клетки в другую за наименьшее общее число ходов. Два коня не могут одновременно находиться в одной клетке.

Формат входных данных

Во входном файле записаны координаты первого и второго коня, затем координаты клеток, куда нужно их переместить.

Формат выходных данных

Программа должна вывести последовательность ходов коней в виде нескольких строк. Первым символом в строке должен быть номер коня (1 или 2), затем, через пробел, координаты клетки, в которую он переставляется. Необходимо вывести любое из возможных оптимальных решений.

Пример

knight2.in	knight2.out
a1 c2 c2 a1	1 b3 1 d4 2 a1 1 c2

Задача S. Островные государства (BFS 0-2)

Имя входного файла:	`island2.in`
Имя выходного файла:	`island2.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Суровые феодальные времена переживала некогда великая островная страна Байтландия. За главенство над всем островом борются два самых сильных барона. Таким образом, каждый город страны контролируется одним из правителей. Как водится издревле, некоторые из городов соединены двусторонними дорогами. Бароны очень не любят друг друга и стараются делать как можно больше пакостей. В частности, теперь для того чтобы пройти по дороге, соединяющей города различных правителей, надо заплатить пошлину — один байтландский рубль. Кроме этого за выезд из городов с четными номерами береться удвоенная пошлина.

Программист Вася живет в городе номер 1. С наступлением лета он собирается съездить в город N на Всебайтландское сборище программистов. Разумеется, он хочет затратить при этом как можно меньше денег и помочь ему здесь, как обычно, предлагается Вам.

Формат входных данных

В первой строке входного файла записано два числа N и M ($1 \leqslant N, M \leqslant 100\,000$) — количество городов и количество дорог соответсвенно.

В следующий строке содержится информация о городах — N чисел 1 или 2 — какому из баронов принадлежит соответствующий город.

В последних M строках записаны пары $1 \leqslant a, b \leqslant N$, $a \neq b$. Каждая пара означает наличие дороги из города a в город b. По дорогам Байтландии можно двигаться в любом направлении.

Формат выходных данных

Если искомого пути не существует, выведите единственное слово `impossible`. В противном случае в первой строке напишите минимальную стоимость и количество посещенных городов, а во вторую выведите эти города в порядке посещения. Если минимальных путей несколько, выведите любой.

Пример

island2.in	island2.out
7 8 1 1 1 1 2 2 1 1 2 2 5 2 3 5 4 4 3 4 7 1 6 6 7	0 5 1 2 3 4 7
5 5 1 1 1 2 1 1 2 2 3 3 4 4 5 2 4	3 5 1 2 3 4 5

Задача T. Расстояние между вершинами (Dijkstra)

Имя входного файла:	`distance.in`
Имя выходного файла:	`distance.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Дан неориентированный взвешенный граф. Найти вес минимального пути между двумя вершинами.

Формат входных данных

Первая строка входного файла содержит натуральные числа N, M, вторая строка содержит натуральные числа S и F ($N \leqslant 5\,000$, $M \leqslant 100\,000$, $1 \leqslant S, F \leqslant N$, $S \neq F$) — количество вершин и ребер графа а также номера вершин, длину пути между которыми требуется найти.

Следующие M строк по три натуральных числа b_i, e_i и w_i — номера концов i-ого ребра и его вес соответственно ($1 \leqslant b_i, e_i \leqslant n$, $0 \leqslant w_i \leqslant 100\,000$).

Формат выходных данных

Первая строка должна содержать одно натуральное число — вес минимального пути между вершинами S и F. Во второй строке через пробел выведите вершины на кратчайшем пути из S в F в порядке обхода.

Если путь из S в F не существует, выведите -1.

Пример

distance.in	distance.out
4 4 1 3 1 2 1 3 4 5 3 2 2 4 1 4	3 1 2 3

Задача U. Лабиринт знаний

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

Участникам сборов подарили билеты на аттракцион “Лабиринт знаний”. Лабиринт представляет собой N комнат, занумерованных от 1 до N, между некоторыми из которых есть двери. Когда человек проходит через дверь, показатель его знаний изменяется на определенную величину, фиксированную для данной двери. Вход в лабиринт находится в комнате 1, выход — в комнате N. Каждый участник сборов проходит лабиринт ровно один раз и наибрает некоторое количество знаний (при входе в лабиринт этот показатель равен нулю). Ваша задача — показать наилучший результат.

Формат входных данных

Первая строка входного файла содержит целые числа N ($1 \leqslant N \leqslant 2\,000$) — количество комнат и M ($1 \leqslant M \leqslant 10\,000$) — количество дверей. В каждой из следующих M строк содержится описание двери — номера комнат, из которой она ведет и в которую она ведет (через дверь в лабиринте можно ходить только в одну сторону), а также целое число, которое прибавляется к количеству знаний при прохождении через дверь (это число по модулю не превышает 10 000). Двери могут вести из комнаты в нее саму, между двумя комнатами может быть более одной двери.

Формат выходных данных

В выходной файл выведите “`:)`” — если можно пройти лабиринт и получить неограниченно большой запас знаний, “`:(`” — если лабиринт пройти нельзя, и максимальное количество набранных знаний в противном случае.

Пример

maze.in	maze.out
2 2 1 2 5 1 2 -5	5

Задача V. Pink Floyd

Имя входного файла:	`pinkfloyd.in`
Имя выходного файла:	`pinkfloyd.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Группа *Pink Floyd* собирается отправиться в новый концертный тур по всему миру. По предыдущему опыту группа знает, что солист *Роджер Уотерс* постоянно нервничает при перелетах. На некоторых маршрутах он теряет вес от волнения, а на других — много ест и набирает вес.

Известно, что чем больше весит Роджер, тем лучше выступает группа, поэтому требуется спланировать перелеты так, чтобы вес Роджера на каждом концерте был максимально возможным.

Группа должна посещать города в том же порядке, в котором она дает концерты. При этом между концертами группа может посещать промежуточные города.

Формат входных данных

Первая строка входного файла содержит три натуральных числа n, m и k — количество городов в мире, количество рейсов и количество концертов, которые должна дать группа соответственно ($n \leqslant 100$, $m \leqslant 10\,000$, $2 \leqslant k \leqslant 10\,000$). Города пронумерованы числами от 1 до n.

Следующие m строк содержат описание рейсов, по одному на строке. Рейс номер i описывается тремя числами b_i, e_i и w_i — номер начального и конечного города рейса и предполагаемое изменение веса Роджера в миллиграммах ($1 \leqslant b_i, e_i \leqslant n$, $-100\,000 \leqslant w_i \leqslant 100\,000$).

Последняя строка содержит числа $a_1, a_2, ..., a_k$ — номера городов, в которых проводятся концерты ($a_i \neq a_{i+1}$). В начале концертного тура группа находится в городе a_1.

Гарантируется, что группа может дать все концерты.

