Задача A. Ода ветру

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

Лук, украшенный безымянными цветами, несет в себе искренние надежды столь же безымянного человека.

Вы получили элегантный лук, известный как Windblume Ode. В оружие вписан массив из n ($n \geqslant 3$) положительных **попарно различных** целых чисел (иными словами, все числа различны).

Найдите наибольшее по количеству элементов подмножество этого массива, сумма которого является составным числом. Положительное целое число x называется составным, если существует положительное целое число y такое, что $1 < y < x$ и x делится на y.

Если существует несколько подмножеств с таким наибольшим размером с составной суммой, можно вывести любое из них. Можно доказать, что при ограничениях задачи такое непустое подмножество всегда существует.

Формат входных данных

Каждый тест содержит несколько наборов входных данных. Первая строка содержит количество наборов входных данных t ($1 \leqslant t \leqslant 100$). Далее следует описание наборов входных данных.

Первая строка каждого набора входных данных содержит целое число n ($3 \leqslant n \leqslant 100$) — длину массива.

Вторая строка каждого набора входных данных содержит n **попарно различных** целых чисел $a_1, a_2, \ldots, a_n$ ($1 \leqslant a_i \leqslant 200$) — элементы массива.

Формат выходных данных

Для каждого набора входных данных выведите две строки.

Первая строка должна содержать одно целое число x: размер наибольшего подмножества с составной суммой. Следующая строка должна содержать x разделенных пробелами целых чисел, обозначающих позиции элементов выбранного вами подмножества в исходном массиве.

Пример

стандартный ввод	стандартный вывод
4 3 8 1 2 4 6 9 4 2 9 1 2 3 4 5 6 7 8 9 3 200 199 198	2 2 1 4 2 1 4 3 9 6 9 1 2 3 4 5 7 8 3 1 2 3

Замечание

В первом наборе входных данных подмножество $\{a_2, a_1\}$ имеет сумму 9, которая является составным числом. Единственное подмножество размера 3 имеет простую сумму, равную 11. Обратите внимание, что вы также могли выбрать подмножество $\{a_1, a_3\}$ с суммой $8+2=10$, которое является составным, так как кратно 2.

Во втором наборе входных данных сумма всех элементов равна 21, что является составным числом. Здесь мы просто берем весь массив в качестве подмножества.

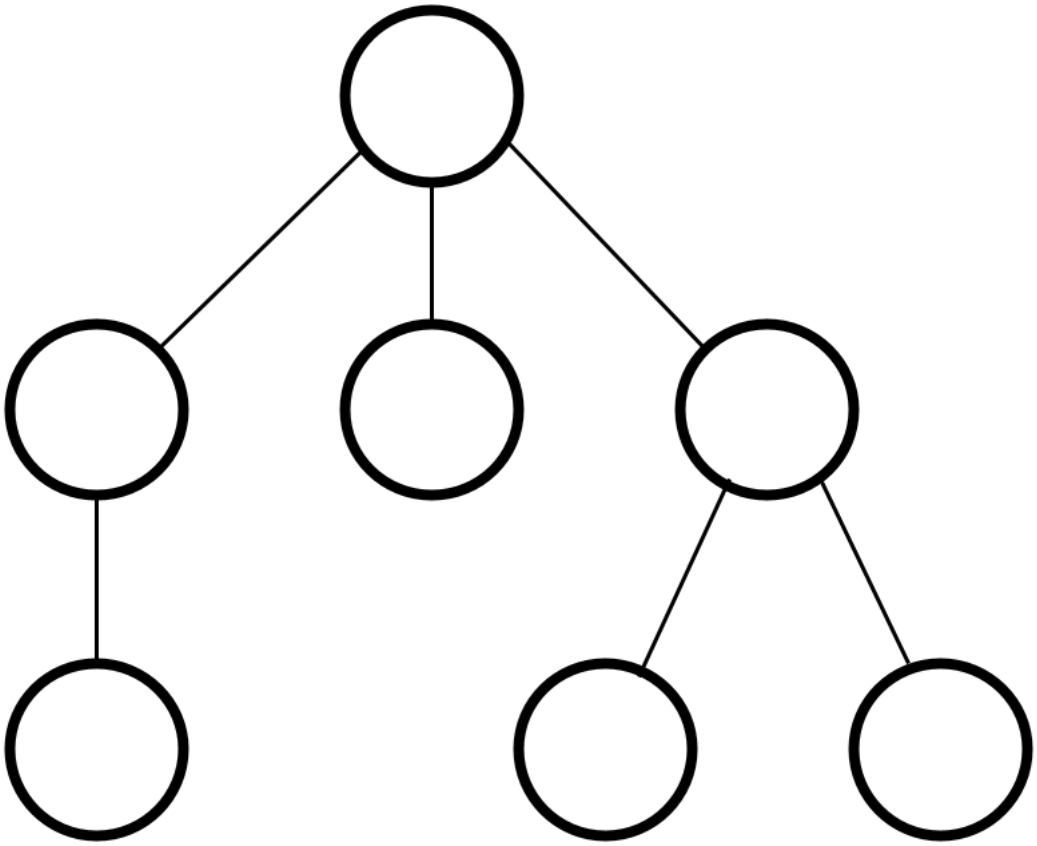
Задача B. Омкар и божественное дерево

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

Лорд Омкар хотел бы, чтобы у него было дерево с n вершинами ($3 \leqslant n \leqslant 10^5$), и попросил своих учеников построить дерево. Однако Лорд Омкар создал m ($\mathbf{1 \leqslant m < n}$) ограничений, чтобы гарантировать, что дерево будет настолько божественным, насколько это возможно.

