

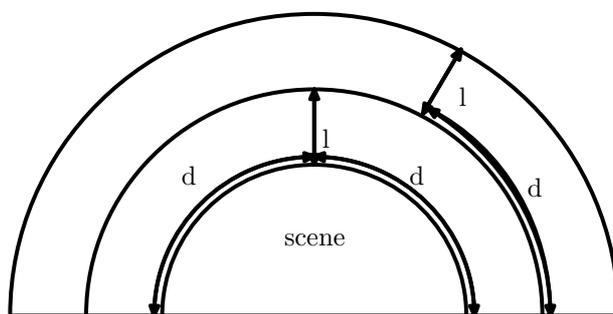
Задача А. Амфитеатр

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

Как известно, древние греки очень любили строить амфитеатры. Амфитеатр — это здание в форме полукруга, часть которого, а именно полукруг с тем же центром, но с меньшим радиусом, и лежащий внутри амфитеатра, занята сценой. Остальное же место отводится под сидения.

Недавно греки современные решили построить новейший амфитеатр с радиусом R метров и радиусом сцены r метров. Теперь они хотят узнать, сколько же зрителей будет вмещать сооружение.

Зрители сидят на скамьях толщины l сантиметров вдоль окружности лицом к сцене. Каждая скамья полностью находится внутри амфитеатра. Один грек, садясь на скамью занимает d сантиметров (вдоль внутренней кромки скамьи). При этом греки могут сидеть вплотную.



Формат входного файла

Четыре натуральных числа R , r , l и d , не превосходящие 1000.

Формат выходного файла

В выходной файл выведите одно число — максимальное количество греков, которые могут уместиться в амфитеатре.

Примеры

<code>amfi.in</code>	<code>amfi.out</code>
2 1 100 314	1
100 10 200 100	7611

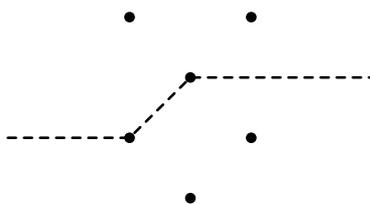
Задача В. Раздел королевства

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

Король умер, да здравствует... эээ, у короля осталось три сына. Они решили разделить королевство на три части, в соответствии со своими профессиями. Старший сын занимается геологией, так что он предпочитает северные края, богатые нефтью и газом. Средний сын занимается туризмом, он хочет получить южные земли с их песчаными пляжами и ласковым морем. А младший сын выбрал карьеру военного, он хочет получить границу между частями своих братьев и охранять их от посягательств друг друга.

В королевстве n городов и братья решили разделить королевство границей, которая бы представляла собой бесконечную ломаную. Бесконечные части ломаной должны быть параллельны направлению запад-восток, а все концы отрезков и лучей ломаной должны лежать в городах. Каждый меридиан (линия, параллельная направлению север-юг) должна пересекать границу ровно в одной точке.

Города на границе будут отданы младшему сыну, города к северу от границы — старшему сыну, а города к югу от границы — среднему сыну. Хотелось бы разделить королевство так, чтобы разница между максимальным и минимальным количеством городов, которое получают сыновья была минимальная. Помогите им!



Формат входного файла

Первая строка входного файла содержит n — количество городов ($1 \leq n \leq 100$). Следующие n строк содержат координаты городов, каждая строка содержит два целых числа: x_i и y_i ($-10^4 \leq x_i, y_i \leq 10^4$, ось Ox направлена с запада на восток, а ось Oy с юга на север).

Формат выходного файла

Первая строка выходного файла должна содержать k — количество городов на границе ($1 \leq k \leq n$). Вторая строка должна содержать k целых чисел — номера городов на границе в порядке прохода вдоль нее с запада на восток. Города пронумерованы от 1 до n в том порядке, в котором они описаны во входном файле. Соседние отрезки ломаной могут быть параллельны, но никакой город, не приведенный во второй строке, не должен лежать на границе.

Пример

<code>division.in</code>	<code>division.out</code>
6	2
0 1	1 4
1 0	
2 1	
1 2	
2 3	
0 3	

Задача С. Новые мэры

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

Во Флатландии n городов, некоторые из которых соединены двунаправленными дорогами. В целях национальной безопасности следующее условие выполнено для каждого города u . Обозначим как $N(u)$ множество городов, с которыми u соединен дорогами. Тогда для каждой пары городов v и w в $N(u)$ существует путь из v в w , проходящий только по дорогам, соединяющим города из $N(u)$ (но не u). Также можно добраться из любого города до любого другого по дорогам.