Формат выходных данных

Первая строка выходного файла должна содержать число l — количество рейсов, которые должна сделать группа. Вторая строка должна содержать l чисел — номера используемых рейсов.

Если существует такая последовательность маршрутов между концертами, что Роджер будет набирать вес неограниченно, то первая строка выходного файла должна содержать строку «`infinitely kind`».

Примеры

pinkfloyd.in	pinkfloyd.out
4 8 5 1 2 -2 2 3 3 3 4 -5 4 1 3 1 3 2 3 1 -2 3 2 -3 2 4 -10 1 3 1 2 4	6 5 6 5 7 2 3
4 8 5 1 2 -2 2 3 3 3 4 -5 4 1 3 1 3 2 3 1 -2 3 2 -3 2 4 10 1 3 1 2 4	infinitely kind

Задача W. День Объединения (Прима)

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

В Байтландии есть целых n городов, но нет ни одной дороги. Король решил исправить эту ситуацию и соединить некоторые города дорогами так, чтобы по этим дорогам можно было бы добраться от любого города до любого другого. Когда строительство будет завершено, Король планирует отпраздновать День Объединения. К сожалению, казна Байтландии почти пуста, поэтому Король требует сэкономить деньги, минимизировав суммарную длину всех построенных дорог.

Формат входных данных

Первая строка входного файла содержит натуральное число n ($1 \leqslant n \leqslant 5\,000$) — количество городов в Байтландии. Каждая из следующих n строк содержит два целых числа x_i, y_i — координаты i-го города ($-10\,000 \leqslant x_i, y_i \leqslant 10\,000$). Никакие два города не расположены в одной точке.

Формат выходных данных

Первая строка выходного файла должна содержать минимальную суммарную длину дорог. Выведите число с точностью не менее 10^{-3}.

Примеры

unionday.in	unionday.out
6 1 1 7 1 2 2 6 2 1 3 7 3	9.65685

Задача X. Остовное дерево 2 (Краскал)

Имя входного файла:	`spantree2.in`
Имя выходного файла:	`spantree2.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Требуется найти в связном графе остовное дерево минимального веса.

Формат входных данных

Первая строка входного файла содержит два натуральных числа n и m — количество вершин и ребер графа соответственно. Следующие m строк содержат описание ребер по одному на строке. Ребро номер i описывается тремя натуральными числами b_i, e_i и w_i — номера концов ребра и его вес соответственно ($1 \leqslant b_i, e_i \leqslant n$, $0 \leqslant w_i \leqslant 100\,000$). $n \leqslant 20\,000, m \leqslant 100\,000$.

Граф является связным.

Формат выходных данных

Первая строка выходного файла должна содержать одно натуральное число — вес минимального остовного дерева.

Примеры

spantree2.in	spantree2.out
4 4 1 2 1 2 3 2 3 4 5 4 1 4	7

Задача ZA. Билеты в кино

Имя входного файла:	`tickets.in`
Имя выходного файла:	`tickets.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Петя и Миша решили пойти в кино. Но, чтобы пойти в кино, надо купить билеты, а два билета стоят n рублей. Ребята решили поделить между собой расходы на билеты, сыграв в следующую игру.

У Пети и Миши есть монеты k типов, достоинством r_1, r_2, ..., r_k рублей. Они по очереди кладут по одной монете любого типа на стол; начинает Петя. Как только на столе оказывается сумма, достаточная для приобретения билетов, тот из ребят, чья очередь класть монету сейчас наступила, объявляется проигравшим и должен дополнительно купить другому мороженое. Количество монет каждого типа у обоих игроков будем считать неограниченным.

Кто выиграет при правильной игре?

Формат входных данных

В первой строке входного файла заданы через пробел два целых числа n и k — требуемая сумма и количество типов монет, соответственно ($0 \leqslant n \leqslant 10\,000$, $1 \leqslant k \leqslant 10$). Во второй строке заданы k целых чисел r_1, r_2, ..., r_k через пробел — достоинства монет ($1 \leqslant r_i \leqslant 100$).

Формат выходных данных

В первой строке выходного файла выведите «`PETYA`», если при правильной игре мороженое достанется Пете, и «`MISHA`», если Мише.

Примеры

tickets.in	tickets.out
10 3 1 2 5	PETYA
2 1 1	MISHA
21 7 6 5 4 3 3 2 1	MISHA

Задача ZB. Игра в разделение

Имя входного файла:	`divgame.in`
Имя выходного файла:	`divgame.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Есть клетчатый прямоугольник размера $w \times h$ клеток. Алиса и Боб делают ходы по очереди; начинает Алиса. Каждый ход состоит из двух частей. Сначала один из имеющихся прямоугольников делится на две непустые части вертикальной или горизонтальной линией, проходящей по линиям сетки. Затем один из двух получившихся прямоугольников также делится на две непустые части линией, также проходящей по линиям сетки и перпендикулярной первой линии.

Проигрывает тот, кто не может сделать ход. Кто выигрывает при правильной игре? Если выигрывает Алиса, какой её ход приводит к выигрышу при дальнейшей правильной игре?

Формат входных данных

В первой строке ввода заданы через пробел два целых числа w и h — ширина и высота исходного прямоугольника ($1 \leqslant w, h \leqslant 100$).

Формат выходных данных

В первой строке выведите имя победителя: «`Alice`», если при правильной игре победит Алиса, и «`Bob`», если победу одержит Боб. Если победит Алиса, в следующих двух строках выведите ход Алисы, ведущий к дальнейшему выигрышу при правильной игре. В первой из этих строк выведите через пробел координаты концов первой линии, которую должна провести Алиса, а во второй — координаты концов второй линии. Для каждого конца линии сначала выводите координату по оси Ox, а затем — координату по оси Oy. Считайте, что левый нижний угол исходного прямоугольника имеет координаты $(0, 0)$, а правый верхний угол — координаты (w, h).

Концы каждой линии можно выводить в любом порядке. Если правильных ходов несколько, можно вывести любой из них.

Примеры

divgame.in	divgame.out
4 2	Alice 0 1 4 1 3 0 3 1
3 3	Bob

Пояснения к примерам

В первом примере Алиса любым возможным ходом разрезает исходный прямоугольник на три такие части, что в них невозможно сделать ещё один ход.

Во втором примере любой ход Алисы оставляет для дальнейших ходов ровно один квадрат размера 2×2 клетки. После этого Боб делит этот квадрат на три непустые части и заканчивает игру.

Задача ZC. Произведение графов

Имя входного файла:	`graphprod.in`
Имя выходного файла:	`graphprod.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Пусть дан ориентированный ациклический граф. Стандартная игра на графе заключается в следующем: изначально на одной из вершин графа (называемой начальной позицией) стоит фишка. Двое игроков по очереди двигают её по рёбрам. Проигрывает тот, кто не может сделать ход.