Дерево на n вершинах — это связный неориентированный граф с n вершинами и $n-1$ ребром. Заметим, что для любых двух вершин существует ровно один простой путь между ними, где простой путь — это путь между двумя вершинами, который не содержит ни одной вершины более одного раза.

Вот пример дерева:



Ограничение состоит из 3 попарно различных целых чисел, a, b и c ($1 \leqslant a, b, c \leqslant n$). Оно означает, что вершина b не может лежать на простом пути между вершинами a и c.

Сможете ли вы помочь Лорду Омкару и стать его самым доверенным учеником? Вам нужно будет найти божественные деревья для нескольких наборов ограничений. Можно показать, что при ограничениях задачи божественное дерево всегда найдется.

Формат входных данных

Каждый тест содержит несколько наборов входных данных. Первая строка содержит количество наборов входных данных t ($1 \leqslant t \leqslant 10^4$). Далее следует описание наборов входных данных.

Первая строка каждого набора входных данных содержит два целых числа n и m ($3 \leqslant n \leqslant 10^5$, $\mathbf{1 \leqslant m < n}$), представляющих размер дерева и количество ограничений.

i-я из следующих m строк содержит три целых числа a_i, b_i, c_i ($1 \leqslant a_i, b_i, c_i \leqslant n$, a, b, c **попарно различны**), обозначающие, что вершина b_i не может лежать на простом пути между вершинами a_i и c_i.

Гарантируется, что сумма n по всем наборам входных данных не превышает 10^5

Формат выходных данных

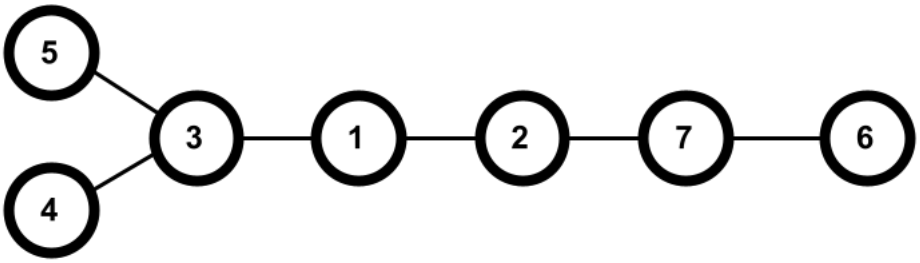
Для каждого набора входных данных выведите $n-1$ строку, обозначающих $n-1$ ребро в дереве. В каждой строке выведите два целых числа u и v ($1 \leqslant u, v \leqslant n$, $u \neq v$), обозначающие, что между вершинами u и v есть ребро. Данные ребра должны образовывать дерево, удовлетворяющее ограничениям Омкара.

Пример

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

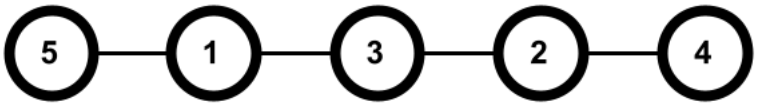
Замечание

Вывод первого набора входных данных соответствует следующему дереву:



Для первого ограничения, простой путь между 1 и 3 — это $1, 3$, и не содержит 2. Простой путь между 3 и 5 — это $3, 5$, который не содержит 4. Простой путь между 5 и 7 — $5, 3, 1, 2, 7$, который не содержит 6. Простой путь между 6 и 4 — $6, 7, 2, 1, 3, 4$, который не содержит 5. Таким образом, это дерево удовлетворяет всем ограничениям.

Вывод второго примера соответствует следующему дереву:



Задача C. Омкар и определение

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

Условие задачи вырисовывается ниже, наполняя вас определенностью.

Рассмотрим таблицу, в которой некоторые клетки пустые, а некоторые заполнены. Назовем клетку в этой таблице **хорошей**, если, начиная с этой клетки, вы можете выйти из таблицы, двигаясь вверх и влево только через пустые клетки. Это касается и стартовой клетки, поэтому все заполненные клетки не являются хорошими. Обратите внимание, вы можете покинуть таблицу из любой крайней левой клетки (в первом столбце), двигаясь влево, а также из любой крайней верхней клетки (в первой строке) — двигаясь вверх.

Назовем таблицу **определяемой**, если, зная только то, какие клетки являются хорошими, а какие — нет, мы можем точно определить, какие клетки заполнены, а какие нет.

Вам дана таблица a размером $n \times m$, т. е. таблица с n строками и m столбцами. Вам нужно ответить на q запросов ($1 \leqslant q \leqslant 2 \cdot 10^5$). Каждый запрос задается двумя целыми числами x_1, x_2 ($1 \leqslant x_1 \leqslant x_2 \leqslant m$) и спрашивает, является ли часть таблицы a, состоящая из столбцов $x_1, x_1 + 1, \ldots, x_2 - 1, x_2$, определяемой.

Формат входных данных

Первая строка содержит два целых числа n, m ($1 \leqslant n, m \leqslant 10^6$, $nm \leqslant 10^6$) — размеры таблицы a.

Далее следуют n строк. В y-й строке содержится m символов, x-й из которых «`X`», если клетка на пересечении y-й строки и x-го столбца заполнена, и «`.`», если она пуста.