Недавно президент Флатландии решил назначить новых мэров во всех городах. Мэра должны быть выбраны из представителей трех основных партий: «Единая Флатландия», «Справедливая Флатландия» и «Либеральная Флатландия». Поскольку президент не хочет чтобы одна из партий стала слишком сильной, он хотел бы, чтобы никакие два города с мэрами из одной и той же партии не были соединены дорогой.

Вы, министр внутренних дел Флатландии, назначены для решения этой задачи.

Формат входного файла

Первая строка входного файла содержит два целых числа: n и m — количество городов и дорог во Флатландии, соответственно ($1 \leq n \leq 500$, $0 \leq m \leq 10\,000$). Следующие m строк описывают дороги, каждая дорога описывается номерами городов, которые она соединяет. Никакая дорога не соединяет город сам с собой, любые два города непосредственно связаны не более чем одной дорогой.

Формат выходного файла

Если можно назначить мэров в города искомым способом, то выведите строку “Plan OK” на первой строке выходного файла. Вторая строка должна содержать n символов, для каждого города выведите ‘R’ если его мэр должен быть назначен из партии «Единая Флатландия», ‘G’ если из партии «Справедливая Флатландия», ‘B’ если из партии «Либеральная Флатландия».

Если назначить мэров искомым способом невозможно, выведите “Plan failed” на первой строке выходного файла.

Пример

<code>mayors.in</code>	<code>mayors.out</code>
5 8 1 2 1 3 1 4 1 5 2 3 3 4 4 5 2 5	Plan OK RGBGB
4 6 1 2 1 3 1 4 2 3 3 4 4 2	Plan failed

Задача D. Числа

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

Задано n положительных целых чисел a_1, a_2, \dots, a_n . Необходимо выбрать из них как можно больше чисел так, чтобы сумма выбранных чисел делилась на 3.

Формат входного файла

Первая строка входного файла содержит целое число n ($1 \leq n \leq 10^5$). Вторая строка входного файла содержит n целых чисел a_1, \dots, a_n , находящихся в диапазоне от 0 до 10^9 .

Формат выходного файла

В первой строке выходного файла выведите количество выбранных чисел. Во второй строке выведите их номера во входном файле в порядке возрастания. Числа нумеруются в порядке их перечисления во входном файле натуральными числами от 1 до n .

Если ответов несколько, выведите любой.

Примеры

<code>numbers.in</code>	<code>numbers.out</code>
3 4 5 6	3 1 2 3
3 4 2 5	2 1 2

Задача Е. Последовательность

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

Пусть x — натуральное число. Обозначим как $s(x)$ сумму цифр квадратов его делителей. Например, $s(6) = 1 + 4 + 9 + (3 + 6) = 23$, $s(10) = 1 + 4 + (2 + 5) + (1 + 0 + 0) = 13$.

Рассмотрим теперь последовательность $a_1 = x$, $a_2 = s(x)$, $a_3 = s(s(x))$, \dots , $a_n = s(a_{n-1})$, \dots .

Скажем, что последовательность a_n имеет длину периода L и длину предпериода P , если при любом $i > P$ верно равенство $a_i = a_{i+L}$, при этом и L и P минимальные числа, удовлетворяющие этому условию.

Ваша задача — написать программу, вычисляющую длину периода и предпериода описанной выше последовательности $a_1 = x$, $a_2 = s(x)$, $a_3 = s(s(x))$, \dots , $a_n = s(a_{n-1})$, \dots .

Формат входного файла

Входной файл содержит целое число x ($1 \leq x \leq 10^9$).

Формат выходного файла

В выходной файл выведите два числа: длину предпериода P и длину периода L . Если последовательность непериодическая, то выведите в выходной файл `-1 -1`.

Примеры

<code>seq.in</code>	<code>seq.out</code>
16	3 7
1	0 1

Задача F. Покорители вселенной

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

Колонисты с земли заселяют вселенную. Все больше и больше молодых семей выбирают в качестве своей судьбы не Землю, а другие планеты. Причем они не могут передумать потом — гипертуннели, используемые для переброски на другие планеты, можно использовать только в одном направлении...

