

## Задача А. Скип или не скип

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	1.5 секунд
Ограничение по памяти:	256 мегабайт

На улице уже 3024 год, идеи для задач давно закончились, а МКОШП теперь проходит в изменённом индивидуальном формате. Жюри выбирают  $n$  задач из всех, которые когда либо существовали, нумеруют их от 1 до  $n$ , присваивают каждой задаче стоимость  $a_i$ , а так же выбирают для каждой задачи некоторый параметр  $b_i$  от 1 до  $n$ .

Изначально тестирующая система выдаёт участнику **первую** задачу. Когда участнику выдают  $i$ -ю задачу, у него есть два варианта:

- либо он сдаёт задачу и получает  $a_i$  баллов.
- либо он может скипнуть задачу, тогда он никогда не сможет её сдать.

Далее тестирующая система выбирает для участника следующую задачу среди задач с номерами  $j$ , такими что:

- если он сдал задачу, она смотрит на задачи с номерами  $j < i$ .
- если он скипнул задачу, она смотрит на задачи с номерами  $j \leq b_i$ .

Среди этих задач она выбирает задачу с **максимальным** номером, которую она ранее **не выдавала** участнику (до этого он её не сдавал и не скипал). Если такой задачи нет, то соревнование для участника **завершается** и его результатом объявляется сумма баллов за все сданные задачи. В частности, если участник сдаёт первую задачу, то соревнование для него завершается.

Проход очень хочет занять первое место, поэтому он тщательно готовился к этой олимпиаде, он прорешал все существующие задачи и теперь может сдать любую задачу, которую жюри могут выбрать на олимпиаду. Перед олимпиадой он задумался, сколько максимум баллов  $S$  он сможет получить в зависимости от выбора жюри.

К сожалению, Проход очень устал от решения задач, помогите ему — для каждого выбора жюри ответьте, какой максимальный балл  $S$  сможет получить Проход.

### Формат входных данных

Каждый тест состоит из нескольких наборов входных данных — выборов жюри. В первой строке находится одно целое число  $t$  ( $1 \leq t \leq 10^5$ ) — количество наборов входных данных. Далее следует описание наборов входных данных.

Первая строка каждого набора входных данных содержит одно целое число  $n$  ( $1 \leq n \leq 4 \cdot 10^5$ ) — число задач, взятых на олимпиаду.

Вторая строка каждого набора входных данных содержит  $n$  целых чисел  $a_1, a_2, \dots, a_n$  ( $1 \leq a_i \leq 10^9$ ) — стоимости задач.

Третья строка каждого набора входных данных содержит  $n$  целых чисел  $b_1, b_2, \dots, b_n$  ( $1 \leq b_i \leq n$ ) — параметр задачи.

Гарантируется, что сумма значений  $n$  по всем наборам входных данных не превосходит  $4 \cdot 10^5$

### Формат выходных данных

Для каждого набора входных данных выведите единственное число  $S$  — максимальное число баллов, которое может набрать Проход.

## Пример

стандартный ввод	стандартный вывод
2	15
2	700
15 15	
2 1	
4	
100 200 300 400	
3 4 4 1	

## Замечание

В первом случае Прохор может скипнуть первую задачу и система выдаст ему вторую задачу, он может сдать её и получить 15 баллов, далее не выданных задач не остается, соревнование для Прохора завершается с  $S = 15$ . Либо Прохор может сдать первую задачу, тоже получить 15 баллов и соревнование завершится.

Во втором случае Прохор пропускает первую, переходит к третьей (она ранее не выдавалась), сдаёт её и получает 300 баллов, так как вторая задача ранее не выдавалась он переходит к ней, пропускает её и переходит к четвёртой (она ранее не выдавалась), сдаёт четвёртую и получает ещё 400 баллов, не выданных задач не осталось, соревнование для Прохора завершается с  $S = 700$ .

## Задача В. Отгадай строку

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

Вася решил сыграть с вами в игру. Он загадывает строку, состоящую из первых  $k$  строчных букв латинского алфавита, после чего выписывает все её непустые суффиксы и сортирует их в лексикографическом<sup>†</sup> порядке. Далее он сообщает вам получившийся порядок суффиксов, где суффиксу соответствует позиция в строке, с которой он начинается. Также он сообщает вам, сколько каждой буквы было в исходной строке.

Ваша задача состоит в том, чтобы отгадать загаданную строку.

<sup>†</sup>Строка  $s$  лексикографически меньше, чем строка  $t$ , если есть индекс  $i$  такой, что  $s_i < t_i$  (в алфавитном порядке), и  $s_j = t_j$  для всех  $j < i$ . Иными словами, для первого такого индекса  $i$ , где строки различны,  $s_i < t_i$ . Если такого индекса не существует, то лексикографически меньшей считается строка меньшей длины.

### Формат входных данных

Каждый тест состоит из нескольких наборов входных данных. В первой строке дано целое число  $t$ , ( $1 \leq t \leq 10\,000$ ) — количество наборов входных данных. Далее даны  $t$  наборов входных данных друг за другом, в следующем формате:

В первой строке дано два целых числа  $n, k$  ( $1 \leq n \leq 200\,000, 1 \leq k \leq 26$ ) — размер строки и количество различных букв в строке. Гарантируется, что загаданная строка содержит только первые  $k$  букв латинского алфавита.