Следующая строка содержит одно целое число q ($1 \leqslant q \leqslant 2 \cdot 10^5$) — количество запросов.

Далее следуют строки по q. Каждая строка содержит два целых числа x_1 и x_2 ($1 \leqslant x_1 \leqslant x_2 \leqslant m$), представляющих запрос, спрашивающий, является ли часть таблицы a, содержащая только столбцы $x_1, x_1 + 1, \ldots, x_2 - 1, x_2$, определяемой.

Формат выходных данных

Для каждого запроса выведите одну строку, содержащую «`YES`», если часть таблицы, указанная в запросе, определяемая, и «`NO`» в противном случае. Вывод не чувствителен к регистру (поэтому «`YEs`» и «`No`» также будут приняты).

Пример

стандартный ввод	стандартный вывод
4 5 ..XXX ...X. ...X. ...X. 5 1 3 3 3 4 5 5 5 1 5	YES YES NO YES NO

Замечание

Для каждого запроса из примера соответствующая часть таблицы изображена дважды: сначала в исходном формате, затем с каждой клеткой, помеченной как «`E`», если она хорошая, и «`N`» в противном случае.

Для первого запроса:

```
..X EEN
... EEE
... EEE
... EEE
```

Для второго запроса:

```
X N
. E
. E
. E
```

Обратите внимание, что вы можете выйти из таблицы, пройдя влево из любой крайней левой клетки (или вверх из любой крайней верхней клетки); вам не нужно достигать левой верхней угловой клетки, чтобы выйти из таблицы.

Для третьего запроса:

```
XX NN
X. NN
X. NN
X. NN
```

Эта часть таблицы не может быть определена только по тому, является ли каждая клетка хорошей, потому что таблица, показанная ниже, также производит показанную выше «таблицу выходимости клеток»:

```
XX
XX
XX
XX
```

Для четвертого запроса:

```
X N
. E
. E
. E
```

Для пятого запроса:

```
..XXX EENNN
...X. EEENN
...X. EEENN
...X. EEENN
```

Этот запрос — это просто вся таблица. Она не может быть определена только по тому, является ли каждая клетка хорошей, потому что таблица, показанная ниже, также производит показанную выше «таблицу выходимости клеток»:

```
..XXX
...XX
...XX
...XX
```

Задача D. Омкар и смысл жизни

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

Оказывается, смысл жизни — это перестановка $p_1, p_2, \ldots, p_n$ целых чисел $1, 2, \ldots, n$ ($2 \leqslant n \leqslant 100$). Омкар, создавший все живое, знает эту перестановку и позволит вам выяснить ее с помощью некоторых запросов.

Запрос состоит из массива $a_1, a_2, \ldots, a_n$ целых чисел от 1 до n. a — это **не** обязательно перестановка. Омкар сначала вычислит попарную сумму a и p, то есть вычислит массив s, где $s_j = p_j + a_j$ для всех $j = 1, 2, \ldots, n$. Затем он найдет наименьший индекс k такой, что s_k встречается в s более одного раза, и скажет вам k. Если такого индекса k не существует, то он скажем вам 0.

Можно выполнить не более $2n$ запросов. Выясните смысл жизни p.

Протокол взаимодействия

Начните взаимодействие с чтения одного целого числа n ($2 \leqslant n \leqslant 100$) — длины перестановки p.

Затем вы можете делать запросы. Запрос состоит из одной строки «? a_1 a_2 ... a_n» ($1 \leqslant a_j \leqslant n$).

Ответом на каждый запрос будет одно целое число k, как описано выше ($0 \leqslant k \leqslant n$).

После вывода запроса не забудьте вывести перевод строки и сбросить буфер вывода. В противном случае вы получите вердикт `Решение «зависло»`. Для сброса буфера используйте:

- `fflush(stdout)` или `cout.flush()` в C++;
- `System.out.flush()` в Java;
- `flush(output)` в Pascal;
- `stdout.flush()` в Python;
- смотрите документацию для других языков.

Чтобы вывести ответ, выведите одну строку «! p_1 p_2 ... p_n» затем завершите работу.

Вы можете сделать не более $2n$ запросов. Вывод ответа не считается запросом.

Формат взлома

Для взлома сначала выведите строку, содержащую n ($2 \leqslant n \leqslant 100$), затем выведите другую строку, содержащую скрытую перестановку $p_1, p_2, \ldots, p_n$ чисел от 1 до n.

Пример

стандартный ввод	стандартный вывод
5 2 0 1 	 ? 4 4 2 3 2 ? 3 5 1 5 5 ? 5 2 4 3 1 ! 3 2 1 5 4

Замечание

В примере скрытая перестановка p равна $[3, 2, 1, 5, 4]$. Было сделано три запроса.

Первый запрос — $a = [4, 4, 2, 3, 2]$. Это дает $s = [3 + 4, 2 + 4, 1 + 2, 5 + 3, 4 + 2] = [7, 6, 3, 8, 6]$. 6 — единственное число, которое встречается более одного раза, и впервые оно появляется на индексе 2, что делает ответ на запрос 2.

Второй запрос — $a = [3, 5, 1, 5, 5]$. Это дает $s = [3+3, 2+5, 1+1, 5+5, 4+5] = [6, 7, 2, 10, 9]$. Здесь нет чисел, которые встречаются более одного раза, поэтому ответ на запрос — 0.