Вы работаете в координационном центре колонизации. В процессе колонизации используется n планет, пронумерованных от 1 до n , при этом Земля имеет номер 1. Планеты пронумерованы в порядке убывания торсионного потенциала, поэтому гипертуннель может существовать только от планеты с меньшим номером к планете с большим номером.

Иногда покорители пространств обнаруживают новый гипертуннель от одной планеты до другой. Иногда, из-за помех в торсионных полях, гипертуннель пропадает (не будем горевать о тех, кто в этот момент путешествовал через него). Ваша задача — отслеживая все эти события, быстро отвечать, существует ли путь от одной планеты до другой, и приводить некоторую другую статистику.

Формат входного файла

Первая строка входного файла содержит n — количество планет ($1 \leq n \leq 200$), и m — количество гипертуннелей на начало рабочего дня. Следующие m строк описывают гипертуннели, каждый туннель описывается двумя целыми числами u и v — номером планеты, с которой туннель ведет и номером планеты, на которую он ведет ($1 \leq u < v \leq n$).

Следующая строка содержит число k — количество событий и запросов ($1 \leq k \leq 100\,000$, количество запросов не превышает 1000). Следующие k строк содержат информацию о событиях и запросах. Каждое событие описывается либо как “+ i j ”, если появляется гипертуннель между планетами i и j , либо как “- i j ”, если гипертуннель между планетами i и j исчезает. Если туннель между планетами уже существует, он не может появиться, аналогично туннель может исчезнуть только если он существует. Запрос имеет вид “? i j ” и означает вопрос, есть ли путь по гипертуннелям с планеты i на планету j . Во всех событиях и запросах $i < j$.

Формат выходного файла

Выведите $k + 1$ строку. Первая строка должна содержать количество пар планет (u, v) , таких что можно добраться от планеты u до планеты v по гипертуннелям, которые есть в начале. После каждого события выведите новое количество таких пар. На каждый запрос выведите “YES” или “NO”.

Пример

<code>settling.in</code>	<code>settling.out</code>
4 4	5
1 2	NO
2 4	6
1 3	6
3 4	3
6	4
? 2 3	YES
+ 2 3	
- 1 3	
- 1 2	
+ 1 4	
? 2 3	

Задача G. Телешоу

Имя входного файла:	<code>show.in</code>
Имя выходного файла:	<code>show.out</code>
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

В последнее время набирают популярность телешоу, в которых телезрителям предлагается испытать свою эрудицию и удачу. Одно из таких шоу идет на канале ЖТВ в Китае.

Шоу протекает следующим образом. Игровое поле состоит из $n \times n$ клеток, в каждую из которых можно записать один иероглиф. На диагонали поля иероглифы записаны в начале игры. Также дано m фраз, каждая из которых содержит ровно n иероглифов. Для каждого i найдется не более двух фраз, таких что их i -й иероглиф равен иероглифу, записанному в i -м ряду на игровом поле.

Игрок должен разместить фразы в ряды игрового поля таким образом, чтобы i -й иероглиф фразы, записанной в i -м ряду совпал с иероглифом, там уже записанном. Кроме того, в каждом столбце игрового поля все иероглифы должны быть различны.

За каждую фразу, которую игрок размещает на поле, он получает небольшой приз, а также суперприз, если он разместит фразу в каждом ряду поля. Помогите понять, можно ли вообще выиграть суперприз.

Формат входного файла

Будем обозначать иероглифы целыми числами от 1 до 10^9 .

Первая строка входного файла содержит число n — размер игрового поля ($1 \leq n \leq 200$). Следующая строка содержит n целых чисел — иероглифы, записанные на диагонали.

Третья строка содержит целое число m ($n \leq m \leq 2n$), за которым следует m строк. Каждая строка содержит n целых чисел — иероглифы, из которых составлена соответствующая фраза. Переставлять иероглифы внутри фразы запрещается. Для каждого i не более двух фраз, таких что i -й иероглиф равен иероглифу, записанному в i -м ряду игрового поля.

Формат выходного файла

Если суперприз можно выиграть, выведите “YES” на первой строке выходного файла. На второй строке в этом случае выведите n целых чисел — фразы, которые надо расположить на игровом поле, перечисленные сверху вниз, фразы пронумерованы от 1 до m в том порядке, в котором они заданы во входном файле.