Во второй строке дано  $n$  целых положительных чисел  $a_i$  ( $1 \leq a_i \leq n$ ) — индексы суффиксов в порядке после лексикографической сортировки. Гарантируется, что индекс каждого суффикса встречается ровно один раз.

В третьей строке дано  $k$  целых  $c_1, c_2, \dots, c_k$  ( $0 \leq c_i \leq n$ ) — количество букв  $a$  в строке, количество букв  $b$  в строке и так далее в алфавитном порядке.

Гарантируется, что  $c_1 + c_2 + \dots + c_k = n$ .

Гарантируется, что сумма  $n$  по всем тестам не превышает 200 000.

### Формат выходных данных

Выведите строку, которую загадал Вася. Гарантируется, что существует ровно одна строка, которая удовлетворяет данным условиям.

### Пример

стандартный ввод	стандартный вывод
4	acab
4 3	bba
3 1 4 2	a
2 1 1	abababbabb
3 2	
3 2 1	
1 2	
1 1	
1	
1	
10 2	
1 3 8 5 10 2 7 4 9 6	
4 6	

## Задача С. Холмы и ямы

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	3 секунды
Ограничение по памяти:	512 мегабайт

В пустынном городе с холмистым ландшафтом мэрия решила выровнять поверхность дороги, закупив самосвал. Дорога разделена на  $n$  участков, пронумерованных от 1 до  $n$  слева направо. Высота поверхности на  $i$ -м участке равна  $a_i$ . Если высота  $i$ -го участка больше 0, то самосвал должен забрать песок с  $i$ -го участка дороги, а если высота  $i$ -го участка меньше 0, то самосвал должен засыпать песком яму на  $i$ -м участке дороги. Гарантируется, что изначальные высоты не равны 0.

Когда самосвал находится на  $i$ -м участке дороги, он может либо забрать оттуда  $x$  единиц песка, и тогда высота поверхности на  $i$ -м участке уменьшится на  $x$ , либо засыпать туда  $x$  единиц песка (при условии, что у него в кузове сейчас есть хотя бы  $x$  единиц песка), и тогда высота поверхности на  $i$ -м участке дороги увеличится на  $x$ .

Самосвал может начать свой путь на любом участке дороги. Перемещение на соседний слева или справа участок дороги занимает 1 минуту, при этом временем загрузки и разгрузки песка можно пренебречь. Кузов самосвала имеет бесконечную вместимость и изначально пуст.

Вам нужно найти минимальное время, за которое самосвал сможет выровнять песок так, чтобы высота на каждом участке стала равна 0. Обратите внимание, что после всех передвижений в кузове самосвала **может остаться песок**. Вам нужно решить эту задачу **независимо**, если оставить только участки с номерами от  $l_i$  до  $r_i$ . При этом использовать песок вне отрезка нельзя.

### Формат входных данных

Каждый тест состоит из нескольких наборов входных данных. Первая строка содержит единственное целое число  $t$  ( $1 \leq t \leq 10^4$ ) — количество наборов входных данных. Далее следует описание наборов входных данных.

Первая строка каждого набора входных данных содержит два целых числа  $n$  и  $q$  ( $1 \leq n, q \leq 3 \cdot 10^5$ ) — количество участков и количество запросов.

Вторая строка каждого набора входных данных содержит  $n$  целых чисел  $a_1, a_2, \dots, a_n$  ( $-10^9 \leq a_i \leq 10^9$ ,  $a_i \neq 0$ ) — изначальная высота на каждом участке.

$i$ -я из следующих  $q$  строк содержит два целых числа  $l_i$  и  $r_i$  ( $1 \leq l_i \leq r_i \leq n$ ) — границы отрезка участков, для которого нужно определить минимальное время.

Гарантируется, что сумма  $n$  по всем наборам входных данных и сумма  $q$  по всем наборам входных данных не превосходит  $3 \cdot 10^5$ .

### Формат выходных данных

Для каждого запроса выведите минимальное время в минутах, необходимое, чтобы выровнять песок на отрезке  $[l_i, r_i]$ , или  $-1$ , если это невозможно.

## Пример

стандартный ввод	стандартный вывод
5	-1
1 1	2
-179	5
1 1	-1
5 3	8
-2 2 -1 3 -1	6
2 4	6
1 5	-1
1 3	2
7 1	
1 1 1 -4 1 1 1	
1 7	
7 2	
2 -2 2 -2 1 2 -1	
1 7	
2 7	
3 2	
-1000000000 1 1000000000	
1 2	
1 3	

## Замечание

В первом наборе входных данных нужно добавить 179 единиц песка на единственном участке. Но забрать их неоткуда, поэтому это невозможно.