Третий запрос — $a = [5, 2, 4, 3, 1]$. Это дает $s = [3+5, 2+2, 1+4, 5+3, 4+1] = [8, 4, 5, 8, 5]$. 5 и 8 встречаются здесь более одного раза. 5 впервые появляется на индексе 3, а 8 впервые появляется на индексе 1, причем $1 < 3$, что делает ответ на запрос 1.

Обратите внимание, что пример приведен только для того, чтобы показать, как работает взаимодействие; не гарантируется, что приведенные выше запросы представляют собой правильную стратегию, с помощью которой можно определить ответ.

Задача E. Момент цветения

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

Она делает все возможное, чтобы безупречно провести последний обряд человека и сохранить баланс инь и ян в мире.

Ху Тао, будучи маленькой проказницей, попыталась напугать вас этой задачей с графом! Вам дан связный неориентированный граф из n вершин с m ребрами. У вас также есть q запросов. Каждый запрос состоит из двух вершин a и b.

Изначально все ребра графа имеют вес 0. Для каждого запроса вы должны выбрать простой путь, начинающийся из a и заканчивающийся в b. Затем к весу каждого ребра вдоль этого пути добавляется 1. Определите, возможно ли, чтобы после обработки всех q запросов все ребра в этом графе имели четный вес. Если да, то выведите выбор путей для каждого запроса.

Если это невозможно, определите наименьшее количество дополнительных запросов, которые можно добавить, чтобы это стало возможным. Можно показать, что при заданных ограничениях это число не превысит 10^{18}.

Простой путь определяется как любой путь, который не посещает вершину более одного раза.

Считается, что ребро имеет четный вес, если его значение кратно 2.

Формат входных данных

Первая строка содержит два целых числа n и m ($2 \leqslant n \leqslant 3 \cdot 10^5$, $n-1 \leqslant m \leqslant \min\left(\frac{n(n-1)}{2}, 3 \cdot 10^5\right)$).

Каждая из следующих m строк содержит два целых числа x и y ($1 \leqslant x, y \leqslant n$, $x \neq y$), указывающих на неориентированное ребро между вершинами x и y. Входные данные не будут содержать петель или дублирующихся ребер, и полученный граф будет связным.

Следующая строка содержит одно целое число q ($1 \leqslant q \leqslant 3 \cdot 10^5$).

Каждая из следующих q строк содержит два целых числа a и b ($1 \leqslant a, b \leqslant n, a \neq b$), описание каждого запроса.

Гарантируется, что $nq \leqslant 3 \cdot 10^5$.

Формат выходных данных

Если можно заставить все веса ребер быть четными, выведите «`YES`» в первой строке, а затем $2q$ строк, указывающих выбор пути для каждого запроса в том же порядке, в котором задаются запросы. Для каждого запроса первая строка должна содержать одно целое число x: количество узлов в выбранном пути. Следующая строка должна содержать x целых чисел p_i, указывающих выбранный путь ($p_1 = a, p_x = b$ и все числа должны лежать между 1 и n). Этот путь не может содержать одну вершину более одного раза и должен быть корректным простым путем в графе.

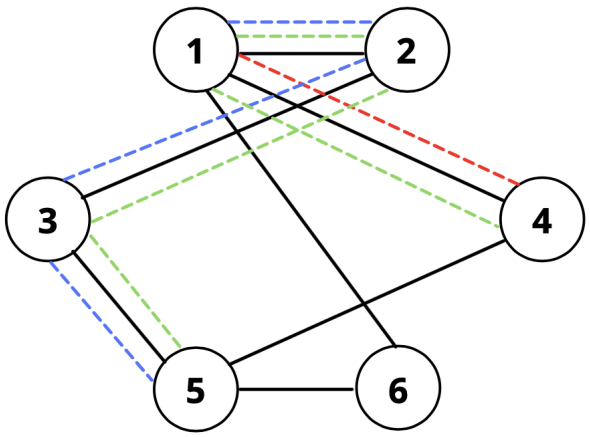
Если невозможно заставить все веса ребер быть четными, выведите «`NO`» в первой строке и минимальное количество запросов, которые нужно добавить, во второй строке.

Примеры

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

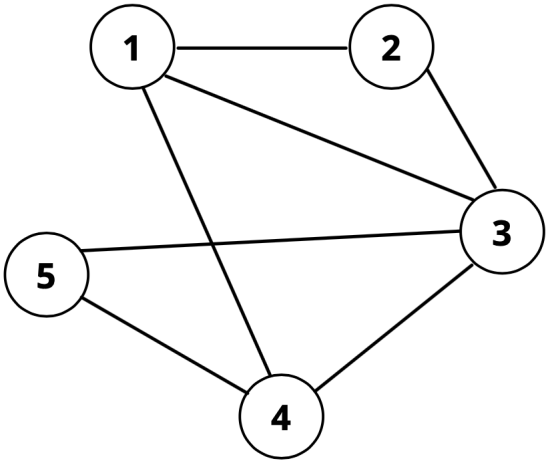
Замечание

Вот как выглядят запросы для первого примера (красный цвет соответствует 1-му запросу, синий — 2-му, а зеленый — 3-му):



Обратите внимание, что каждое ребро в графе входит либо в 0, либо в 2 цветных пути.

Граф во втором примере выглядит следующим образом:



Не существует такого назначения путей, которое заставит все ребра иметь четные веса при заданных запросах. Чтобы получить набор запросов, удовлетворяющих условию, нужно добавить по крайней мере 2 новых запроса.

