Codeforces Round #805 (Div. 3)

A. Round Down the Price

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

At the store, the salespeople want to make all prices *round*.

In this problem, a number that is a power of 10 is called a *round* number. For example, the numbers $10^0 = 1$, $10^1 = 10$, $10^2 = 100$ are *round* numbers, but 20, 110 and 256 are not *round* numbers.

So, if an item is worth m bourles (the value of the item is not greater than 10^9), the sellers want to change its value to the nearest *round* number that is not greater than m. They ask you: by how many bourles should you **decrease** the value of the item to make it worth exactly 10^k bourles, where the value of k — is the maximum possible (k — any non-negative integer).

For example, let the item have a value of 178-bourles. Then the new price of the item will be 100, and the answer will be $178 - 100 = 78$.

Input

The first line of input data contains a single integer t ($1 \le t \le 10^4$) — the number of test cases .

Each test case is a string containing a single integer m ($1 \le m \le 10^9$) — the price of the item.

Output

For each test case, output on a separate line a single integer d ($0 \le d < m$) such that if you reduce the cost of the item by d bourles, the cost of the item will be the maximal possible *round* number. More formally: $m - d = 10^k$, where k — the maximum possible non-negative integer.

Example

input
7 1 2 178 20 999999999 9000 987654321
output
0 1 78 10 899999999 8000 887654321

Note

In the example:

- $1 - 0 = 10^0$,
- $2 - 1 = 10^0$,
- $178 - 78 = 10^2$,
- $20 - 10 = 10^1$,
- $999999999 - 899999999 = 10^8$,
- $9000 - 8000 = 10^3$,
- $987654321 - 887654321 = 10^8$.

Note that in each test case, we get the maximum possible *round* number.

B. Polycarp Writes a String from Memory

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp has a poor memory. Each day he can remember no more than 3 of different letters.

Polycarp wants to write a non-empty string of s consisting of lowercase Latin letters, taking **minimum** number of days. In how many days will he be able to do it?

Polycarp initially has an empty string and can only add characters to the end of that string.

For example, if Polycarp wants to write the string `lollipops`, he will do it in 2 days:

- on the first day Polycarp will memorize the letters `l`, `o`, `i` and write `lolli`;
- On the second day Polycarp will remember the letters `p`, `o`, `s`, add `pops` to the resulting line and get the line `lollipops`.

If Polycarp wants to write the string `stringology`, he will do it in 4 days:

- in the first day will be written part `str`;
- on day two will be written part `ing`;
- on the third day, part of `olog` will be written;
- on the fourth day, part of `y` will be written.

For a given string s, print the minimum number of days it will take Polycarp to write it.

Input
The first line of input data contains a single integer t ($1 \le t \le 10^4$) — the number of test cases.

Each test case consists of a non-empty string s consisting of lowercase Latin letters (the length of the string s does not exceed $2 \cdot 10^5$) — the string Polycarp wants to construct.

It is guaranteed that the sum of string lengths s over all test cases does not exceed $2 \cdot 10^5$.

Output
For each test case, print a single number — **minimum** number of days it will take Polycarp to write the string s from memory.

Example

input
6 lollipops stringology abracadabra codeforces test f
output
2 4 3 4 1 1

C. Train and Queries

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Along the railroad there are stations indexed from 1 to 10^9. An express train always travels along a route consisting of n stations with indices $u_1, u_2, \ldots, u_n$, where ($1 \le u_i \le 10^9$). The train travels along the route from left to right. It starts at station u_1, then stops at station u_2, then at u_3, and so on. Station u_n — the terminus.

It is possible that the train will visit the same station more than once. That is, there may be duplicates among the values $u_1, u_2, \ldots, u_n$.

You are given k queries, each containing two different integers a_j and b_j ($1 \le a_j, b_j \le 10^9$). For each query, determine whether it is possible to travel by train from the station with index a_j to the station with index b_j.

For example, let the train route consist of 6 of stations with indices $[3, 7, 1, 5, 1, 4]$ and give 3 of the following queries:

- $a_1 = 3, b_1 = 5$
  It is possible to travel from station 3 to station 5 by taking a section of the route consisting of stations $[3, 7, 1, 5]$. Answer: YES.
- $a_2 = 1, b_2 = 7$
  You cannot travel from station 1 to station 7 because the train cannot travel in the opposite direction. Answer: NO.
- $a_3 = 3, b_3 = 10$
  It is not possible to travel from station 3 to station 10 because station 10 is not part of the train's route. Answer: NO.