Во втором наборе входных данных:

- В первом запросе самосвал может начать путь на втором участке. Он может забрать 2 единицы песка, после чего на втором участке высота станет равна 0. Затем самосвал может переехать на третий участок. Он может высыпать туда 1 единицу песка, после чего на третьем участке высота станет равна 0. Затем самосвал может переехать на четвёртый участок. Там он может забрать 3 единицы песка, после чего на четвёртом участке высота станет равна 0. Суммарно самосвал потратит на перемещения 2 минуты.
- Во втором запросе самосвал может начать путь на четвёртом участке. Он может забрать 3 единицы песка, после чего на четвёртом участке высота станет равна 0. Затем самосвал может переехать на пятый участок. Он может высыпать туда 1 единицу песка, после чего на пятом участке высота станет равна 0. Затем самосвал может переехать на четвёртый участок, а потом на третий. Он может высыпать 1 единицу песка, после чего на третьем участке высота станет равна 0. Затем самосвал может переехать на второй участок. Он может забрать 2 единицы песка. Затем он может переехать на первый участок. Он может высыпать 2 единицы песка, после чего на первом участке высота станет равна 0. Суммарно самосвал потратит на перемещения 5 минут.
- В третьем запросе самосвал не сможет сделать высоту на каждом участке равной 0.

## Задача D. Хорошие раскраски 6

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	1.5 секунд
Ограничение по памяти:	512 мегабайт

Нокотан большой фанат раскрашивания табличек. Она придумала число  $c$  ( $1 \leq c \leq 17$ ) и решила, что перед входом в здание клуба должна висеть таблица со следующими свойствами:

- Высота таблицы  $h$  и ширина таблицы  $w$  удовлетворяют ограничениям ( $1 \leq h, w \leq 1500$ ).
- Каждая клетка может быть раскрашена в какое-то **подмножество** из  $c$  цветов (возможно пустое). Формально пусть  $a_{i,j}$  — число, записанное в таблице в клетке с координатами  $i, j$ , тогда  $0 \leq a_{i,j} < 2^c$  и считается, что клетка покрашена в том числе в цвет с номером  $k$  ( $0 \leq k < c$ ), если  $\lfloor \frac{a_{i,j}}{2^k} \rfloor \bmod 2 = 1$ .
- Пусть  $1 \leq x < 2^c$  и  $cnt_x$  — число клеток, на которых записано число  $x$ . Тогда должно выполняться неравенство  $1 \leq cnt_x \leq 20$ .
- Каждый цвет образует одну компоненту связности. Формально для любого  $0 \leq k < c$  клетки, в которых  $k$ -й бит записи ненулевой ( $\lfloor \frac{a_{i,j}}{2^k} \rfloor \bmod 2 = 1$ ), образуют одну компоненту связности. (Две клетки находятся в одной компоненте связности, если от одной из них можно добраться до второй, переходя в соседние четыре по горизонтали или вертикали клетки множества).

Помогите Нокотан найти таблицу, удовлетворяющую всем свойствам для заданного  $c$ .

### Формат входных данных

В единственной строке входных данных содержится целое число  $c$  ( $1 \leq c \leq 17$ ) — количество цветов, в которые должна быть раскрашена таблица.

### Формат выходных данных

В первой строке выведите пару целых чисел  $h, w$  ( $1 \leq h, w \leq 1500$ ) — высота и ширина таблицы соответственно. В следующих  $h$  строках выведите по  $w$  целых чисел  $a_{i,j}$  ( $0 \leq a_{i,j} < 2^c$ ) — значения в таблице. Таблица должна удовлетворять четырем условиям, описанным в условии задачи.

### Примеры

стандартный ввод	стандартный вывод
1	1 3 1 0 0
2	3 1 1 3 2
3	2 6 7 7 7 7 7 7 1 2 3 4 5 6

### Замечание

Во втором тестовом примере клетки, покрашенные в цвет с номером 0, имеют координаты  $(0, 0)$  и  $(1, 0)$  и поэтому образуют 1 компоненту связности. Клетки, покрашенные в цвет с номером 1, имеют координаты  $(1, 0)$  и  $(2, 0)$  и поэтому также образуют 1 компоненту связности. Несложно увидеть, что все остальные условия тоже выполнены.

В третьем тестовом примере количество клеток с каждым значением, кроме 0 и 7, равняется 1. Количество клеток с 7 равняется 6, что подходит под интервал  $[1, 20]$ , и на количество клеток с 0 в условии не накладывает ограничений. Каждый цвет образует 1 компоненту связности, так как все клетки инцидентны к первому ряду, который связан по всем цветам.

## Задача Е. Покупка колы

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

Есть вендинговый аппарат, в котором продаётся кола. Всего в аппарате есть  $n$  ячеек. Вы знаете, что изначально в ячейке  $i$  находятся  $a_i$  банок колы. Также в аппарате есть  $n$  кнопок, каждая кнопка соответствует какой-то ячейке, при этом каждой ячейке соответствует ровно одна кнопка. К сожалению, обозначения на кнопках стёрлись, поэтому вы **не знаете**, какая кнопка соответствует какой ячейке.