Задача F. Защитник детских мечт

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

Даже если вы просто оставите их в покое, они разлетятся на куски сами по себе. Значит, кто-то должен их защищать, верно?

Вы снова играете с Тьюсером в городе Лиюэ. Водя эксцентричного малыша по городу, вы заметили кое-что интересное в его структуре.

Лиюэ можно представить в виде направленного графа, содержащего n вершин. Вершины пронумерованы от 1 до n. Ребро из вершины a в вершину b существует тогда и только тогда, когда $a < b$.

Путь между вершинами a и b определяется как последовательность ребер таких, можно начать путь в a, пройти по всем этим ребрам в соответствующем неправлении, и закончить в b. Длина пути равна количеству ребер. Радужный путь длины x определяется как такой путь в графе, что среди этих x ребер существует по крайней мере 2 разных цвета.

Любимое число Тьюсера — k. Вам интересно следующее: если вам нужно обозначить каждое ребро цветом, то какое минимальное количество цветов понадобится для того, чтобы все пути длины k или больше были радужными?

Тьюсер хочет удивить своего старшего брата картой Лиюэ. Он также хочет узнать допустимую раскраску ребер, которая использует минимальное количество цветов. Пожалуйста, помогите ему решить эту задачу!

Формат входных данных

Единственная строка ввода содержит два целых числа n и k ($2 \leqslant k < n \leqslant 1000$).

Формат выходных данных

В первой строке выведите c, минимальное количество цветов, необходимое для выполнения вышеуказанных требований.

Во второй строке выведите допустимую раскраску ребер в виде массива из $\frac{n(n-1)}{2}$ целых чисел от 1 до c. В ней должно присутствовать ровно c различных цветов. Выведите ребра в порядке возрастания сначала по начальному вершине, затем по второй вершине.

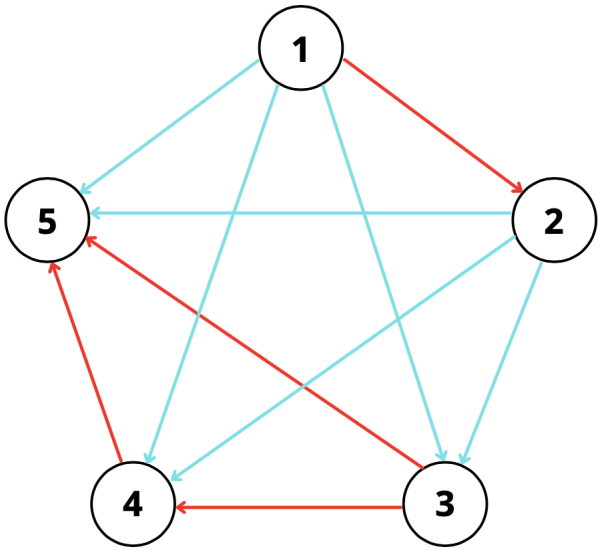
Например, если $n = 4$, то цвета ребер нужно выводить в таком порядке: (1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)

Примеры

стандартный ввод	стандартный вывод
5 3	2 1 2 2 2 2 2 2 1 1 1
5 2	3 3 2 2 1 2 2 1 3 1 1
8 7	2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
3 2	2 1 2 2

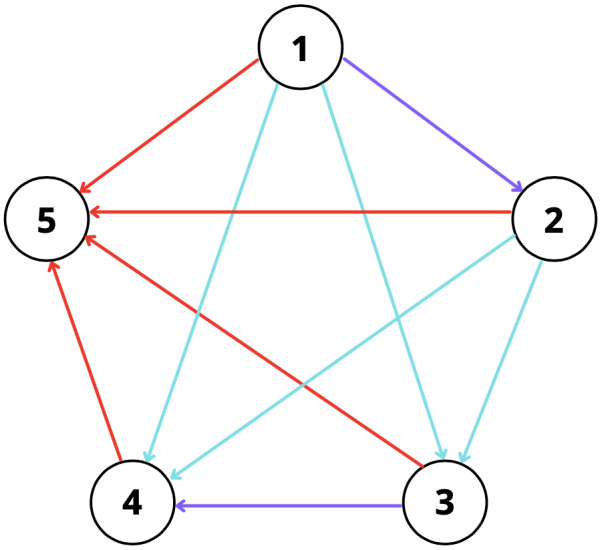
Замечание

Соответствующая конструкция для первого примера выглядит следующим образом:



Невозможно удовлетворить ограничениям, используя менее 2 цветов.

Соответствующая конструкция для второго примера выглядит следующим образом:



Можно показать, что не существует конструкции, использующей менее 3 цветов.

Задача G. Омкар и путешествия во времени

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

El Psy Kongroo.

Омкар смотрит *Steins;Gate*.

В *Steins;Gate*, Окабэ Ринтару нужно выполнить n заданий ($1 \leqslant n \leqslant 2 \cdot 10^5$). К сожалению, он не знает, когда ему нужно выполнить задания.