В теории игр часто рассматриваются более сложные игры. Например, прямое произведение двух игр на графах. Прямое произведение игр — это следующая игра: изначально на каждом графе в начальной позиции стоит по фишке. За ход игрок двигает обе фишки по рёбрам (каждую фишку двигает в собственном графе). Проигрывает тот, кто не может сделать ход. То есть тот, кто не может сделать ход хотя бы в одной игре.

Ваша задача — опеределить, кто выиграет при правильной игре.

Формат входных данных

На первой строке будут даны числа N_1 и M_1 — количество вершин и рёбер в первом графе ($1 \leqslant N_1, M_1 \leqslant 100\,000$). На следующих M_1 строках содержится по два числа x и y ($1 \leqslant x, y \leqslant N_1$).

В следующих $M_2 + 1$ строках задан второй граф в том же формате.

Заканчивается входной файл списком пар начальных вершин, для которых нужно решить задачу. На первой строке задано число T ($1 \leqslant T \leqslant 100\,000$) — количество пар начальных вершин. В следующих T строках указаны пары вершин v_1 и v_2 ($1 \leqslant v_1 \leqslant N_1$, $1 \leqslant v_2 \leqslant N_2$).

Учтите, что в графах могут быть кратные рёбра.

Формат выходных данных

На каждую из T пар начальных вершин выведите строку “`first`”, если при правильной игре выиграет первый, и “`second`”, если второй.

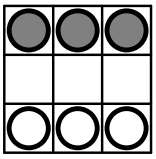
Пример

graphprod.in	graphprod.out
3 2 1 2 2 3 2 1 1 2 2 2 1 3 2	first second

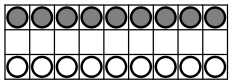
Задача ZD. Игра в пешки

Имя входного файла:	`pawns.in`
Имя выходного файла:	`pawns.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

В свободное время Дед Мороз Петрович и Дед Мороз Егорыч играют в следующую игру. На доске 3×3 пешки расставляются следующим образом:



Пешки ходят и бьют по обычным шахматным правилам, к которым добавляется ещё одно: бить обязательно. Проигрывает тот, кто не может сделать ход. Первыми ходят белые. В течение 100 лет Петрович играл белыми и всегда выигрывал. Однажды Егорычу это надоело, и он принёс доску 3×5. Но и теперь он, играя чёрными, до сих пор проигрывает. «Как так?» — подумал Егорыч и решил купить доску $3 \times N$:



Вот тут Петровичу надо подумать, какими играть, чтобы выиграть. Вам нужно помочь ему это сделать.

Формат входных данных

Целое число N ($1 \leqslant N \leqslant 10^9$).

Формат выходных данных

«`White`», если Петровичу надо играть белыми, и «`Black`», если надо играть чёрными. Егорыч и Петрович каждый раз ходят по наилучшей для себя стратегии.

Примеры

pawns.in	pawns.out
3	White
4	Black
5	White

Задача ZE. Вариация Нима

Имя входного файла:	`varnim.in`
Имя выходного файла:	`varnim.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

На столе лежат n кучек камней: a_1 камней в первой кучке, a_2 камней во второй, ..., a_n в n-ой. Двое играют в игру, делая ходы по очереди. За один ход игрок может либо взять произвольное ненулевое количество камней (возможно, все) из одной любой кучки, либо произвольным образом разделить любую существующую кучку, в которой не меньше двух камней, на две непустые кучки. Проигрывает тот, кто не может сделать ход. Кто выигрывает при правильной игре?

Формат входных данных

В первой строке задано целое число t — количество тестов ($1 \leqslant t \leqslant 100$). Следующие t строк содержат сами тесты. Каждая из них начинается с целого числа n — количества кучек ($1 \leqslant n \leqslant 100$). Далее следует n целых чисел a_1, a_2, ..., a_n через пробел — количество камней в кучках ($1 \leqslant a_i \leqslant 10^9$).

Формат выходных данных

Выведите t строк; в i-й строке выведите «`FIRST`», если в i-м тесте при правильной игре выигрывает первый игрок, и «`SECOND`», если второй.

Пример

varnim.in	varnim.out
3 1 1 2 1 1 3 1 2 3	FIRST SECOND FIRST

Задача ZF. Woodcut. Дровосек

Имя входного файла:	`woodcut.in`
Имя выходного файла:	`woodcut.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Двое играют в следующую игру: имеется дерево с отмеченной вершиной (корнем). Игроки ходят по очереди. За ход игрок рузрубает ветку (стирает ребро), причем из двух получившихся компонент связности остается только та, которая содержит корень — остальная отваливается и больше в игре не участвует. Проигрывает тот, кто не может сделать ход. Определите, может ли выиграть первый игрок, и если да, то укажите любой из его выигрышных ходов.

Формат входных данных

В первой строке входного файла находятся 2 числа, N и R — количество вершин дерева и номер корня ($1 < N \leqslant 100\,000$, $1 \leqslant R \leqslant N$). Далее следуют $N - 1$ строка, в каждой из которых находятся два числа — номера вершин, которые соединяет очередное ребро.

Формат выходных данных

Выведите в выходной файл одно число 1 или 2 — номер игрока, который выигрывает при правильной игре. Если выигрывает первый игрок, то выведите также любой его выигрышный ход, т.е. порядковый номер ребра во входном файле, которое ему достаточно разрубить первым ходом (число от 1 до $N - 1$).

Пример

woodcut.in	woodcut.out
5 5 2 3 1 3 2 5 4 5	1 1

Задача ZG. Терминатор

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

Два игрока играют в настольную игру. Игровое поле представляет собой квадратный лабиринт, 8×8 клеток. В некоторых клетках располагаются стенки. Один игрок управляет фишкой-терминатором, а второй — фишкой-беглецом. Игроки ходят по очереди, ходы пропускать нельзя (гарантируется, что ход всегда возможен). За один ход игрок может переместить свою фишку в любую из свободных клеток, расположенных рядом с исходной по горизонтали, вертикали или по диагонали (то есть ходом короля). Терминатор, кроме того, может стрелять в беглеца ракетами. Выстрел идет по прямой в любом направлении по горизонтали, вертикали или диагонали. Если беглец оказывается на линии выстрела терминатора и не прикрыт стенками, то терминатор незамедлительно делает выстрел (вне зависимости от того, чей ход), и беглец проигрывает. Начальное положение фишек задано. Первый ход делает беглец. Он выигрывает, если сделает ход с восьмой строки за пределы игрового поля, так как остальные границы поля окружены стенками.

Вопрос задачи: может ли беглец выиграть при оптимальной игре обеих сторон?

Формат входных данных

Во входном файле задано игровое поле. Свободная клетка обозначена цифрой 0, а клетка со стенкой — цифрой 1. Клетка, в которой находится беглец, обозначена цифрой 2, а клетка с терминатором — цифрой 3.