Когда вы нажимаете на кнопку, соответствующую ячейке  $i$ , из  $i$ -й ячейки выпадает одна банка колы, если они там ещё остались. После нажатия банка выпадает так быстро, что отследить, из какой именно ячейки она выпала, невозможно. Содержимое ячеек закрыто от ваших глаз, поэтому вы не можете в реальном времени видеть, сколько банок осталось в каждой из ячеек, единственное, что вам известно, это изначально количества банок в ячейках:  $a_1, a_2, \dots, a_n$ .

После каждого нажатия вы узнаете, выпала ли вам банка колы из ячейки, и забираете её себе, если да.

Какое минимальное количество нажатий на кнопки нужно произвести, чтобы гарантированно получить хотя бы  $k$  банок колы? Вы можете адаптировать свою стратегию по ходу нажатий, основываясь на том, выпала вам банка или нет. Гарантируется, что всего в аппарате есть хотя бы  $k$  банок колы, иными словами,  $k \leq a_1 + a_2 + \dots + a_n$ .

### Формат входных данных

Каждый тест состоит из нескольких наборов входных данных. Первая строка содержит целое число  $t$  ( $1 \leq t \leq 10^4$ ) — количество наборов входных данных. Далее следует описание наборов входных данных.

Первая строка каждого набора входных данных содержит два целых числа  $n$  и  $k$  ( $1 \leq n \leq 2 \cdot 10^5$ ,  $1 \leq k \leq 10^9$ ) — количество ячеек в аппарате и необходимое количество банок колы.

Вторая строка каждого набора входных данных содержит  $n$  целых чисел  $a_1, a_2, \dots, a_n$  ( $1 \leq a_i \leq 10^9$ ) — количества банок в ячейках.

Гарантируется, что  $k \leq a_1 + a_2 + \dots + a_n$ , то есть всего в аппарате есть хотя бы  $k$  банок колы.

Гарантируется, что сумма значений  $n$  по всем наборам входных данных не превосходит  $2 \cdot 10^5$ .

### Формат выходных данных

Для каждого набора входных данных выведите одно целое число — минимальное количество нажатий на кнопки, которое нужно сделать, чтобы гарантированно получить хотя бы  $k$  банок колы.

### Пример

стандартный ввод	стандартный вывод
3	5
3 4	53
2 1 3	1000000000
10 50	
1 1 3 8 8 9 12 13 27 27	
2 1000000000	
1000000000 500000000	

### Замечание

В первом наборе входных данных одна из оптимальных стратегий такова:

Нажимаем на первую кнопку дважды. Первое нажатие гарантированно выдаст банку колы. Далее есть два варианта:

- Если второе нажатие не выдало банку колы, мы знаем, что эта кнопка обязательно соответствует второй ячейке, так как  $a_2 = 1$  и  $a_1 > 1$ ,  $a_3 > 1$ . Тогда мы можем нажать на вторую

кнопку два раза, а на третью кнопку один раз, и так как  $a_1 \geq 2$  и  $a_3 \geq 2$ : мы гарантированно получим три банки колы за эти три нажатия. Итого за 5 нажатий мы получим 4 банки колы.

- Если второе нажатие выдало банку колы, мы можем сделать одно нажатие на вторую кнопку и одно нажатие на третью кнопку. Каждое из этих нажатий гарантированно выдаст нам банку колы. Итого за 4 нажатия мы получим 4 банки колы.

Можно показать, что за 4 нажатия невозможно гарантировать получение 4 банок колы, поэтому ответ 5.



## Задача F. НВПБП

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

*Подпоследовательность* — это последовательность, которую можно получить из массива путем удаления некоторых элементов, не меняя порядок оставшихся элементов. Например, последовательность  $[1, 3, 6]$  — это подпоследовательность массива  $[1, 2, 3, 6, 4, 5]$ . Длиной подпоследовательности называют количество элементов в ней.

*Наибольшая возрастающая подпоследовательность (НВП)* — это подпоследовательность массива наибольшей длины, каждый следующий элемент которой **строго** больше предыдущего.

Для массива  $a$  длины  $n$  определим  $\overline{a_{[l,r]}}$  как  $a_1, \dots, a_{l-1}, a_{r+1}, \dots, a_n$ , то есть весь массив за **исключением** подотрезка  $[l, r]$ .

*Наибольшая возрастающая подпоследовательность без подотрезка  $[l, r]$  (НВПБП)* — наибольшая возрастающая подпоследовательность массива  $\overline{a_{[l,r]}}$ . Обозначим длину этой НВПБП за  $f(\overline{a_{[l,r]}})$ .

Дан массив  $a$  длины  $n$ . Пусть длина его НВП равна  $f(a)$ . Необходимо ответить на  $q$  запросов. Каждый запрос задается парой чисел  $l, r$ . В ответ на запрос необходимо вывести, выполняется ли равенство  $f(a) = f(\overline{a_{[l,r]}})$ .

### Формат входных данных

В первой строке ввода находятся два целых числа  $n, q$  — длина массива  $a$  и число запросов. ( $1 \leq n, q \leq 4 \cdot 10^5$ ).

Во второй строке ввода находятся  $n$  целых чисел — массив  $a$ . ( $1 \leq a_i \leq 10^9$ ).