Изначально время равно 0. Путешествие во времени происходит по следующим правилам:

- Для каждого $k = 1, 2, \ldots, n$, Окабэ поймет в момент времени b_k, что он должен был выполнить k-е задание в момент времени a_k ($a_k < b_k$).
- Когда он осознает это, если k-е задание уже было выполнено в момент времени a_k, Окабэ сохраняет обычный ход времени. В противном случае, он перемещается во времени к моменту a_k.
- Если Окабэ переместится во времени к моменту a_k, то все задания после этого времени снова станут невыполненными. То есть, для каждого i, если $a_i > a_k$, a_i станет невыполненным, если оно было выполненным (если оно было невыполненным, ничего не изменится).
- У Окабэ плохая память, поэтому он может путешествовать во времени во время a_k **только сразу после того, как** попадет во время b_k и узнает, что он должен был выполнить k-е задание во время a_k. То есть, даже если Окабэ уже выполнял k-е задание ранее, он не вспомнит об этом до того, как опять наткнется на информацию об этом задании в момент времени b_k.

Пожалуйста, ознакомьтесь с примером путешествия во времени, который приведен в примечаниях.

Существует определенный набор s заданий, такой, что в первый момент, когда все задания из s будут одновременно выполнены (независимо от того, выполнены ли в данный момент какие-либо другие задания), произойдет забавная сцена. Окабэ нравится эта сцена, и он хочет знать, сколько раз Окабэ будет перемещаться во времени, прежде чем эта сцена произойдет. Найдите это число по модулю $10^9 + 7$. Можно доказать, что в конце концов все n заданий будут выполнены, поэтому ответ всегда существует.

Формат входных данных

Первая строка содержит целое число n ($1 \leqslant n \leqslant 2 \cdot 10^5$) — количество заданий, которые должен выполнить Окабэ.

Далее следует n строк. В k-й из этих строк содержатся два целых числа a_k и b_k ($1 \leqslant a_k < b_k \leqslant 2n$) — время, в которое Окабэ должен выполнить k-ю задачу и время, когда он узнает об этом соответственно. Все $2n$ этих времен попарно различны (поэтому каждое время от 1 до $2n$ включительно встречается во входных данных ровно один раз).

Следующая строка содержит единственное целое число t ($1 \leqslant t \leqslant n$) — размер набора s задач, которые приводят к смешной сцене.

Последняя строка содержит t целых чисел $s_1, s_2, \ldots, s_t$ — ($1 \leqslant s_k \leqslant n$, числа $s_1, s_2, \ldots, s_t$ различны) — множество s задач.

Формат выходных данных

Выведите одно целое число — количество раз, которое время Окабэ будет путешествовать во времени, пока все задачи в наборе s не будут одновременно завершены, по модулю $10^9 + 7$,

Примеры

стандартный ввод	стандартный вывод
2 1 4 2 3 2 1 2	3
2 1 4 2 3 1 1	2
1 1 2 1 1	1
6 10 12 3 7 4 6 2 9 5 8 1 11 3 2 4 6	17
16 31 32 3 26 17 19 4 24 1 28 15 21 12 16 18 29 20 23 7 8 11 14 9 22 6 30 5 10 25 27 2 13 6 3 8 2 5 12 11	138

Замечание

Для первого примера все задания должны быть выполнены, чтобы произошла забавная сценка.

Первоначально время равно 0. Ничего не происходит до момента времени 3, когда Окабэ понимает, что он должен был выполнить 2-ю задачу в момент времени 2. Затем он перемещается во времени в момент времени 2 и выполняет задание.

Поскольку задание уже выполнено, он не перемещается во времени снова, когда время снова

становится 3. Однако в момент времени 4 он перемещается в момент времени 1, чтобы выполнить 1-е задание.

Это отменяет 2-е задание. Это означает, что задание 2 на данный момент не выполнено, а значит, смешная сцена не произойдет в этот момент, несмотря на то что задание 1 на данный момент выполнено, и Окабэ ранее выполнял задачу 2.

Когда снова наступит время 3, он снова вернется во время 2 и снова выполнит 2-е задание.

Теперь все задания выполнены, а Окабэ успел переместиться во времени 3 раза.

Во втором примере Окабэ предстоит выполнить те же задания. Однако на этот раз для того, чтобы произошла забавная сцена, необходимо выполнить только первое задание. Прочитав приведенный выше пример, вы можете увидеть, что это произойдет, как только Окабэ переместится во времени в момент 2.

Задача H. Омкар и туры

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

Омкар устраивает туры по своей стране, Омкарландии! В Омкарландии n городов, и, что довольно любопытно, существует ровно $n-1$ двунаправленных дорог, соединяющих города друг с другом. Гарантируется, что вы можете добраться до любого города из любого другого города через сеть дорог.

Каждый город имеет значение удовольствия e. У каждой дороги есть пропускная способность c, обозначающее максимальное количество автомобилей, которое может на ней находиться, и соответствующая плата за проезд t. Однако система взимания платы в Омкарландии имеет интересную особенность: если автомобиль проезжает по нескольким дорогам за одну поездку, он платит только самую высокую плату за проезд по любой отдельной дороге, по которой он проехал. (Другими словами, он платит $\max t$ по всем дорогам, по которым они проехал). Если автомобиль не проезжает ни одной дороги, он платит 0 за проезд.

Омкар решил принять q туристических групп. Каждая туристическая группа состоит из v автомобилей, стартующих из города x. (Следует помнить, что туристическая группа с v автомобилями может передвигаться только по дорогам с пропускной способностью $\geqslant v$). Будучи организатором тура, Омкар хочет, чтобы его группы получили как можно больше удовольствия, но при этом он должен возместить своим группам расходы на проезд по дорогам, которые они должны заплатить. Таким образом, для каждой туристической группы Омкар хочет знать две вещи: во-первых, какое максимальное значение удовольствия имеет город y — город с максимальным значением удовольствия среди тех, до которых туристическая группа может добраться из своего начального города, и, во-вторых, сколько Омкар должен будет заплатить с каждого автомобиля, чтобы возместить расходы всей группы на поездку из x в y? (Эта поездка из x в y всегда будет проходить по кратчайшему пути из x в y).