Input

The first line of the input contains an integer t ($1 \le t \le 10^4$) —the number of test cases in the test.

The descriptions of the test cases follow.

The first line of each test case is empty.

The second line of each test case contains two integers: n and k ($1 \le n \le 2 \cdot 10^5, 1 \le k \le 2 \cdot 10^5$) —the number of stations the train route consists of and the number of queries.

The third line of each test case contains exactly n integers $u_1, u_2, \ldots, u_n$ ($1 \le u_i \le 10^9$). The values $u_1, u_2, \ldots, u_n$ are not necessarily different.

The following k lines contain two different integers a_j and b_j ($1 \le a_j, b_j \le 10^9$) describing the query with index j.

It is guaranteed that the sum of n values over all test cases in the test does not exceed $2 \cdot 10^5$. Similarly, it is guaranteed that the sum of k values over all test cases in the test also does not exceed $2 \cdot 10^5$

Output

For each test case, output on a separate line:

- YES, if you can travel by train from the station with index a_j to the station with index b_j
- NO otherwise.

You can output YES and NO in any case (for example, strings yEs, yes, Yes and YES will be recognized as a positive response).

Example

input
3 6 3 3 7 1 5 1 4 3 5 1 7 3 10 3 3 1 2 1 2 1 1 2 4 5 7 5 2 1 1 1 2 4 4 1 3 1 4 2 1 4 1 1 2
output
YES NO NO YES YES NO NO YES YES NO YES

Note

The first test case is explained in the problem statement.

D. Not a Cheap String

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Let s be a string of lowercase Latin letters. Its price is the sum of the indices of letters (an integer between 1 and 26) that are included in it. For example, the price of the string abca is $1 + 2 + 3 + 1 = 7$.

The string w and the integer p are given. Remove the minimal number of letters from w so that its price becomes less than or equal to p and print the resulting string. Note that the resulting string may be empty. You can delete arbitrary letters, they do not have to go in a row. If the price of a given string w is less than or equal to p, then nothing needs to be deleted and w must be output.

Note that when you delete a letter from w, the order of the remaining letters is preserved. For example, if you delete the letter e from the string test, you get tst.

Input
The first line of input contains an integer t ($1 \le t \le 10^4$) — the number of test cases in the test. The following are descriptions of t test cases.

Each case consists of two lines.

The first of them is the string w, it is non-empty and consists of lowercase Latin letters. Its length does not exceed $2 \cdot 10^5$.

The second line contains an integer p ($1 \le p \le 5\,200\,000$).

It is guaranteed that the sum of string lengths w over all test cases does not exceed $2 \cdot 10^5$.

Output
Output exactly t rows, the i-th of them should contain the answer to the i-th set of input data. Print the longest string that is obtained from w by deleting letters such that its price is less or equal to p. If there are several answers, then output any of them.

Note that the empty string — is one of the possible answers. In this case, just output an empty string.

Example

input
5 abca 2 abca 6 codeforces 1 codeforces 10 codeforces 100
output
aa abc cdc codeforces

E. Split Into Two Sets

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp was recently given a set of n (number n — even) dominoes. Each domino contains two integers from 1 to n.

Can he divide all the dominoes into two sets so that all the numbers on the dominoes of each set are different? Each domino must go into exactly one of the two sets.

For example, if he has 4 dominoes: $\{1, 4\}$, $\{1, 3\}$, $\{3, 2\}$ and $\{4, 2\}$, then Polycarp will be able to divide them into two sets in the required way. The first set can include the first and third dominoes ($\{1, 4\}$ and $\{3, 2\}$), and the second set — the second and fourth ones ($\{1, 3\}$ and $\{4, 2\}$).

Input
The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases.

The descriptions of the test cases follow.

The first line of each test case contains a single even integer n ($2 \le n \le 2 \cdot 10^5$) — the number of dominoes.

The next n lines contain pairs of numbers a_i and b_i ($1 \le a_i, b_i \le n$) describing the numbers on the i-th domino.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output
For each test case print:

- `YES`, if it is possible to divide n dominoes into two sets so that the numbers on the dominoes of each set are different;
- `NO` if this is not possible.