В следующих  $q$  строках находятся по два целых числа  $l_i$  и  $r_i$  — запросы. ( $1 \leq l_i \leq r_i \leq n$ ).

### Формат выходных данных

Выведите  $q$  строк. В  $i$ -й строке выведите «YES» (без кавычек), если  $f(a) = f(\overline{a_{[l_i, r_i]}})$ , и «NO» иначе.

### Пример

стандартный ввод	стандартный вывод
7 5	YES
1 2 5 4 7 3 6	YES
6 7	YES
4 6	YES
3 3	NO
3 5	
4 7	

### Замечание

НВП массива  $[1, 2, 5, 4, 7, 3, 6]$  равна  $[1, 2, 5, 7]$ , то есть длина равна 4.

В первом запросе  $\overline{a_{[6,7]}} = [1, 2, 5, 4, 7]$ . Во втором запросе  $\overline{a_{[4,6]}} = [1, 2, 5, 6]$ . Очевидно, что  $f(a) = f(\overline{a_{[6,7]}}) = f(\overline{a_{[4,6]}}) = 4$ .

## Задача G. Склеивание массивов

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

Даны  $n$  массивов  $a_1, \dots, a_n$ . Длина каждого массива равна двум. Вам надо склеить массивы в один массив длины  $2n$  так, чтобы количество *инверсий*<sup>1</sup> в получившемся массиве было как можно меньше. Более формально, вам надо выбрать перестановку  $p$  чисел от 1 до  $n$ , чтобы массив  $[a_{p_1,1}, a_{p_1,2}, \dots, a_{p_n,1}, a_{p_n,2}]$  содержал как можно меньше инверсий.

<sup>1</sup> Количество *инверсий* в массиве  $b$  — это количество пар индексов  $i$  и  $j$ , таких что  $i < j$  и  $b_i > b_j$ .

### Формат входных данных

Каждый тест состоит из нескольких наборов входных данных. Первая строка содержит единственное целое число  $t$  ( $1 \leq t \leq 10^4$ ) — количество наборов входных данных. Далее следует описание наборов входных данных.

Первая строка каждого набора входных данных содержит единственное целое число  $n$  ( $1 \leq n \leq 100\,000$ ) — количество массивов.

Каждая из следующих  $n$  строк содержит два целых числа  $a_{i,1}$  и  $a_{i,2}$  ( $1 \leq a_{i,j} \leq 10^9$ ) — элементы  $i$ -го массива.

Гарантируется, что сумма  $n$  по всем наборам входных данных не превосходит  $10^5$ .

### Формат выходных данных

Для каждого набора входных данных выведите  $2n$  целых чисел — элементы полученного вами массива. Если существует несколько решений, выведите любое из них.

### Примеры

стандартный ввод	стандартный вывод
1 3 3 2 4 3 2 1	2 1 3 2 4 3
1 5 5 10 2 3 9 6 4 1 8 7	4 1 2 3 5 10 8 7 9 6

## Задача Н. Ч+К+С

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

Вам дано два сильносвязных<sup>†</sup> ориентированных графа, в каждом из которых ровно  $n$  вершин, но может быть разное количество ребер. Посмотрев на них внимательно, вы заметили важную особенность — длина любого цикла в этих графах делится на  $k$ .

Отнесем все  $2n$  вершин к одному из двух классов: **входящие** или **исходящие**. Для каждой вершины вам известен её класс.

Перед вами стоит следующая задача: определить, можно ли провести ровно  $n$  ориентированных ребер между исходными графами так, чтобы выполнялись следующие четыре условия:

- Концы любого добавленного ребра лежат в разных графах.
- Из каждой **исходящей** вершины исходило ровно одно добавленное ребро.
- В каждую **входящую** вершину входило ровно одно добавленное ребро.
- В получившемся графе длина любого цикла делится на  $k$ .

<sup>†</sup>Сильносвязный граф — это граф, в котором из каждой вершины можно добраться до любой другой.

### Формат входных данных

Каждый тест состоит из нескольких наборов входных данных. Первая строка содержит одно целое число  $t$  ( $1 \leq t \leq 10^4$ ) — количество наборов входных данных. Далее следует описание наборов входных данных.

Первая строка каждого набора входных данных содержит два целых числа  $n, k$  ( $2 \leq k \leq n \leq 2 \cdot 10^5$ ) — количество вершин в каждом из графов и значение, на которое делится длина каждого цикла.

Вторая строка каждого набора входных данных содержит последовательность из  $n$  чисел, каждое из которых 0, либо 1 — классы вершин первого графа. Если  $i$ -е число в последовательности равно 0, значит,  $i$ -я вершина входящая, иначе эта вершина исходящая.

Следующая строка содержит единственное число  $m_1$  ( $1 \leq m_1 \leq 5 \cdot 10^5$ ) — количество ребер в первом графе.

Следующие  $m_1$  строк содержат описания ребер графа. В  $i$ -й строке вводится пара чисел  $v_i, u_i$  ( $1 \leq v_i, u_i \leq n$ ) — ребро в первом графе, ведущее из  $v_i$  в  $u_i$ .