Формат выходных данных

В выходной файл выведите число 1, если беглец выигрывает, и -1 — в противном случае.

Примеры

terminator.in	terminator.out
01000000 10100000 31100000 00020000 00000000 00000000 00000000 00000000	-1

Задача ZH. Сумма игр

Имя входного файла:	`smith.in`
Имя выходного файла:	`smith.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Пусть дан ориентированный граф. Стандартная игра на графе заключается в следующем: изначально на одной из вершин графа (называемой начальной позицией) стоит фишка. Двое игроков по очереди двигают её по рёбрам. Проигрывает тот, кто не может сделать ход.

В теории игр часто рассматриваются более сложные игры. Например, прямая сумма двух игр на графах. Прямая сумма игр — это следующая игра: изначально на каждом графе в начальной позиции стоит по фишке. За ход игрок выбирает любую фишку и двигает по ребру соответствующего графа. Проигрывает тот, кто не может сделать ход.

Ваша задача — опеределить, кто выиграет при правильной игре.

Формат входных данных

На первой строке будут даны числа N_1 и M_1 — количество вершин и рёбер в первом графе ($1 \leqslant N_1, M_1 \leqslant 10\,000$). На следующих M_1 строках содержится по два числа x и y ($1 \leqslant x, y \leqslant N_1$).

В следующих $M_2 + 1$ строках задан второй граф в том же формате.

Заканчивается входной файл списком пар начальных вершин, для которых нужно решить задачу. На первой строке задано число T ($1 \leqslant T \leqslant 100\,000$) — количество пар начальных вершин. В следующих T строках указаны пары вершин v_1 и v_2 ($1 \leqslant v_1 \leqslant N_1$, $1 \leqslant v_2 \leqslant N_2$).

Формат выходных данных

На каждую из T пар начальных вершин выведите строку “`first`”, если при правильной игре выиграет первый, “`second`”, если второй, или “`draw`”, если будет ничья.

Пример

smith.in	smith.out
3 2 1 2 2 3 2 1 1 2 2 1 1 3 2	first second

Задача ZI. Улиточки

Имя входного файла:	`snails.in`
Имя выходного файла:	`snails.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Две улиточки Маша и Петя сейчас находятся в на лужайке с абрикосами и хотят добраться до своего домика. Лужайки пронумерованы числами от 1 до n и соединены дорожками (может быть несколько дорожек соединяющих две лужайки, могут быть дорожки, соединяющие лужайку с собой же). В виду соображений гигиены, если по дорожке проползла улиточка, то вторая по той же дорожке уже ползти не может. Помогите Пете и Маше добраться до домика.

Формат входных данных

В первой строке файла записаны четыре целых числа — n, m, a и h (количество лужаек, количество дорог, номер лужайки с абрикосами и номер домика).

В следующих m строках записаны пары чисел. Пара чисел (x, y) означает, что есть дорожка с лужайки x до лужайки y (из-за особенностей улиток и местности дорожки односторонние).

Ограничения: $2 \leqslant n \leqslant 10^5, 0 \leqslant m \leqslant 10^5, s \neq t$.

Формат выходных данных

Если существует решение, то выведите `YES` и на двух отдельных строчках сначала путь для Машеньки (т.к. дам нужно пропускать вперед), затем путь для Пети. Если решения не существует, выведите `NO`. Если решений несколько, выведите любое.

Пример

snails.in	snails.out
3 3 1 3 1 2 1 3 2 3	YES 1 3 1 2 3

Задача ZJ. Разрез

Имя входного файла:	`cut.in`
Имя выходного файла:	`cut.out`
Ограничение по времени:	0.5 секунды
Ограничение по памяти:	256 мегабайта

Найдите минимальный разрез между вершинами 1 и n в заданном неориентированном графе.

Формат входных данных

На первой строке входного файла содержится n ($1 \leqslant n \leqslant 100$) — число вершин в графе и m ($0 \leqslant m \leqslant 400$) — количество ребер. На следующих m строках входного файла содержится описание ребер. Ребро описывается номерами вершин, которые оно соединяет, и его пропускной способностью (положительное целое число, не превосходящее десять тысяч), при этом никакие две вершины не соединяются более чем одним сегментом.

Формат выходных данных

На первой строке выходного файла должны содержаться количество ребер в минимальном разрезе и их суммарная пропускная способность. На следующей строке выведите возрастающую последовательность номеров ребер (ребра нумеруются в том порядке, в каком они были заданы во входном файле).

Пример

cut.in	cut.out
3 3 1 2 3 1 3 5 3 2 7	2 8 1 2

Задача ZK. Просто поток

Имя входного файла:	`flow.in`
Имя выходного файла:	`flow.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Дана система из узлов и труб, по которым может течь вода. Для каждой трубы известна наибольшая скорость, с которой вода может протекать через нее. Известно, что вода течет по трубам таким образом, что за единицу времени в каждый узел (за исключением двух — источника и стока) втекает ровно столько воды, сколько из него вытекает.

Ваша задача — найти наибольшее количество воды, которое за единицу времени может протекать между источником и стоком, а также скорость течения воды по каждой из труб.

Трубы являются двусторонними, то есть вода в них может течь в любом направлении. Между любой парой узлов может быть более одной трубы.

Формат входных данных

В первой строке записано натуральное число N — количество узлов в системе ($2 \leqslant N \leqslant 100$). Известно, что источник имеет номер 1, а сток номер N. Во второй строке записано натуральное M ($1 \leqslant M \leqslant 5000$) — количество труб в системе. Далее в M строках идет описание труб. Каждая труба задается тройкой целых чисел A_i, B_i, C_i, где A_i, B_i — номера узлов, которые соединяет данная труба ($A_i \neq B_i$), а C_i ($0 \leqslant C_i \leqslant 10^4$) — наибольшая допустимая скорость течения воды через данную трубу.

Формат выходных данных

В первой строке выведите наибольшее количество воды, которое протекает между источником и стоком за единицу времени. Далее выведите M строк, в каждой из которых выведите скорость течения воды по соответствующей трубе. Если направление не совпадает с порядком узлов, заданным во входных данных, то выводите скорость со знаком минус. Числа выводите с точностью 10^{-3}.

Пример

flow.in	flow.out
2 2 1 2 1 2 1 3	4.0000000 1.0000000 -3.0000000

Задача ZL. Паросочетание

Имя входного файла:	`pairs.in`
Имя выходного файла:	`pairs.out`
Ограничение по времени:	0.5 секунды
Ограничение по памяти:	256 мегабайт

Двудольным графом называется граф $(V, E), E \subset V \times V$ такой, что его множество вершин V можно разбить на два подмножества A и B, для которых $\forall (e_1, e_2) \in E\ e_1 \in A, e_2 \in B$ и $A, B \subset E, A \cap B = \varnothing$.