Если выиграть суперприз нельзя, выведите на первой строке выходного файла слово “NO”.

Пример

show.in	show.out
4 1 2 3 4 7 1 5 2 5 1 5 4 1 2 3 3 1 3 2 4 1 2 2 2 2 3 3 3 3 5 4 5 4	YES 2 5 6 7
4 1 2 3 4 7 1 5 2 5 1 3 4 1 2 3 3 1 3 2 4 1 2 2 2 2 3 3 3 3 5 4 5 4	NO

Задача Н. Сумма на отрезке

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

Задано n целых чисел a_1, a_2, \dots, a_n . Суммой на отрезке $[l, r]$ ($1 \leq l \leq r \leq n$) назовем число, равное $\sum_{i=l}^r a_i = a_l + \dots + a_r$.

Напишите программу, которая определяет, существует ли отрезок, сумма на котором равна x .

Формат входного файла

Первая строка входного файла содержит целое число n ($1 \leq n \leq 100\,000$). Вторая строка входного файла содержит n целых чисел a_i , отделенных друг от друга пробелами. Все a_i не превосходят 10^9 по абсолютной величине. Третья строка входного файла содержит целое число x ($|x| \leq 10^{14}$).

Формат выходного файла

В первой строке выходного файла выведите два числа l и r ($l \leq r$), задающих искомый отрезок. Если сумма ни на одном отрезке не равна x , то выведите `-1 -1`.

Пример

<code>sum.in</code>	<code>sum.out</code>
4 1 4 2 3 5	3 4
4 1 2 3 4 -5	-1 -1

Задача I. Сложный тест

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

Жюри интернет-олимпиад готовит очередную, 239-ю олимпиаду 566-го сезона. На этот раз задача подразумевает решение с помощью алгоритма Дейкстры, и жюри хочет подготовить сложный тест для этого алгоритма.

Алгоритм Дейкстры используется для поиска кратчайшего пути от одной вершины до остальных в графе, веса ребер в котором неотрицательный. Алгоритм работает следующим образом. Пусть G — взвешенный ориентированный граф, обозначим его множество вершин как V , множество ребер как E , а весовую функцию как $w : E \rightarrow \mathbb{N}^+$. Пусть все вершины графа достижимы из некоторой вершины s , кратчайшие пути из которой и будем искать.

Алгоритм поддерживает множество вершин U , которое в начале работы пусто. Каждая вершина помечена некоторым целым числом либо $+\infty$, исходно все вершины помечены $+\infty$, кроме s , которая помечена 0. Обозначим пометку вершины v как $d[v]$.

Шаг алгоритма заключается в следующем. Выбирается вершина с минимальной пометкой, не принадлежащая U . Обозначим эту вершину как u . Вершина u добавляется в множество U , а также для каждого ребра $uv \in E$ производится *релаксация*. Релаксация заменяет значение $d[v]$ значением $\min(d[v], d[u] + w(uv))$. Если пометка вершины v в процессе релаксации меняется, говорят, что релаксация является *активной*. Алгоритм завершается когда все вершины принадлежат множеству U .

Теперь жюри хочет создать граф с n вершинами и m ребрами, такой что алгоритм Дейкстры, будучи запущенным на этом графе, производит как можно больше активных релаксаций. Чтобы избежать недетерминированного выбора, потребуем, чтобы каждый раз при выборе вершины с минимальной пометкой среди вершин не в U такая вершина была единственной.

Формат входного файла

Первая строка входного файла содержит два целых числа: n и m — количество вершин и ребер в требуемом графе, соответственно ($4 \leq n \leq 1000$, $n - 1 \leq m \leq n^2/5$).

Формат выходного файла

Выведите m строк — ребра графа. Каждая строка должна содержать три целых числа: начало ребра, конец ребра и вес ребра. Веса должны быть неотрицательными и не должны превышать 10^6 . Все вершины должны быть достижимы из вершины 1.

Алгоритм Дейкстры, запущенный на выведенном графе с $s = 1$ должен совершить максимальное возможное количество активных релаксаций среди всех графов с n вершинами и m ребрами.

В графе не должно быть петель или параллельных ребер.

Пример

test.in	test.out
4 3	1 2 0 1 3 1 1 4 2
5 5	1 2 0 1 3 1 2 4 4 3 4 2 1 5 2