Далее в таком же формате вводятся классы вершин второго графа, количество ребер второго графа  $m_2$  и сами ребра второго графа.

Гарантируется, что сумма  $n$  и сумма  $k$  не превосходит  $2 \cdot 10^5$  каждая по всем наборам входных данных, а сумма  $m_1$  и сумма  $m_2$  не превосходит  $5 \cdot 10^5$  каждая по всем наборам входных данных.

### Формат выходных данных

Для каждого набора входных данных выведите "YES" (без кавычек), если можно корректно провести  $n$  новых ребер и "NO" иначе.

## Пример

стандартный ввод	стандартный вывод
3	YES
4 2	NO
1 0 0 1	YES
4	
1 2	
2 3	
3 4	
4 1	
1 0 0 1	
4	
1 3	
3 2	
2 4	
4 1	
4 2	
1 1 0 1	
4	
1 2	
2 3	
3 4	
4 1	
1 0 1 1	
4	
1 3	
3 2	
2 4	
4 1	
4 2	
1 1 1 1	
4	
1 2	
2 3	
3 4	
4 1	
0 0 0 0	
6	
1 2	
2 1	
1 3	
3 1	
1 4	
4 1	

## Замечание

В первом тестовом случае можно провести из первого графа во второй граф ребра (1, 3), (4, 2) и из второго в первый ребра (1, 2), (4, 3).

Во втором тестовом случае можно как угодно сопоставить вершинам из первого графа вершины второго графа.

## Задача I. Простая задача для любителей

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

На бесконечной плоскости заданы  $n$  секторов, каждый сектор представляет собой направленный угол  $AOB$ , где  $O$  — начало координат, направление угла считается против часовой стрелки от луча  $OA$  к лучу  $OB$ . Величина угла не превосходит  $180^\circ$ .

Даны  $q$  точек на плоскости. Каждая точка имеет свой вес  $w_i$ .

Для каждого сектора необходимо найти суммарный вес точек, принадлежащих данному сектору. Точка принадлежит сектору, если она находится внутри или на его границе. Обратите внимание, что одна точка может принадлежать нескольким секторам.

### Формат входных данных

Первая строка содержит одно целое число  $n$  ( $1 \leq n \leq 2 \cdot 10^5$ ) — количество секторов. В следующих  $n$  строках описаны секторы.

Каждый сектор задается четырьмя целыми числами  $x_A, y_A, x_B$  и  $y_B$  ( $0 \leq |x_A|, |y_A|, |x_B|, |y_B| \leq 10^9$ ) — координаты точек  $A$  и  $B$ . Гарантируется, что точки  $A$  и  $B$  не совпадают с началом координат, и величина направленного угла лежит в диапазоне  $(0^\circ; 180^\circ]$ .

Следующая строка содержит одно целое число  $q$  ( $1 \leq q \leq 2 \cdot 10^5$ ) — количество точек. В следующих  $q$  строках описаны точки.

Каждая точка задается тремя целыми числами  $x, y$  и  $w$  ( $0 \leq |x|, |y| \leq 10^9, 0 \leq w \leq 10^9$ ) — координаты и вес. Гарантируется, что точка  $(x, y)$  не совпадает с началом координат.

### Формат выходных данных

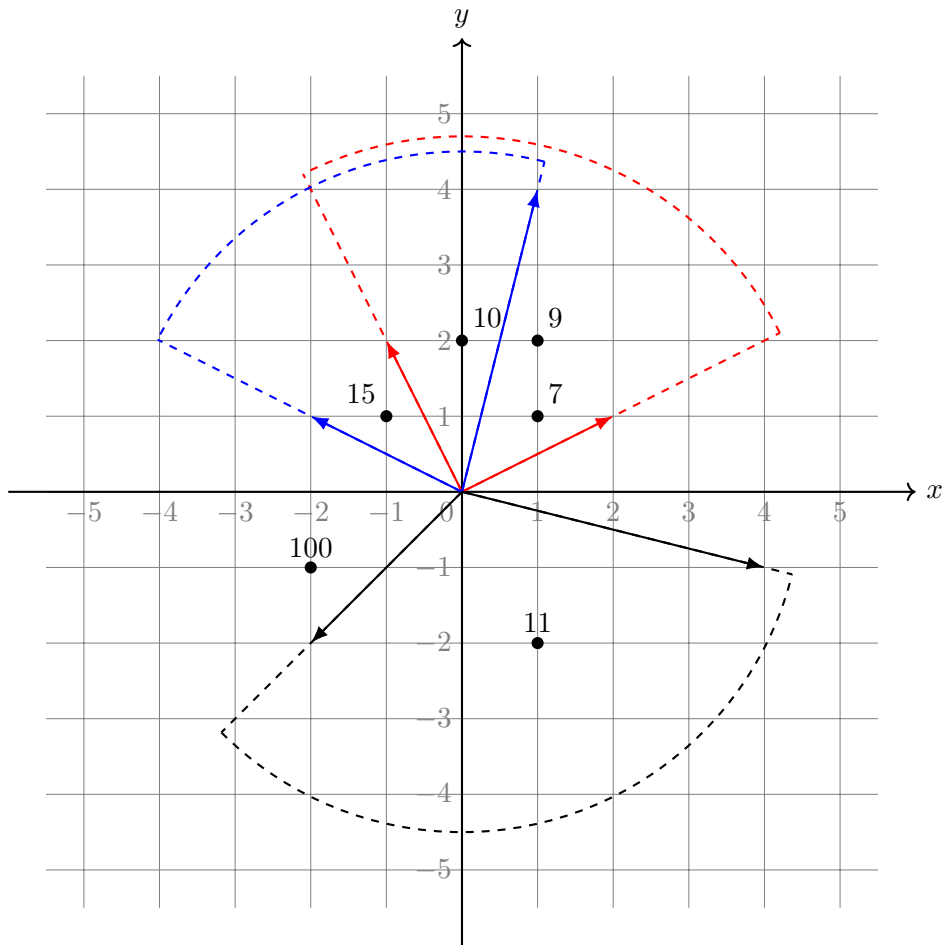
Выведите  $n$  целых чисел  $s_1, s_2, \dots, s_n$ , где  $s_i$  — суммарный вес точек, принадлежащих  $i$ -му сектору.