Паросочетанием в двудольном графе называется любой его набор несмежных ребер, то есть такой набор $S \subset E$, что для любых двух ребер $e_1 = (u_1, v_1), e_2 = (u_2, v_2)$ из S выполнено $u_1 \neq u_2$ и $v_1 \neq v_2$.

Ваша задача — найти максимальное паросочтание в двудольном графе, то есть паросочетание с максимально возможным числом ребер.

Формат входных данных

В первой строке записаны два целых числа n и m ($1 \leqslant n, m \leqslant 250$) — число вершин в A и число вершин в B.

Далее следуют n строк с описаниями ребер. i-я вершина из A описана в $i + 1$-й строке файла. Каждая из этих строк содержит номера вершин из B, соединенных с i-й вершиной A. Вершины в A и B нумеруются независимо (с единицы). Список завершается числом 0.

Формат выходных данных

Первая строка выходного файла должна содержать одно целое число l — количество ребер в максимальном паросочетании. Далее должны следовать l строк, в каждой из которых должны быть два целых числа u_j и v_j — концы ребер паросочетания в A и B, соотвественно.

Пример

pairs.in	pairs.out
2 2 1 2 0 2 0	2 1 1 2 2

Задача ZM. Максимальный поток

Имя входного файла:	`flow2.in`
Имя выходного файла:	`flow2.out`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мегабайт

Вам задан ориентированный граф G. Каждое ребро имеет некоторую пропускную способность. Найдите максимальный поток между вершинами 1 и n.

Формат входных данных

Первая строка входного файла содержит n и m — число вершин и ребер в графе ($2 \leqslant n \leqslant 500$, $1 \leqslant m \leqslant 10\,000$). Последующие строки описывают ребра. Каждое ребро задается тремя целыми числами: начальная вершина ребра, конечная вершина ребра и пропускная способность ребра. Пропускные способности не превосходят 10^9.

Формат выходных данных

Выведите величину максимального потока между вершинами 1 и n.

Далее для каждого ребра выведите величину потока, текущую по этому ребру.

Примеры

flow2.in	flow2.out
4 5 1 2 1 1 3 2 3 2 1 2 4 2 3 4 1	3.0 1.0 2.0 1.0 2.0 1.0

Задача ZN. Максимальный поток(быстро)

Имя входного файла:	`flow2.in`
Имя выходного файла:	`flow2.out`
Ограничение по времени:	0.5 секунды
Ограничение по памяти:	256 мегабайт

Вам задан ориентированный граф G. Каждое ребро имеет некоторую пропускную способность. Найдите максимальный поток между вершинами 1 и n.

Формат входных данных

Первая строка входного файла содержит n и m — число вершин и ребер в графе ($2 \leqslant n \leqslant 500$, $1 \leqslant m \leqslant 10\,000$). Последующие строки описывают ребра. Каждое ребро задается тремя целыми числами: начальная вершина ребра, конечная вершина ребра и пропускная способность ребра. Пропускные способности не превосходят 10^9.

Формат выходных данных

Выведите величину максимального потока между вершинами 1 и n.

Далее для каждого ребра выведите величину потока, текущую по этому ребру.

Примеры

flow2.in	flow2.out
4 5 1 2 1 1 3 2 3 2 1 2 4 2 3 4 1	3.0 1.0 2.0 1.0 2.0 1.0

Задача ZO. Равномерный поток

Имя входного файла:	`flow.in`
Имя выходного файла:	`flow.out`
Ограничение по времени:	5 секунд
Ограничение по памяти:	256 мегабайт

Дана система из узлов и труб, по которым может течь вода. Для каждой трубы известна наибольшая скорость, с которой вода может протекать через нее. Известно, что вода течет по трубам таким образом, что за единицу времени в каждый узел (за исключением двух — источника и стока) втекает ровно столько воды, сколько из него вытекает. Более того, известно, что для любой пары узлов (включая источник и сток) сумма скоростей течения воды вдоль любого пути, их соединяющего, постоянна для данной пары узлов. Сумма берется таким образом, что если труба представлена в пути против направления движения воды в ней, то соответствующее слагаемое берется со знаком минус.

Ваша задача — найти наибольшее количество воды, которое за единицу времени может протекать между источником и стоком, а также скорость течения воды по каждой из труб.

Трубы являются двусторонними, то есть вода в них может течь в любом направлении. Между любой парой узлов может быть более одной трубы.

Формат входных данных

В первой строке записано натуральное число N — количество узлов в системе ($2 \leqslant N \leqslant 100$). Известно, что источник имеет номер 1, а сток номер N. Во второй строке записано натуральное M ($1 \leqslant M \leqslant 5000$) — количество труб в системе. Далее в M строках идет описание труб. Каждая труба задается тройкой целых чисел A_i, B_i, C_i, где A_i, B_i — номера узлов, которые соединяет данная труба ($A_i \neq B_i$), а C_i ($0 \leqslant C_i \leqslant 10^4$) — наибольшая допустимая скорость течения воды через данную трубу.

Формат выходных данных

В первой строке выведите наибольшее количество воды, которое протекает между источником и стоком за единицу времени. Далее выведите M строк, в каждой из которых выведите скорость течения воды по соответствующей трубе. Если направление не совпадает с порядком узлов, заданным во входных данных, то выводите скорость со знаком минус. Числа выводите с точностью 10^{-3}.

Пример

flow.in	flow.out
2 1 1 2 1	1.0000000000 1.0000000000
3 2 1 2 1 2 3 2	1.0000000000 1.0000000000 1.0000000000

Задача ZP. Задача о назначениях

Имя входного файла:	`assignment.in`
Имя выходного файла:	`assignment.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Дана целочисленная матрица C размера $n \times n$. Требуется выбрать n ячеек так, чтобы в каждой строке и каждом столбце была выбрана ровно одна ячейка и сумма значений в выбранных ячейках было минимальна.

Формат входных данных

Первая строка входного файла содержит n ($2 \leqslant n \leqslant 300$). Каждая из последующих n строк содержит по n чисел: C_{ij} Все значения во входном файле неотрицательны и не превосходят 10^6.

Формат выходных данных

В первую строку выходного файла выведите одно число — искомая минимизируемая величина. Далее выведите n строк по два числа в каждой — номер строки и столбца клетки, участвующей в оптимальном назначении.

Пары чисел можно выводить в произвольном порядке.

Примеры

assignment.in	assignment.out
3 3 2 1 1 3 2 2 1 3	3 2 1 3 2 1 3

Задача ZQ. Общий предок

Имя входного файла:	`lca.in`
Имя выходного файла:	`lca.out`
Ограничение по времени:	0.5 секунды
Ограничение по памяти:	256 мегабайт

Дано подвешенное дерево с корнем в 1-й вершине и M запросов вида "найти у двух вершин наименьшего общего предка".