You can print `YES` and `NO` in any case (for example, the strings `yEs`, `yes`, `Yes` and `YES` will be recognized as a positive answer).

Example

input
6 4 1 2

```
4 3
2 1
3 4
6
1 2
4 5
1 3
4 6
2 3
5 6
2
1 1
2 2
2
1 2
2 1
8
2 1
1 2
4 3
4 3
5 6
5 7
8 6
7 8
8
1 2
2 1
4 3
5 3
5 4
6 7
8 6
7 8
```

output

```
YES
NO
NO
YES
YES
NO
```

Note

In the first test case, the dominoes can be divided as follows:

- First set of dominoes: $[\{1, 2\}, \{4, 3\}]$
- Second set of dominoes: $[\{2, 1\}, \{3, 4\}]$

In other words, in the first set we take dominoes with numbers 1 and 2, and in the second set we take dominoes with numbers 3 and 4.
In the second test case, there's no way to divide dominoes into 2 sets, at least one of them will contain repeated number.

F. Equate Multisets

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Multiset —is a set of numbers in which there can be equal elements, and the order of the numbers does not matter. Two multisets are equal when each value occurs the same number of times. For example, the multisets $\{2, 2, 4\}$ and $\{2, 4, 2\}$ are equal, but the multisets $\{1, 2, 2\}$ and $\{1, 1, 2\}$ — are not.

You are given two multisets a and b, each consisting of n integers.

In a single operation, any element of the b multiset can be doubled or halved (rounded down). In other words, you have one of the following operations available for an element x of the b multiset:

- replace x with $x \cdot 2$,
- or replace x with $\lfloor \frac{x}{2} \rfloor$ (round down).

Note that you cannot change the elements of the a multiset.

See if you can make the multiset b become equal to the multiset a in an arbitrary number of operations (maybe 0).

For example, if $n = 4$, $a = \{4, 24, 5, 2\}$, $b = \{4, 1, 6, 11\}$, then the answer is yes. We can proceed as follows:

- Replace 1 with $1 \cdot 2 = 2$. We get $b = \{4, 2, 6, 11\}$.
- Replace 11 with $\lfloor \frac{11}{2} \rfloor = 5$. We get $b = \{4, 2, 6, 5\}$.
- Replace 6 with $6 \cdot 2 = 12$. We get $b = \{4, 2, 12, 5\}$.
- Replace 12 with $12 \cdot 2 = 24$. We get $b = \{4, 2, 24, 5\}$.
- Got equal multisets $a = \{4, 24, 5, 2\}$ and $b = \{4, 2, 24, 5\}$.

Input
The first line of input data contains a single integer t ($1 \le t \le 10^4$) —the number of test cases.

Each test case consists of three lines.

The first line of the test case contains an integer n ($1 \le n \le 2 \cdot 10^5$) —the number of elements in the multisets a and b.

The second line gives n integers: $a_1, a_2, \ldots, a_n$ ($1 \le a_1 \le a_2 \le \cdots \le a_n \le 10^9$) —the elements of the multiset a. Note that the elements may be equal.

The third line contains n integers: $b_1, b_2, \ldots, b_n$ ($1 \le b_1 \le b_2 \le \cdots \le b_n \le 10^9$) — elements of the multiset b. Note that the elements may be equal.

It is guaranteed that the sum of n values over all test cases does not exceed $2 \cdot 10^5$.

Output
For each test case, print on a separate line:

- YES if you can make the multiset b become equal to a,
- NO otherwise.

You can output YES and NO in any case (for example, strings yEs, yes, Yes and YES will be recognized as positive answer).

Example

input
5 4 2 4 5 24 1 4 6 11 3 1 4 17 4 5 31 5 4 7 10 13 14 2 14 14 26 42 5 2 2 4 4 4 28 46 62 71 98 6 1 2 10 16 64 80 20 43 60 74 85 99
output
YES NO YES YES YES

Note
The first example is explained in the statement.

In the second example, it is impossible to get the value 31 from the numbers of the multiset b by available operations.

