Станции (stations)

Магистральная сеть Сингапура (SIB) состоит из n станций, каждой из которых присвоен **индекс** от 0 до $n-1$. В сети также присутствуют $n-1$ двусторонних соединений, пронумерованных от 0 до $n-2$, каждое из которых соединяет две различных станции. Две станции, соединенные напрямую, называются соседями.

Путем от станции x до станции y называется последовательность попарно различных станций $a_0, a_1, \cdots, a_p$ такая, что $a_0 = x$, $a_p = y$, а любые две последовательные станции являются соседями. Гарантируется, что существует **ровно один** путь от любой станции x до любой другой станции y.

Любая из станций x может создать пакет (фрагмент данных) и отправить его любой станции y, которая называется **пунктом назначения** пакета. Этот пакет должен быть передан от станции x до станции y следующим образом. Пусть пакет в текущий момент находится на станции z, а пунктом назначения этого пакета является станция y ($z \neq y$). В этом случае станция

1. выполняет **процедуру маршрутизации**, которая определяет соседа z, который лежит на единственном пути от z до y, и
2. передает пакет этому соседу.

Однако, станции имеют ограниченный размер памяти и не содержат весь список соединений в SIB для использования в процедуре маршрутизации.

Ваша задача состоит в том, чтобы составить схему маршрутизации SIB, которая состоит из двух функций.

- Первая функция получает на вход число n, список соединений SIB и число $k \geq n-1$. Она присваивает каждой из станций уникальный **идентификатор** в диапазоне от 0 до k, включительно.

- Вторая функция представляет собой процедуру маршрутизации, которая будет передана на каждую из станций после присваивания идентификаторов. Эта функций получает **только** следующие входные данные:

  - s, **идентификатор** станции, на которой пакет находится в текущий момент,
  - t, **идентификатор** станции, которая является пунктом назначения пакета ($t \neq s$),
  - c, список **идентификаторов** всех соседей станции s.

  Функция должна вернуть **идентификатор** соседа s, куда пакет должен быть передан дальше.

В одной из подзадач финальная оценка решения зависит от того, какой максимальный идентификатор будет присвоен какой-либо станции (чем меньше, тем лучше).

Детали реализации

Вы должны реализовать две функции:

```
int[] label(int n, int k, int[] u, int[] v)
```

- n: число станций в SIB.
- k: максимальный идентификатор, который может быть использован.
- u и v: два массива размера $n-1$, описывающие соединения. Для каждого i ($0 \leq i \leq n-2$), соединение i находится между станциями $u[i]$ и $v[i]$.
- Функция должна вернуть массив L длины n. Для каждого i ($0 \leq i \leq n-1$) значение $L[i]$ должно содержать идентификатор, присвоенный станции i. Все элементы массива L должны быть попарно различны и должны находиться в диапазоне от 0 до k, включительно.

```
int find_next_station(int s, int t, int[] c)
```

- s: идентификатор станции, на которой сейчас находится пакет.
- t: идентификатор станции, которая является пунктом назначения пакета.
- c: массив, содержащий список идентификаторов соседей s. Массив c отсортирован по возрастанию.
- Функция должна вернуть идентификатор соседа s, куда пакет должен быть передан дальше.

Каждый тест содержит один или несколько сценариев (описаний различных вариантов сетей SIB). Для теста, состоящего из r сценариев, **программа**, вызывающая описанные выше функции, будет запущена в точности два раза следующим образом.

Во время первого запуска программы:

- функция `label` будет вызвана r раз,
- результаты вызовов будут сохранены тестирующей системой, и
- функция `find_next_station` не будет вызвана ни разу.

Во время второго запуска программы:

- функция `find_next_station` может быть вызвана несколько раз,
- идентификаторы, используемые в функции `find_next_station`, соответствуют идентификаторам, полученным функцией `label` для одного из **выбранных произвольным образом** сценариев из первого запуска, и
- функция `label` не будет вызвана ни разу.

В частности, никакая информаций, сохраненная в статические или глобальные переменные во время первого запуска, не будет доступна внутри функции `find_next_station`.

Примеры

Рассмотрим следующий вызов функции:

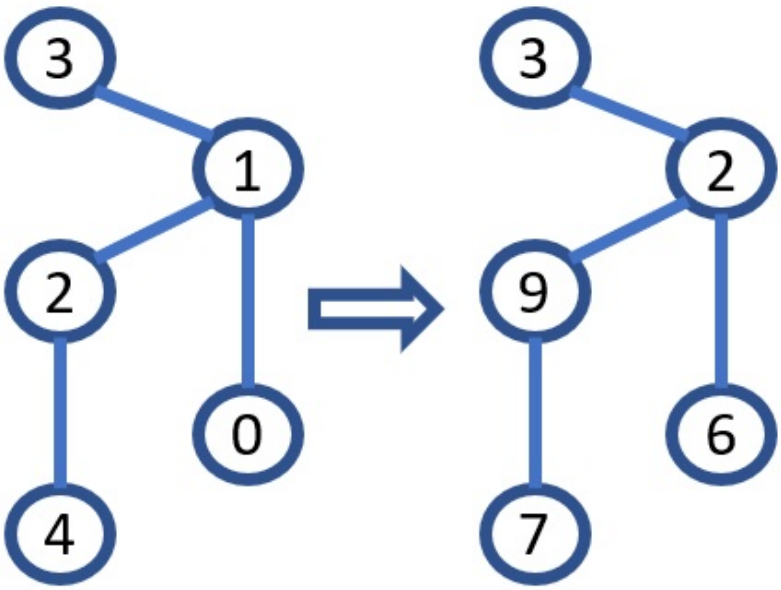
```
label(5, 10, [0, 1, 1, 2], [1, 2, 3, 4])
```