Формат входных данных

В первой строке файла записано одно число N — количество вершин. В следующих $N-1$ строках записаны числа. Число x на строке $2 \leqslant i \leqslant N$ означает, что x — отец вершин i. $(x < i)$. На следующей строке число M. Следующие M строк содержат запросы вида (x, y) — найти наименьшего предка вершин x и y. Ограничения: $1 \leqslant N \leqslant 5 \cdot 10^4, 0 \leqslant M \leqslant 5 \cdot 10^4$.

Формат выходных данных

M ответов на запросы.

Пример

lca.in	lca.out
5 1 1 2 3 2 2 3 4 5	1 1

Задача ZR. Словарь

Имя входного файла:	`dictionary.in`
Имя выходного файла:	`dictionary.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Дан набор слов и текст, требуется определить для каждого слова, присутствует ли оно в тексте как подстрока.

Формат входных данных

В первой строке дан текст (не более 10^6 строчных латинских букв). Далее дано число M — количество слов в словаре.

В следующих M строках записаны слова (не более 30 строчных латинских букв). Слова различны и отсортированы в лексикографическом порядке.

Суммарная длина слов в словаре не более 10^5.

Формат выходных данных

M строк вида `Yes`, если слово присутствует, и `No` иначе.

Пример

dictionary.in	dictionary.out
trololo 3 abacabadabacaba olo trol	No Yes Yes

Задача ZS. Сравнения подстрок

Имя входного файла:	`substrcmp.in`
Имя выходного файла:	`substrcmp.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Дана строка. Нужно уметь отвечать на запросы вида: равны ли подстроки [a..b] и [c..d].

Формат входных данных

Сперва строка S (не более 10^5 строчных латинских букв). Далее число M — количество запросов.

В следующих M строках запросы a,b,c,d. $0 \leqslant M \leqslant 10^5, 1 \leqslant a \leqslant b \leqslant |S|, 1 \leqslant c \leqslant d \leqslant |S|$

Формат выходных данных

M строк. Выведите `Yes`, если подстроки совпадают, и `No` иначе.

Пример

substrcmp.in	substrcmp.out
trololo 3 1 7 1 7 3 5 5 7 1 1 1 5	Yes Yes No

Задача ZT. Префикс-функция

Имя входного файла:	`prefix.in`
Имя выходного файла:	`prefix.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мебибайт

Дана строка s. Найдите сумму значений префикс-функции для всех позиций строки s.

Формат входных данных

Во входном файле записана единственная строка s ($1 \leqslant |s| \leqslant 150\,000$).

Формат выходных данных

В выходной файл выведите одно число — ответ на задачу.

Пример

prefix.in	prefix.out
abcaabacabc	11

Задача ZU. Кубики

Имя входного файла:	`cubes.in`
Имя выходного файла:	`cubes.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Привидение Петя любит играть со своими кубиками. Он любит выкладывать их в ряд и разглядывать свое творение. Однако недавно друзья решили подшутить над Петей и поставили в его игровой комнате зеркало. Ведь всем известно, что привидения не отражаются в зеркале! А кубики отражаются.

Теперь Петя видит перед собой N цветных кубиков, но не знает, какие из этих кубиков нестоящие, а какие — всего лишь отражение в зеркале. Помогите Пете! Выясните, сколько кубиков может быть у Пети. Петя видит отражение всех кубиков в зеркале и часть кубиков, которая находится перед ним. Часть кубиков может быть позади Пети, их он не видит.

Формат входных данных

Первая строка входного файла содержит число N ($1 \leqslant N \leqslant 100\,000$) и количество различных цветов, в которые могут быть раскрашены кубики — M ($1 \leqslant M \leqslant 100\,000$). Следующая строка содержит N целых чисел от 1 до M — цвета кубиков.

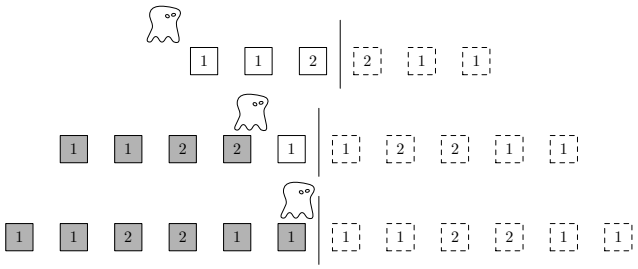
Формат выходных данных

Выведите в выходной файл все такие K, что у Пети может быть K кубиков.

Пример

cubes.in	cubes.out
6 2 1 1 2 2 1 1	3 5 6

В приведенном примере взаимные расположения Пети, кубиков и зеркала приведены на рисунке. Петя смотрит вправо, затененные на рисунке кубики находятся позади Пети и поэтому он их не видит.



Задача ZV. Основание строки

Имя входного файла:	`basis.in`
Имя выходного файла:	`basis.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Строка S была записана много раз подряд, после чего из получившейся строки взяли подстроку и дали вам. Ваша задача определить минимально возможную длину исходной строки S.

Формат входных данных

В первой и единственной строке входного файла записана строка, которая содержит только латинские буквы, длина строки не превышает 50 000 символов.

Формат выходных данных

В выходной файл выведите ответ на задачу.

Пример

basis.in	basis.out
zzz	1
bcabcab	3

Задача ZW. Циклические суффиксы

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

Рассмотрим строку $S = s_1 s_2 s_3 \dots s_{n-1} s_n$ над алфавитом Σ. *Циклическим расширением* порядка m строки S назовем строку $s_1 s_2 s_3 \dots s_{n-1} s_n s_1 s_2 \dots$ из m символов; это значит, что мы приписываем строку S саму к себе, пока не получим требуемую длину, и берем префикс длины m.

Циклической строкой $\tilde{S}$ назовем бесконечное циклическое расширение строки S.

Рассмотрим суффиксы циклической строки $\tilde{S}$. Очевидно, существует не более $|S|$ различных суффиксов: $(n+1)$-ый суффикс совпадает с первым, $(n+2)$-ой совпадает со вторым, и так далее. Более того, различных суффиксов может быть даже меньше. Например, если $S =$ `abab`, первые четыре суффикса циклической строки $\tilde{S}$ — это:

$$\begin{aligned}
\tilde{S}_1 &= \texttt{abababab}\dots \\
\tilde{S}_2 &= \texttt{bababababa}\dots \\
\tilde{S}_3 &= \texttt{abababab}\dots \\
\tilde{S}_4 &= \texttt{bababababa}\dots
\end{aligned}$$

Здесь существует всего два различных суффикса, в то время как $|S| = 4$.

Отсортируем первые $|S|$ суффиксов $\tilde{S}$ лексикографически. Если два суффикса совпадают, первым поставим суффикс с меньшим индексом. Теперь нас интересует следующий вопрос: на каком месте в этом списке стоит сама строка $\tilde{S}$?

