Станциялар (stations)

Сингапурдун Интернет тору (СИТ) 0дөн $(n-1)$ге чейинки **индекси** дайындалган n станциядан турат. 0 дөн $(n-2)$ге чейинки номурланган $(n-1)$ эки багыттуу жолчо бар. Ар бир жолчо эки ар түрдүү станцияны бириктирет. Бир жолчо менен байланышкан эки станция кошуналар деп аталат.

x-станциядан y-станцияга чейинки жол: $a_0-, a_1-, \ldots, a_p-$ ар түрдуу станциялардын удаалаштыгы, $a_0 = x$, $a_p = y$, мында жолдогу ар эки удаа станция кошуналар. Каалаган x-станциясынан башка каалаган y-станциясына **бир гана** жол бар.

Каалаган x-станция пакетти (маалыматтын топтому) түзүп, башка y-станцияга жөнөтө алат, ал y-станция пакеттин **максаты** деп аталат. Бул пакет x-станциядан y-станцияга чейинки бир гана жол боюнча төмөнкүдөйчө жиберилиши керек. Учурда y-станцияга багытталган пакетти азыр камтыган z-станцияны ($z \neq y$) карап көрөлү. Бул кырдаалда z-станция:

1. z-станциядан y'-станцияга чейинки бир гана жолдо турган z-станциянын кошунасын аныктаган **маршруттоо процедурасын** ишке ашырат жана
2. пакетти ушул кошунага жөнөтөт.

Бирок, станциялардын эс тутуму чектелген жана аны жолчолордун тизмесин СИТте маршруттоо процедурасында колдонуу үчүн сактабайт.

Сиздин милдетиңиз эки процедурадан турган СИТ үчүн маршруттоо схемасын ишке ашыруу.

- Биринчи процедурага n берилген, СИТтеги жолчолордун тизмеси жана $k >= n-1$ бүтүн сандары. Ал ар бир станцияга **бир гана** бүтүн **лейбл** ($0 \ldots k$ дан), кошо алат.

- Экинчи процедура - маршруттоо процедурасы, ал бардык станцияларга лейблди берилгенден кийин жайылтылат. Ага төмөнкүдөй маалыматтарды **гана** берилет:

  - s, учурда пакетти кармаган станциянын **лейбли** ,
  - t, пакеттин максаттуу станциясынын **лейбли** ($t \neq s$),
  - c, s-станциянын бардык коңшуларынын **лейбллеринин** тизмеси.

  Ал пакет s-станциянын жөнөтүлө турган кошунасынын **лейблин** кайтарып бериши керек.

Бир тапшырма бөлүгүндө, сиздин чыгарылышыңыздын упайы каалаган станцияга берилген максималдуу лейблинин маанисине жараша болот (жалпысынан, кичинерээк жакшыраак болот).

Implementation details

You should implement the following procedures:

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

- n: number of stations in the SIB.
- k: maximum label that can be used.
- u and v: arrays of size $n-1$ describing the links. For each i ($0 \le i \le n-2$), link i connects stations with indices $u[i]$ and $v[i]$.
- This procedure should return a single array L of size n. For each i ($0 \le i \le n-1$) $L[i]$ is the label assigned to station with index i. All elements of array L must be unique and between 0 and k, inclusive.

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

- s: label of the station holding a packet.
- t: label of the packet's target station.
- c: an array giving the list of the labels of all neighbours of s. The array c is sorted in ascending order.
- This procedure should return the label of a neighbour of s that the packet should be forwarded to.

Each test case involves one or more independent scenarios (i.e., different SIB descriptions). For a test case involving r scenarios, a **program** that calls the above procedures is run exactly two times, as follows.

During the first run of the program:

- `label` procedure is called r times,
- the returned labels are stored by the grading system, and
- `find_next_station` is not called.

During the second run of the program:

- `find_next_station` may be called multiple times. In each call, an **arbitrary** scenario is chosen, and the labels returned by the call to `label` procedure in that scenario are used as the inputs to `find_next_station`.
- `label` is not called.

In particular, any information saved to static or global variables in the first run of the program is not available within `find_next_station` procedure.