В сети находятся 5 станций, 4 соединения, соответствующие парам $(0, 1)$, $(1, 2)$, $(1, 3)$ и $(2, 4)$. Каждый из идентификаторов может быть равен числу от 0 до $k = 10$.

Для того, чтобы назначать идентификаторы следующим образом

Index	Label
0	6
1	2
2	9
3	3
4	7

функция `label` должна вернуть массив $[6, 2, 9, 3, 7]$. На рисунке ниже картинка слева демонстрирует индексы вершин, а картинка справа присвоенные идентификаторы.



Пусть идентификаторы были присвоены способом, описанным выше. Рассмотрим следующий вызов функции:

```
find_next_station(9, 6, [2, 7])
```

Пакет находится на станции с идентификатором 9, а пункт назначения пакета имеет идентификатор 6. Идентификаторы станций, лежащих на пути от текущей станции к пункту назначения, равны $[9, 2, 6]$. Таким образом, функция должна вернуть значение 2, которое соответствует идентификатору станции, куда пакет должен быть передан дальше (эта станция имеет индекс 1).

Рассмотрим другой возможный вызов функции:

```
find_next_station(2, 3, [3, 6, 9])
```

Функция должна вернуть значение 3, так как пункт назначения, имеющий идентификатор 3, является соседом станции с идентификатором 2, а значит, может получить пакет напрямую.

Ограничения

- $1 \leq r \leq 10$

Для каждого вызова функции `label`:

- $2 \leq n \leq 1000$
- $k \geq n - 1$
- $0 \leq u[i], v[i] \leq n - 1$ (для всех $0 \leq i \leq n - 2$)

Для каждого вызова функции `find_next_station`, входные данные соответствуют одному произвольно выбранному вызову функции `label`. Рассмотрим идентификаторы, которые получились в результате этого вызова. Тогда:

- s и t обозначают идентификаторы двух различных станцией.
- c содержит идентификаторы всех соседей станции с идентификатором s, отсортированные по возрастания.

Для каждого из тестовых примеров общая длина всех массивов c, переданных функции `find_next_station`, не превышает $100\ 000$, суммарно для всех сценариев.

Подзадачи

1. (5 баллов) $k = 1000$, ни одна из станции не имеет более 2 соседей.
2. (8 баллов) $k = 1000$, соединение i соответствует станциям $i + 1$ и $\left\lfloor \frac{i}{2} \right\rfloor$.
3. (16 баллов) $k = 1\ 000\ 000$, не более одной станции имеет более 2 соседей.
4. (10 баллов) $n \leq 8$, $k = 10^9$
5. (61 балл) $k = 10^9$

В подзадаче 5 вы можете получить частичный балл. Пусть m обозначает максимальное

значение элементов `label` среди всех сценариев. Ваш балл за эту подзадач вычисляется следующим образом:

Максимальный идентификатор	Балл
$m \geq 10^9$	0
$2000 \leq m < 10^9$	$50 \cdot \log_{5 \cdot 10^5}(\frac{10^9}{m})$
$1000 < m < 2000$	50
$m \leq 1000$	61

Пример проверяющего модуля

Проверяющий модуль считывает данные в следующем формате:

- строка 1: r

Далее следуют r блоков данных, каждый описывает один из сценариев. Каждый из блоков имеет следующий формат:

- строка 1: n k
- строка $2 + i$ ($0 \leq i \leq n - 2$): $u[i]$ $v[i]$
- строка $1 + n$: q: количество вызовов функции `find_next_station`.
- строка $2 + n + j$ ($0 \leq j \leq q - 1$): $z[j]$ $y[j]$ $w[j]$: **индексы** станций, используемых в j-м вызове функции `find_next_station`. Станция $z[j]$ содержит пакет в текущий момент, станция $y[j]$ является пунктом назначения, а станция $w[j]$ обозначает место, куда пакет должен быть передан дальше.

Проверяющий модуль выводит данные в следующем формате:

- строка 1: m

Далее следуют r блоков данных, соответствующих сценариям из входных данных. Каждый из блоков имеет следующий формат:

- строка $1 + j$ ($0 \leq j \leq q - 1$): **индекс** станции, чей **идентификатор** был получен в результате j-го вызова функции `find_next_station` в данном сценарии.

Обратите внимание, что один запуск проверяющего модуля вызывает обе функции `label` и `find_next_station`.