Например, рассмотрим строку $S =$ `cabcab`:

$$\begin{aligned}
(1)\quad \tilde{S}_2 &= \texttt{abcabcabca}\dots \\
(2)\quad \tilde{S}_5 &= \texttt{abcabcabca}\dots \\
(3)\quad \tilde{S}_3 &= \texttt{bcabcabcab}\dots \\
(4)\quad \tilde{S}_6 &= \texttt{bcabcabcab}\dots \\
(5)\quad \tilde{S}_1 &= \texttt{cabcabcabc}\dots \\
(6)\quad \tilde{S}_4 &= \texttt{cabcabcabc}\dots
\end{aligned}$$

Здесь циклическая строка $\tilde{S} = \tilde{S}_1$ находится на пятом месте.

Вам дана строка S. Ваша задача — найти позицию циклической строки $\tilde{S}$ в описанном порядке.

Формат входных данных

Во входном файле записана единственная строка S ($1 \leqslant |S| \leqslant 1\,000\,000$), состоящая из прописных латинских букв.

Формат выходных данных

В выходной файл выведите единственное число — номер строки $\tilde{S}$ в описанном порядке среди первых $|S|$ суффиксов.

Примеры

cyclic.in	cyclic.out
abracadabra	3
cabcab	5

Задача ZX. Палиндромы

Имя входного файла:	`palindrome.in`
Имя выходного файла:	`palindrome.out`
Ограничение по времени:	0.5 секунд
Ограничение по памяти:	128 мебибайт

Строка называется палиндромом, если она одинаково читается как слева направо, так и справа налево. Например, `abba` — палиндром, а `omax` — нет. Для строки α будем обозначать $\alpha[i..j]$ ее подстроку длины $j-i+1$ с i-й по j-ю позицию включительно (позиции нумеруются с единицу). Для заданной строки α длины N ($1 \leqslant N \leqslant 100\,000$) требуется подсчитать число q пар (i, j), $1 \leqslant i < j \leqslant n$, таких что $\alpha[i..j]$ является палиндромом.

Формат входных данных

Входной файл содержит одну строку α длины N, состоящую из маленьких латинских букв.

Формат выходных данных

В выходной файл выведите искомое число q.

Примеры

palindrome.in	palindrome.out
aaa	3
abba	2
omax	0

Задача ZY. Преобразование строковых функций

Имя входного файла:	`trans.in`
Имя выходного файла:	`trans.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мебибайт

Для строки S определим Z-функцию следующим образом: $Z[i] = lcp(S, S[i..|S|])$, где $lcp(S_1, S_2)$ равно длине наибольшего общего префикса строк S_1 и S_2. Например, для $S = abacabaa$ Z-функция равна $[8, 0, 1, 0, 3, 0, 1, 1]$.

Для строки S определим ее префикс-функцию: $\pi[i] = \max\{k | 0 \leqslant k < i, S[1..k] = S[i-k+1..i]\}$. Например, для $S = abacabaa$ ее префикс-функция имеет вид: $[0, 0, 1, 0, 1, 2, 3, 1]$.

Для некоторой строки S была посчитана ее Z-функция, а строка S была утеряна. Ваша задача получить ее префикс-функцию по заданной Z-функции.

Формат входных данных

В первой строке входного файла содержится натуральное число N ($1 \leqslant N \leqslant 200\,000$), где N — длина S. Во второй строке записана Z-функция строки S.

Формат выходных данных

Выведите N чисел — искомую префикс-функцию.

Пример

trans.in	trans.out
8 8 0 1 0 3 0 1 1	0 0 1 0 1 2 3 1

Задача ZZ. Преобразование строковых функций: обратная задача

Имя входного файла:	`invtrans.in`
Имя выходного файла:	`invtrans.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мебибайт

Для строки S определим Z-функцию следующим образом: $Z[i] = lcp(S, S[i..|S|])$, где $lcp(S_1, S_2)$ равно длине наибольшего общего префикса строк S_1 и S_2. Например, для $S = abacabaa$ Z-функция равна $[8, 0, 1, 0, 3, 0, 1, 1]$.

Для строки S определим ее префикс-функцию: $\pi[i] = \max\{k | 0 \leqslant k < i, S[1..k] = S[i-k+1..i]\}$. Например, для $S = abacabaa$ ее префикс-функция имеет вид: $[0, 0, 1, 0, 1, 2, 3, 1]$.

Для некоторой строки S была посчитана ее префикс-функция, а строка S была утеряна. Ваша задача получить ее Z-функцию по заданной префикс-функции.

Формат входных данных

В первой строке входного файла содержится натуральное число N ($1 \leqslant N \leqslant 200\,000$), где N — длина S. Во второй строке записана префикс-функция строки S.

Формат выходных данных

Выведите N чисел — искомую Z-функцию.

Пример

invtrans.in	invtrans.out
8 0 0 1 0 1 2 3 1	8 0 1 0 3 0 1 1

Задача ZZA. Обратная префикс-функция

Имя входного файла:	`inverse_pref.in`
Имя выходного файла:	`inverse_pref.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мебибайт

Для строки S определим ее префикс-функцию: $\pi[i] = \max\{k|0 \leqslant k < i, S[1..k] = S[i-k+1..i]\}$ для всех $1 \leqslant i \leqslant N$, где N — длина строки. Например, для $S = abacabaa$ ее префикс-функция имеет вид: $[0, 0, 1, 0, 1, 2, 3, 1]$.

Ваша задача — по заданной префикс-функции восстановить строку. В качестве символов строки разрешается использование M первых строчных букв латинского алфавита.

Формат входных данных

Входной файл состоит из одного или более набора входных данных.

В первой строке каждого набора записаны два целых числа N, M ($N \geqslant 1, 1 \leqslant M \leqslant 26$). Во второй строке записана последовательность целых чисел $\pi[1], \pi[2], \ldots, \pi[N]$. Все числа в последовательности целые неотрицательные, не превосходящие 10^6.

Сумма значений N по всем наборам не превосходит 10^6, количество наборов входных данных не превосходит 10^5.

Формат выходных данных

Выведите в первую строку выходного файла `YES`, если существует искомое слово, и `NO` в противном случае. В случае положительного ответа выведите во вторую строку выходного файла выведите искомое слово. Если решений несколько, выведите любое.

Пример

inverse_pref.in	inverse_pref.out
8 3 0 0 1 0 1 2 3 1 1 1 10	YES abacabaa NO

Задача ZZB. Суффиксный массив

Имя входного файла:	`suffarray.in`
Имя выходного файла:	`suffarray.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Данна строка, требуется построить суффиксный массив для этой строки. Суффиксный массив — лексикографически отсортированный массив всех суффиксов строки. Каждый суффикс задается целым числом — позицией начала.

Строка s лексикографически меньше строки t, если есть такое i, что $s_i < t_i$ и $s_j = t_j$ для всех $j < i$. Или, если такого i не существует и строка s короче строки t.