### Пример

стандартный ввод	стандартный вывод
3	26 25 11
2 1 -1 2	
1 4 -2 1	
-2 -2 4 -1	
6	
0 2 10	
1 2 9	
-1 1 15	
1 1 7	
-2 -1 100	
1 -2 11	

## Замечание

На рисунке изображен пример из условия. Рядом с каждой точкой подписан её вес.



## Задача J. Много игр

Имя входного файла: стандартный ввод  
Имя выходного файла: стандартный вывод  
Ограничение по времени: 1.5 секунд  
Ограничение по памяти: 256 мегабайт

Недавно вы получили редчайший билет в единственное казино в мире, в котором и правда можно что-то заработать, и вы хотите на полную воспользоваться этой возможностью.

Условия в этом казино следующие:

- В казино есть  $n$  игр, в которые предлагается сыграть.
- В каждую из игр можно сыграть не более одного раза.
- Каждая игра характеризуется двумя параметрами:  $p_i$  ( $1 \leq p_i \leq 100$ ) и  $w_i$  — вероятность победы в игре в процентах и выигрыш за победу.
- Если вы проиграете хоть в одну игру, в которую решите сыграть, — вы не получите вообще ничего (даже за выигранные игры).

Вам нужно заранее выбрать набор игр, в которые вы будете играть, таким образом, чтобы максимизировать **матожидание** выигрыша в казино.

В данном случае, если вы выберете игры с индексами  $i_1, \dots, i_k$ , то вы выиграете во все из них с вероятностью  $\prod_{j=1}^k \frac{p_{i_j}}{100}$ , и в таком случае получите  $\sum_{j=1}^k w_{i_j}$  выигрыша.

То есть **матожидание** вашего выигрыша будет  $\left( \prod_{j=1}^k \frac{p_{i_j}}{100} \right) \cdot \left( \sum_{j=1}^k w_{i_j} \right)$ .

Чтобы не разориться, владельцы казино поставили ограничение матожидания выигрыша каждой отдельной игры, так что для любого  $i$  выполнено  $w_i \cdot p_i \leq 200\,000$ .

Ваша задача — найти максимальное матожидание выигрыша, которое можно получить, выбрав какое-то подмножество игр в казино.

### Формат входных данных

В первой строке входных данных дано одно целое число  $n$  ( $1 \leq n \leq 200\,000$ ) — количество игр, в которые предлагается сыграть.

В следующих  $n$  строках находятся по два целых числа  $p_i, w_i$  ( $1 \leq p_i \leq 100$ ,  $1 \leq w_i$ ,  $p_i \cdot w_i \leq 200\,000$ ) — вероятность победы и размер выигрыша в  $i$ -й игре.

### Формат выходных данных

Выведите одно число — максимальное матожидание выигрыша в казино, которое можно получить, выбрав некоторое подмножество игр.

Ответ будет принят, если его абсолютная или относительная погрешность не превосходит  $10^{-6}$ .

### Пример

стандартный ввод	стандартный вывод
3 80 80 70 100 50 200	112.00000000

### Замечание

В первом примере нужно выбрать первую и третью игры. В таком случае матожидание выигрыша будет  $\left( \frac{80}{100} \cdot \frac{50}{100} \right) \cdot (80 + 200) = 112$ .

## Задача К. Древо жизни

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

В сердце древнего королевства растет легендарное Древо Жизни — единственное в своем роде и источник магической силы всего мира. Дерево состоит из  $n$  узлов. Каждый узел этого дерева — это волшебный источник, связанный с другими такими же источниками через магические каналы (ребра). Всего в дереве  $n - 1$  каналов,  $i$ -й из которых соединяет узлы  $v_i$  и  $u_i$ . Также между любыми двумя узлами в дереве существует единственный простой путь по каналам.

