Navigation

Taiwan has a high density of high mountains, featuring over 250 peaks above 3,000 meters. A-Ming plans to construct robotic zipline delivery networks in the Central Mountain Range of Taiwan, to transport packages between the remote villages.

In A-Ming's plan, each network consists of N nodes, numbered from 0 to $N-1$. Among these nodes, **exactly** $K = 6$ of them are power stations and other $N-K$ nodes are all village nodes. The nodes are connected by $N-1$ **bidirectional** cables, numbered from 0 to $N-2$. For each i ($0 \leq i \leq N-2$), cable i connects nodes $U[i]$ and $V[i]$. Each cable connects two different nodes, and each pair of nodes is connected by at most one cable. It is guaranteed that it is possible to move between any pair of nodes using the cables.

The **distance** of nodes a and b is denoted by $d(a, b)$ and defined as follows.

- If $a = b$ then $d(a, b) = 0$.
- Otherwise, $d(a, b)$ is the minimum number of cables needed to move from a to b.

We say that the **branching factor** of the network is at least δ if each node in the network is either connected to exactly 1 or at least δ cables. A-Ming knows a positive integer B such that the branching factor of the network is at least B.

A-Ming prepared 100 different types of cables, numbered $0, 1, \ldots, 99$. Each cable in the network will be built with one of those types.

The robots will ride along the zipline cables to carry out delivery tasks. A-Ming would like to install a navigation program that helps the robots to return to the power stations and charge themselves. However, the robots have extremely low memory. Hence, in designing the navigation program, the robots navigate only based on the number of power stations $K = 6$, the guaranteed branching factor B, and the types of cable attached to the robot's current location.

When a robot wants to navigate at a village node u connected to two or more cables, the robot first scans the cables connected to u in an arbitrary order. The program is then provided with a list containing the types of the cables in the order. For each of the cables, the program needs to count the number of power stations such that following the cable decreases the distance between the robot and the station.

Specifically, let $v_1, \ldots, v_k$ ($k \geq 2$) be the nodes directly connected to u by a cable and c_i ($1 \leq i \leq k$) be the type of the cable connecting node u and v_i. The program is then provided with K, B, and the list $c_1, c_2, \ldots, c_k$. For each i ($1 \leq i \leq k$), the program has to determine the number of power stations p such that $d(v_i, p) < d(u, p)$.

Your task is to devise a strategy to assign the cable types and design the navigation program. The score of your solution depends on the number of different type of cables used (see Subtasks and Scoring section for details).

Implementation details

You need to implement two procedures, one for assigning the cable types and another for the robots' program.

The procedure you should implement to **assign the cable types** is:

```
std::vector<int> construct_network(int N, int K, int B,
                                   std::vector<int> U, std::vector<int> V,
                                   std::vector<int> P)
```

- N: the number of nodes.
- K: the number of power stations.
- B: the guaranteed minimum branching factor of the network.
- U, V: arrays of lengths $N - 1$ describing cable connections.
- P: an array of length $K = 6$ describing the power station nodes.
- This procedure is called **at most** $10\,000$ **times** for each test case.

This procedure should return an array T of length $N - 1$:

- A cable with type $T[i]$ is used to connect nodes $U[i]$ and $V[i]$.
- Each element $T[i]$ must satisfy $0 \leq T[i] < 100$.

The procedure you should implement for the **robots' program** is:

```
std::vector<int> navigate(int K, int B, std::vector<int> C)
```

- K: the number of power stations.
- B: the guaranteed minimum branching factor of the network.
- C: the list of the types of all cables connected to some **village node connected to at least** 2 **cables**, in an arbitrary order.
- It is guaranteed that the length of C is at least B.
- This procedure is called at most $100\,000$ times for each test case.

- The procedure navigate may not rely on the original network structure; in particular, the grading system may reorder all calls to navigate across the different constructed networks within the same test case.

This procedure should return an array D:

- The length of array D must be the same as array C.
- $D[i]$ must be the number of power stations where following the cable corresponding to $C[i]$ decreases the distance between the robot and the station.

Note that in the grading system, the `construct_network` and `navigate` procedures are called **in two separate processes**.

Constraints

- $K = 6$.
- $K < N \leq 100\,000$.
- The sum of N over all calls to `construct_network` does not exceed $100\,000$ in each test case.
- $0 \leq U[i] < V[i] < N$ for each $0 \leq i \leq N - 2$.
- It is possible to move from any node to any other node using the cables.
- $0 \leq P[0] < P[1] < \cdots < P[K-1] < N$.
- Each array C is a permutation of the list of the types of all cables connected to some village node that is connected to at least 2 cables.
- The sum of the lengths of array C over all calls to `navigate` does not exceed $200\,000$ in each test case.

Subtasks and Scoring

Subtask	Score	Additional Constraints
1	6	$B = 2, N = 7$.
2	14	$B = 2, U[i] = i, V[i] = i + 1$ for each i such that $0 \leq i < N - 1$.
3	20	$B = 5$.
4	20	$B = 4$.
5	40	$B = 2$.

In any of the test cases, if at least one of the following conditions holds, then the score of your solution for that test case will be 0 (reported as `Output isn't correct` in CMS):

- The return value of any call to `construct_network` is invalid.

- The return value of any call to `navigate` is incorrect.

Otherwise, let S be the smallest integer greater than all numbers in every array T returned from each `construct_network` call. In other words, S is the smallest integer such that all constructed networks only use cables of type $0, 1, \ldots, S-1$.