Example

Consider the following call:

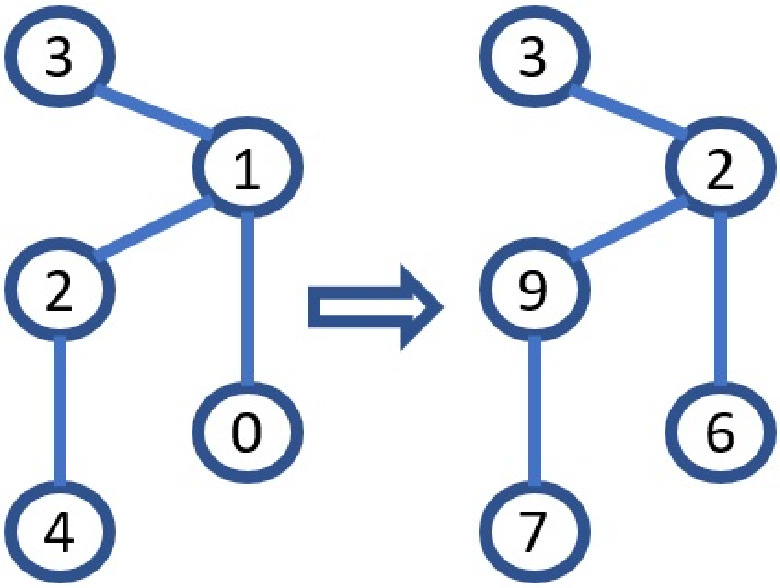
```
label(5, 10, [0, 1, 1, 2], [1, 2, 3, 4])
```

There are a total of 5 stations, and 4 links connecting pairs of stations with indices $(0, 1)$, $(1, 2)$, $(1, 3)$ and $(2, 4)$. Each label can be an integer from 0 to $k = 10$.

In order to report the following labelling:

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

the `label` procedure should return [6, 2, 9, 3, 7]. The numbers in the following figure show the indices (left panel) and assigned labels (right panel).



Assume the labels have been assigned as described above and consider the following call:

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

This means that the station holding the packet has label 9, and the target station has label 6. The labels of stations on the path to the target station are $[9, 2, 6]$. Hence, the call should return 2, which is the label of the station that the packet should be forwarded to (which has index 1).

Consider another possible call:

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

The procedure should return 3, since the target station with label 3 is a neighbour of the station with label 2, and hence should receive the packet directly.

Constraints

- $1 \le r \le 10$

For each call to `label`:

- $2 \le n \le 1000$
- $k \ge n - 1$
- $0 \le u[i], v[i] \le n - 1$ (for all $0 \le i \le n - 2$)

For each call to `find_next_station`, the input comes from an arbitrarily chosen previous call to `label`. Consider the labels it produced. Then:

- s and t are labels of two different stations.
- c is the sequence of all labels of neighbours of the station with label s, in ascending order.

For each test case, the total length of all arrays c passed to the procedure `find_next_station` does not exceed $100\,000$ for all scenarios combined.

Subtasks

1. (5 points) $k = 1000$, no station has more than 2 neighbours.
2. (8 points) $k = 1000$, link i connects stations $i + 1$ and $\lfloor \frac{i}{2} \rfloor$.
3. (16 points) $k = 1\,000\,000$, at most one station has more than 2 neighbours.
4. (10 points) $n \le 8$, $k = 10^9$
5. (61 points) $k = 10^9$

In subtask 5 you can obtain a partial score. Let m be the maximum label value returned by `label` across all scenarios. Your score for this subtask is calculated according to the following table:

Maximum label	Score
$m \ge 10^9$	0
$2000 \le m < 10^9$	$50 \cdot \log_{5 \cdot 10^5}(\frac{10^9}{m})$
$1000 < m < 2000$	50
$m \le 1000$	61

Sample grader

The sample grader reads the input in the following format:

- line 1: r

r blocks follow, each describing a single scenario. The format of each block is as follows:

- line 1: $n \; k$
- line $2 + i$ ($0 \le i \le n - 2$): $u[i] \; v[i]$
- line $1 + n$: q: the number of calls to `find_next_station`.
- line $2 + n + j$ ($0 \le j \le q - 1$): $z[j] \; y[j] \; w[j]$: **indices** of stations involved in the j-th call to `find_next_station`. The station $z[j]$ holds the packet, the station $y[j]$ is the packet's target, and the station $w[j]$ is the station that the packet should be forwarded to.

The sample grader prints the result in the following format:

- line 1: m

r blocks corresponding to the consecutive scenarios in the input follow. The format of each block is as follows:

- line $1 + j$ ($0 \le j \le q - 1$): **index** of the station, whose **label** was returned by the j-th call to `find_next_station` in this scenario.

Note that each run of the sample grader calls both `label` and `find_next_station`.