Однако магическая энергия, текущая через эти каналы, должна быть сбалансирована, иначе мощь Древа Жизни может нарушить естественный порядок и вызвать катастрофические последствия. Мудрецы королевства обнаружили, что когда два магических канала сходятся в одном узле, между ними возникает опасная «магическая резонансная вибрация». Чтобы защитить Древо Жизни и сохранить его баланс, необходимо выбрать несколько путей и провести через них специальные ритуалы. Путь — это последовательность различных узлов  $v_1, v_2, \dots, v_k$  для натурального  $k$ , где каждая пара соседних узлов  $v_i$  и  $v_{i+1}$  соединена каналом (для любого  $1 \leq i \leq k - 1$ ). При проведении мудрецами ритуала через такой путь блокируются все резонансные вибрации между парами соседних каналов в пути, а именно между каналами  $(v_i, v_{i+1})$  и  $(v_{i+1}, v_{i+2})$  для каждого  $1 \leq i \leq k - 2$ .

Задача мудрецов: выбрать минимальное количество путей и провести через них ритуалы, чтобы заблокировать все резонансные вибрации. Это означает, что для каждой пары каналов, исходящих из одного узла, должен существовать хотя бы один выбранный путь, который охватывает **оба** этих канала.

Помогите мудрецам найти минимальное количество таких путей, чтобы магический баланс Древа Жизни был сохранен, и его сила продолжала питать весь мир!

### Формат входных данных

Каждый тест состоит из нескольких наборов входных данных. Первая строка содержит одно целое число  $t$  ( $1 \leq t \leq 40\,000$ ) — количество наборов входных данных. Далее следует описание наборов входных данных.

Первая строка каждого набора входных данных содержит одно целое число  $n$  ( $2 \leq n \leq 500\,000$ ) — количество узлов в Древе Жизни.

В  $i$ -й из следующих  $n - 1$  строк каждого набора входных данных содержатся два целых числа  $v_i$  и  $u_i$  ( $1 \leq v_i < u_i \leq n$ ) — канал, соединяющий узлы  $v_i$  и  $u_i$ .

Гарантируется, что между любыми двумя узлами существует единственный простой путь по каналам.

Гарантируется, что сумма  $n$  по всем наборам входных данных не превосходит 500 000.

### Формат выходных данных

Для каждого набора входных данных выведите одно целое число — минимальное количество путей, которое нужно выбрать мудрецам, чтобы предотвратить катастрофу.



## Пример

стандартный ввод	стандартный вывод
4	1
4	0
1 2	7
2 3	3
3 4	
2	
1 2	
8	
3 7	
2 4	
1 2	
2 5	
3 6	
1 3	
3 8	
6	
2 3	
1 2	
3 6	
1 5	
1 4	

## Замечание

В первом тестовом примере достаточно выбрать путь (1, 2, 3, 4), таким образом ответ 1.  
Во втором тестовом примере нет пар каналов, исходящих из одного узла, поэтому ответ 0.

## Задача L. Выгодный процент

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

«Не худший» вклад известного банка, который имеет максимальную ставку в линейке вкладов этого банка, можно открыть на сумму от 100 000 рублей, и только если сумма открываемого вклада будет превышать некоторую величину  $b$ , индивидуально рассчитываемую каждому вкладчику банком, исходя из суммы имевшихся вкладов за последние два месяца.

На текущем счету у Филиппа находятся  $a$  рублей, ему очень нравится «Не худший» вклад, поэтому он хочет его открыть на как можно большую сумму. Однако приложение банка позволяет открыть «Не худший» вклад, только если он будет открыт не менее чем на  $b$  рублей.

По счастливой случайности, Лена заметила, что если открыть любой другой вклад на сумму  $x$  из имеющихся на текущем счету средств, то величина  $b$  уменьшится на  $2x$  или станет равна 100 000, если  $b - 2x < 100\,000$ . При этом закрыть этот другой вклад и вернуть деньги не получится.

Все расчеты этого банка ведутся в целых числах, поэтому нет возможности открывать вклады на нецелое число рублей.

На какую максимальную сумму Филипп сможет открыть «Не худший» вклад, возможно, сначала используя часть денег для открытия других вкладов.

Обратите внимание, на другие вклады нет ограничения по сумме средств, и их можно открывать сколько угодно, но деньги, используемые для их открытия, нельзя будет использовать для открытия «Не худшего» вклада.

### Формат входных данных

В единственной строке входных данных находятся два натуральных числа  $a$  ( $0 < a \leq 1\,000\,000\,000$ ) и  $b$  ( $100\,000 \leq b \leq 1\,000\,000\,000$ ) — сумма денег в рублях на текущем счету и минимальная сумма, на которую банк позволит открыть вклад под максимальный процент, соответственно.

### Формат выходных данных

Выведите одно целое число — максимальную сумму, на которую Филипп сможет открыть «Не худший» вклад, если будет действовать оптимально.

### Примеры

стандартный ввод	стандартный вывод
200000 300000	100000
100000 200000	0

### Замечание

В первом примере Филипп может сначала открыть не самый выгодный вклад на 100 000 рублей, тогда на оставшиеся 100 000 он сможет открыть «Не худший» вклад.

Во втором примере открыть «Не худший» вклад не получится.