Здесь s_i — код i-го символа строки s.

Формат входных данных

Файл состоит из единственной строки. Эта строка — **английский литературный текст**. Длина текста не превосходит 10^5. Коды всех символов в тексте от 32 до 127.

Формат выходных данных

Выведите N чисел — суффиксный массив данной строки.

Пример

suffarray.in	suffarray.out
99 bottles of beer.	14 3 11 19 2 1 15 4 16 17 9 13 8 12 5 18 10 7 6

Задача ZZC. LCP для суффиксного массива

Имя входного файла:	`sufflcp.in`
Имя выходного файла:	`sufflcp.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Дана строка длины N и отсортированный массив суффиксов этой строки (т.е. суффиксный массив), вам нужно вычислить `LCP`. При сортировке строка `a` считается меньше строки `aa`. `LCP` — наибольший общий префикс двух последовательных суффиксов в суффиксном массиве.

Формат входных данных

В первой строке число N ($1 \leqslant N \leqslant 10^5$). На второй строке файла дана N строчных латинских букв. В третьей строке N чисел от 1 до N — суффиксный массив (числом i кодируется суффикс, начинающийся с i-го символа).

Формат выходных данных

Выведите $N - 1$ число — значения `LCP`.

Пример

sufflcp.in	sufflcp.out
5 cacao 2 4 1 3 5	1 0 2 0

Замечание

Суффиксный массив для строки `cacao`:

```
acao
ao
cacao
cao
o
```

Задача ZZD. Подстроки

Имя входного файла:	`substr.in`
Имя выходного файла:	`substr.out`
Ограничение по времени:	0.4 секунды
Ограничение по памяти:	16 мегабайт

Дана строка s. Вам требуется подсчитать количество её различных подстрок. Пустую строку учитывать не следует.

Формат входных данных

В единственной строке входного файла содержится данная строка s, состоящая из строчных латинских букв. Длина строки не превосходит 20 000 символов.

Формат выходных данных

В единственной строке выходного файла выведите единственное число — количество различных подстрок s.

substr.in	substr.out
aaaa	4
abacaba	21

Задача ZZE. Башни

Имя входного файла:	`towers.in`
Имя выходного файла:	`towers.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	64 мебибайта

Задано число n и последовательность из n чисел. Требуется рассмотреть все возможные циклические сдвиги заданной последовательности, отсортировать их в лексикографическом порядке, и вывести сумму наибольших общих префиксов соседних в этом порядке сдвигов.

Формат входных данных

Тестовый пример состоит из двух строк. Первая из них содержит целое число $1 \leqslant n \leqslant 50000$ — количество магических башен. Вторая строка содержит n чисел в интервале от 0 до 100 — заданную последовательность.

Формат выходных данных

Выведите одно число — искомую сумму.

Пример

towers.in	towers.out
11 12 8 18 18 8 18 18 8 15 15 8	13

Задача ZZF. Рефрен

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

Рассмотрим последовательность n целых чисел от 1 до m. Подпоследовательность подряд идущих чисел называется *рефреном*, если произведение ее длины на количество вхождений в последовательность максимально.

По заданной последовательности требуется найти ее рефрен.

Формат входных данных

Первая строка входного файла содержит два целых числа: n и m ($1 \leqslant n \leqslant 150\,000$, $1 \leqslant m \leqslant 10$).

Вторая строка содержит n целых чисел от 1 до m.

Формат выходных данных

Первая строка выходного файла должна содержать произведение длины рефрена на количество ее вхождений. Вторая строка должна содержать длину рефрена. Третья строка должна содержать последовательность которая является рефреном.

Пример

refrain.in	refrain.out
9 3 1 2 1 2 1 3 1 2 1	9 3 1 2 1

Задача ZZG. Суффиксное дерево

Имя входного файла:	`suftree.in`
Имя выходного файла:	`suftree.out`
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

Дана строка s. Постройте сжатое суффиксное дерево для строки s и выведите его. Найдите такое дерево, которое содержит минимальное количество вершин.

Формат входных данных

В первой строке записана строка s ($1 \leqslant |s| \leqslant 10^5$), последний символ строки доллар «$», остальные символы строки маленькие латинские буквы.

Формат выходных данных

Пронумеруйте вершины дерева от 0 до $n-1$ в порядке обхода в глубину, обходя поддеревья в порядке лексикографической сортировки исходящих из вершины рёбер. Используйтся ASCII-коды символов для определения их порядка.

В первой строке выведите число n – количество вершин дерева. В следующих $n-1$ строках выведите описание вершин дерева, кроме корня, в порядке увелечения их номеров.

Описание вершины дерева v состоит из трёх целых чисел: p, lf, rf, где p ($0 \leqslant p \leqslant n, p \neq v$) – номер родителя текущей вершины. На ребер ведущем из p в v написана подстрока $s[lf..rf)$ ($0 \leqslant lf < rf \leqslant |s|$).

Примеры

suftree.in	suftree.out
aaa$	7 0 3 4 0 0 1 2 3 4 2 1 2 4 3 4 4 2 4
b$	3 0 1 2 0 0 2
ababa$	10 0 5 6 0 0 1 2 5 6 2 1 3 4 5 6 4 3 6 0 1 3 7 5 6 7 3 6

Задача ZZH. Blackboard

Имя входного файла:	`blackboard.in`
Имя выходного файла:	`blackboard.out`
Ограничение по времени:	1 second
Ограничение по памяти:	256 megabytes

A new electronic blackboard has been recently developed by the Anti-Chalk Membership (ACM). It holds a sequence of bits (displayed as zeroes and ones). Since the new blackboard is a high-tech product no chalk is allowed nearby. To alter the content of the blackboard it has four buttons conveniently located on its sides. The buttons are named L0, R0, L1, and R1. When a user presses L0 button the leftmost bit of the sequence is erased, hence shifting the sequence one position to the left. Next, a 0-bit is appended to the right of the sequence. Button L1 acts similarly except for it adds 1-bit to the rightafter erasing the leftmost bit. Buttons R0 and R1 act in a similar way but shift the sequence in the opposite direction. Namely, button R0 erases the rightmost bit shifting the sequence to the right and then appends 0-bit to the left. Button R1 appends 1-bit instead of 0-bit. Changing the information displayed on the blackboard is quite cumbersome, so the user would normally want to minimize the number of button presses. Given the initial and the final configurations, your task is to transform the former to the latter with the fewest possible operations.

Формат входных данных

The first line of the input contains the initial sequence currently displayed on the blackboard. The second line contains the desired final sequence. Both sequences are nonempty, have the same length not exceeding 100 000, and consist of zeros and ones (without spaces).

Формат выходных данных

Output the minimum number of button presses.

Пример

blackboard.in	blackboard.out
0111 0110	2
0110 1111	3