В случае, если существует несколько городов с максимальным значением удовольствия, Омкар позволит своей туристической группе выбрать, в какой из них они хотят поехать. Поэтому, чтобы подготовиться ко всем возможным сценариям, он хочет знать сумму денег на автомобиль, которая ему необходима, чтобы гарантировать, что он сможет возместить расходы группы независимо от того, какой город они выберут.

Формат входных данных

Первая строка содержит два целых числа n и q ($2 \leqslant n \leqslant 2 \cdot 10^5$, $1 \leqslant q \leqslant 2 \cdot 10^5$), обозначающих количество городов и количество групп, соответственно.

Следующая строка содержит n целых чисел $e_1, e_2, \ldots, e_n$ ($1 \leqslant e_i \leqslant 10^9$), где e_i представляет собой значение удовольствия для города i.

Следующие $n-1$ строки содержат по четыре целых числа a, b, c и t ($1 \leqslant a, b \leqslant n$, $1 \leqslant c \leqslant 10^9$, $1 \leqslant t \leqslant 10^9$), обозначающих дорогу между городом a и городом b с пропускной способностью c и платой за проезд t.

Следующие q строк содержат по два целых числа v и x ($1 \leqslant v \leqslant 10^9$, $1 \leqslant x \leqslant n$), обозначающих количество автомобилей в туристической группе и начальном городе, соответственно.

Формат выходных данных

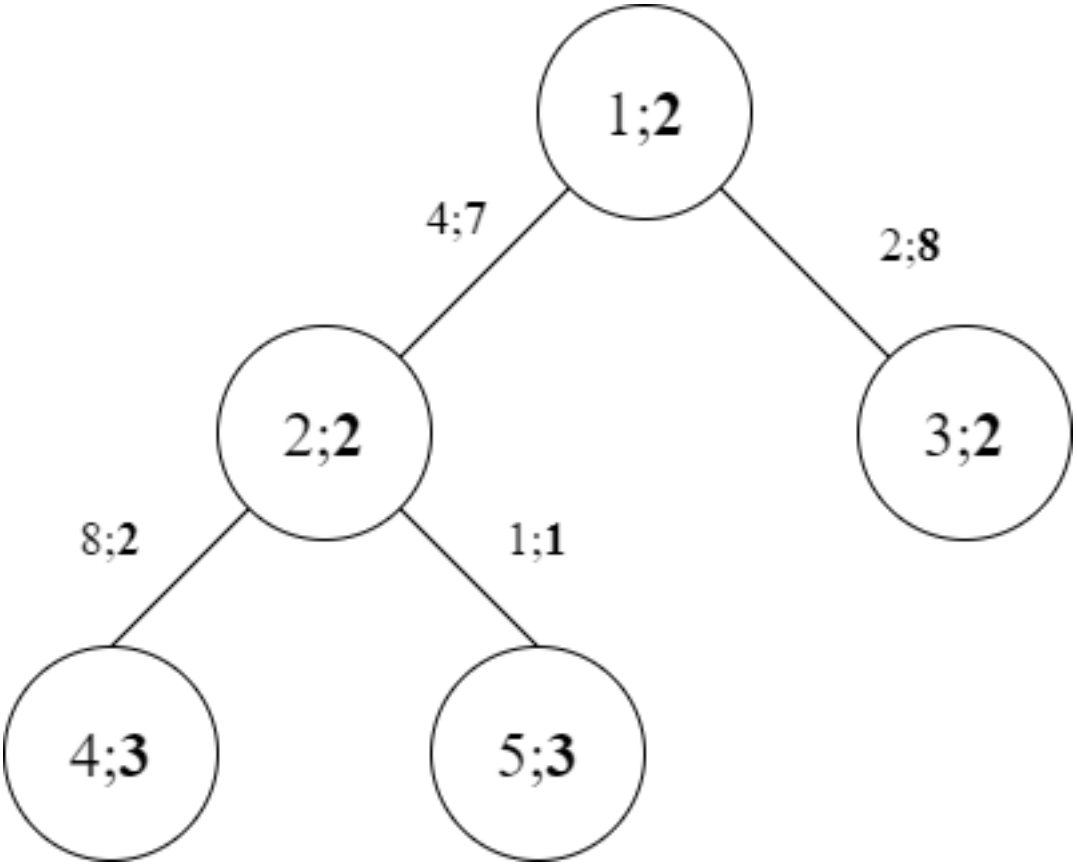
Выведите q строк. i-я строка должна содержать два целых числа: максимально возможное значение удовольствия города, до которого может добраться i-я туристическая группа, и количество денег на автомобиль, которое необходимо Омкару, чтобы гарантировать возмещение расходов i-й туристической группы.

Примеры

стандартный ввод	стандартный вывод
5 3 2 2 3 3 3 1 2 4 7 1 3 2 8 2 4 8 2 2 5 1 1 1 3 9 5 6 2	3 8 3 0 3 2
5 5 1 2 3 4 5 1 2 4 1 1 3 3 1 1 4 2 1 2 5 1 1 5 1 4 1 3 1 2 1 1 1	1 0 2 1 3 1 4 1 5 1
5 5 1 2 2 2 2 1 2 5 8 1 3 6 3 1 4 4 5 1 5 7 1 4 1 5 1 6 1 7 1 8 1	2 8 2 8 2 3 2 1 1 0

Замечание

Карта с первого примера показана ниже. Для вершин нежирные числа представляют индексы, а жирные — значения удовольствия. Для ребер нежирные цифры обозначают пропускную способность, а жирные — стоимость проезда.