Your score for each subtask depends on S, as follows:

Condition	Subtask 1	Subtask 2	Subtask 3 and 4	Subtask 5
$100 < S$	0	0	0	0
$16 \leq S \leq 100$	2	2	$12 - \log_2 S$	$24 - 2\log_2 S$
$10 \leq S \leq 15$	2	4	$16 - 0.5S$	$32 - S$
$7 \leq S \leq 9$	2	6	$21 - S$	$42 - 2S$
$S = 6$	2	10	15	30
$S = 5$	**6**	**14**	17.5	**40**
$S \leq 4$	**6**	**14**	**20**	**40**

In particular, in Subtask 1, 2 and 5, you will receive full point if $S \leq 5$, and in Subtask 3 and 4, you will receive full point if $S \leq 4$.

Example

Consider the scenario in which $N = 9$, $K = 6$, and $B = 2$.

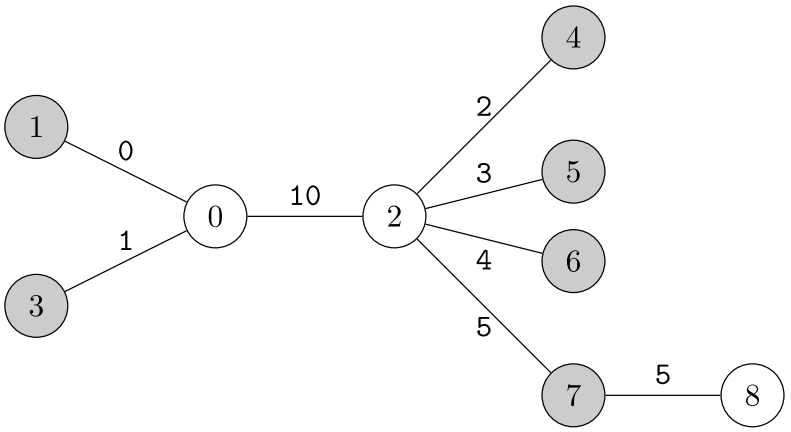
The network plan is specified by $U = [0, 0, 0, 2, 2, 2, 2, 7], V = [1, 2, 3, 4, 5, 6, 7, 8]$. The power stations are $P = [1, 3, 4, 5, 6, 7]$. Consider the following call to the first process:

```
construct_network(9, 6, 2, [0, 0, 0, 2, 2, 2, 2, 7],
                           [1, 2, 3, 4, 5, 6, 7, 8],
                           [1, 3, 4, 5, 6, 7])
```

Suppose A-Ming builds the network by returning:

```
[0, 10, 1, 2, 3, 4, 5, 5]
```

The constructed network is illustrated as follows.



Then, in the second process, suppose the robot needs to navigate at node 0. The node 0 is connected to node $1, 2$ and 3. If the robot reads the cable type in this order, the following call will be made:

```
navigate(6, 2, [0, 10, 1])
```

Based on the network structure:

- The first power station, node 1, has the shortest distance to itself.
  In particular, $0 = d(1,1) < 1 = d(0,1) < 2 = d(2,1) = d(3,1)$.
- The second power station, node 3, has the shortest distance to itself.
  In particular, $0 = d(3,3) < 1 = d(0,3) < 2 = d(1,3) = d(2,3)$.
- Other power stations, node 4, 5, 6, and 7, have shorter distances to node 2.
  In particular, $1 = d(2,u) < 2 = d(0,u) < 3 = d(1,u) = d(3,u)$ when $u = 4, 5, 6,$ or 7.

The number of power stations with distance decreased after following cables $(0,1)$, $(0,2)$, and $(0,3)$ are 1, 4 and 1, respectively. Hence, the procedure should return:

```
[1, 4, 1]
```

Note that `navigate` can also be called for different permutations of C. For example, `navigate(6, 2, [1, 0, 10])` is also a possible call for node 0. The procedure should return `[1, 1, 4]` accordingly.

Suppose the robot needs to navigate at node 2. The following call is made:

```
navigate(6, 2, [10, 2, 3, 4, 5])
```

The procedure should return:

```
[2, 1, 1, 1, 1]
```

In this network, the procedure `navigate` would **never** be called for other nodes, since:

- node $1, 3, 4, 5, 6$ and 7 are all power stations, and
- node 8 is connected to only one cable.

In this test case, the used cable types are $0, 1, 2, 3, 4, 5$, and 10. Thus, $S = 11$ will be used to determine the score.

The attachment package for this task contains two other sample inputs and outputs for the sample grader in addition to this example.

Sample Grader

The sample grader calls both `construct_network` and `navigate` in the same process. In addition, when the sample grader calls `navigate`, the cable types are listed in the order in which their corresponding cables appear in the input. Both behaviors are different from the actual grader.

Input format:

```
T
(scenario 0)
(scenario 1)
...
(scenario T-1)
```

Each scenario is in the following format:

```
N
B
U[0] V[0]
U[1] V[1]
...
U[N-2] V[N-2]
K
P[0] P[1] ... P[K-1]
Q
X[0] X[1] ... X[Q-1]
```

In each scenario, the sample grader will call `navigate` Q times, the i-th call with the information of node $X[i-1]$. Note that the sample grader supports setting K to integers other than 6.

Output format:

If any of the returned arrays of `construct_network` or `navigate` is invalid, the sample grader aborts and prints the reason. Otherwise, the sample grader outputs each array D returned from the `navigate` call in a line.

After all scenarios have been processed, the sample grader prints a line to the standard error:

```
S = S'
```

where S' is the smallest integer greater than all numbers in every array T returned from each `construct_network` call in the test case.