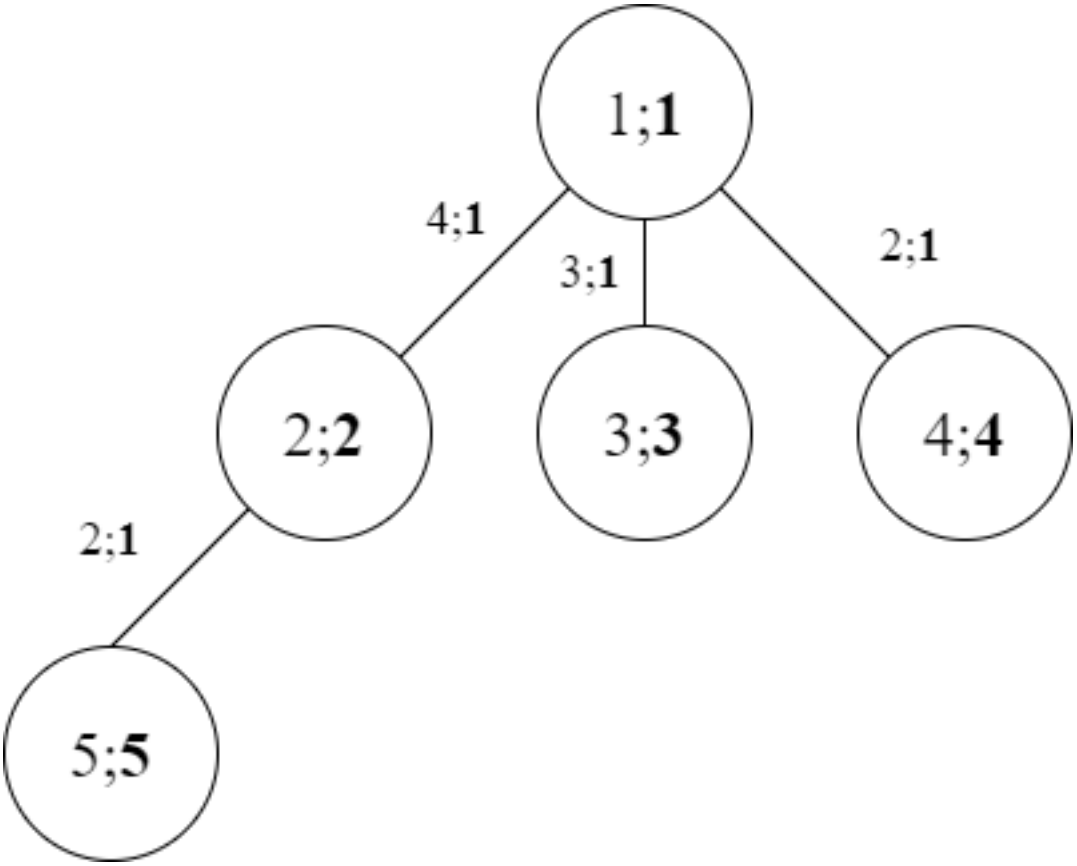


Для первого запроса туристическая группа размером 1, стартующая из города 3, может достичь городов 1, 2, 3, 4 и 5. Таким образом, наибольшее значение удовольствия, которого они могут достичь, равно 3. Если туристическая группа решит поехать в город 4, Омкар должен будет заплатить 8 за машину, что является максимумом.

Для второго запроса туристическая группа размером 9, стартующая из города 5, может достичь только города 5. Таким образом, наибольшее достижимое значение удовольствия по-прежнему составляет 3, и Омкар заплатит 0 за автомобиль.

Для третьего запроса туристическая группа размером 6, стартующая из города 2, может достичь городов 2 и 4. Наибольшее достижимое значение удовольствия снова равно 3. Если туристическая группа решит поехать в город 4, Омкар должен будет заплатить 2 за машину, что является максимумом.

Карта второй выборки показана ниже:



Для первого запроса, туристическая группа размером 5, стартующая из города 1, может достичь только города 1. Таким образом, их максимальная ценность удовольствия составляет 1, а стоимость, которую придется заплатить Омкару, равна 0 за автомобиль.

Для второго запроса туристическая группа размером 4, стартующая из города 1, может добраться до городов 1 и 2. Таким образом, их максимальная ценность удовольствия составляет 2, а Омкар заплатит 1 за автомобиль.

Для третьего запроса, туристическая группа размером 3, стартующая из города 1, может добраться до городов 1, 2 и 3. Таким образом, их максимальная ценность удовольствия составляет 3, и Омкар заплатит 1 за автомобиль.

Для четвертого запроса туристическая группа размером 2, стартующая из города 1, может добраться до городов 1, 2, 3 и 4. Таким образом, их максимальная ценность удовольствия составляет 4, и Омкар заплатит 1 за автомобиль.

Для пятого запроса туристическая группа размером 1, стартующая из города 1, может добраться до городов 1, 2, 3, 4 и 5. Таким образом, их максимальное удовольствие составляет 5, и Омкар заплатит 1 за машину.