In the third example, we can proceed as follows:

- Replace 2 with $2 \cdot 2 = 4$. We get $b = \{4, 14, 14, 26, 42\}$.
- Replace 14 with $\lfloor \frac{14}{2} \rfloor = 7$. We get $b = \{4, 7, 14, 26, 42\}$.
- Replace 26 with $\lfloor \frac{26}{2} \rfloor = 13$. We get $b = \{4, 7, 14, 13, 42\}$.
- Replace 42 with $\lfloor \frac{42}{2} \rfloor = 21$. We get $b = \{4, 7, 14, 13, 21\}$.
- Replace 21 with $\lfloor \frac{21}{2} \rfloor = 10$. We get $b = \{4, 7, 14, 13, 10\}$.
- Got equal multisets $a = \{4, 7, 10, 13, 14\}$ and $b = \{4, 7, 14, 13, 10\}$.

G1. Passable Paths (easy version)

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

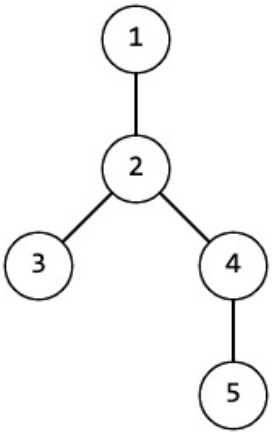
This is an easy version of the problem. The only difference between an easy and a hard version is in the number of queries.

Polycarp grew a tree from n vertices. We remind you that a tree of n vertices is an undirected connected graph of n vertices and $n-1$ edges that does not contain cycles.

He calls a set of vertices *passable* if there is such a path in the tree that passes through each vertex of this set without passing through any edge twice. The path can visit other vertices (not from this set).

In other words, a set of vertices is called *passable* if there is a simple path that passes through all the vertices of this set (and possibly some other).

For example, for a tree below sets $\{3, 2, 5\}$, $\{1, 5, 4\}$, $\{1, 4\}$ are *passable*, and $\{1, 3, 5\}$, $\{1, 2, 3, 4, 5\}$ are not.



Polycarp asks you to answer q queries. Each query is a set of vertices. For each query, you need to determine whether the corresponding set of vertices is **passable**.

Input

The first line of input contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — number of vertices.

Following $n - 1$ lines a description of the tree..

Each line contains two integers u and v ($1 \le u, v \le n$, $u \ne v$) — indices of vertices connected by an edge.

Following line contains single integer q ($1 \le q \le 5$) — number of queries.

The following $2 \cdot q$ lines contain descriptions of sets.

The first line of the description contains an integer k ($1 \le k \le n$) — the size of the set.

The second line of the description contains k of distinct integers $p_1, p_2, \ldots, p_k$ ($1 \le p_i \le n$) — indices of the vertices of the set.

It is guaranteed that the sum of k values for all queries does not exceed $2 \cdot 10^5$.

Output

Output q lines, each of which contains the answer to the corresponding query. As an answer, output "YES" if the set is **passable**, and "NO" otherwise.

You can output the answer in any case (for example, the strings "yEs", "yes", "Yes" and "YES" will be recognized as a positive answer).

Examples

input
5 1 2 2 3 2 4 4 5 5 3 3 2 5 5 1 2 3 4 5 2 1 4 3 1 3 5 3 1 5 4

output
YES NO YES NO YES

input
5 1 2 3 2 2 4 5 2 4 2

```
3 1
3
3 4 5
3
2 3 5
1
1
```

output

```
YES
NO
YES
YES
```

G2. Passable Paths (hard version)

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

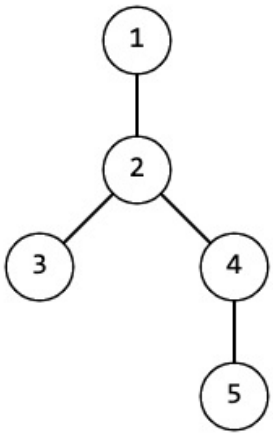
This is a hard version of the problem. The only difference between an easy and a hard version is in the number of queries.

Polycarp grew a tree from n vertices. We remind you that a tree of n vertices is an undirected connected graph of n vertices and $n-1$ edges that does not contain cycles.

He calls a set of vertices *passable* if there is such a path in the tree that passes through each vertex of this set without passing through any edge twice. The path can visit other vertices (not from this set).

In other words, a set of vertices is called *passable* if there is a simple path that passes through all the vertices of this set (and possibly some other).

For example, for a tree below sets $\{3, 2, 5\}$, $\{1, 5, 4\}$, $\{1, 4\}$ are *passable*, and $\{1, 3, 5\}$, $\{1, 2, 3, 4, 5\}$ are not.



Polycarp asks you to answer q queries. Each query is a set of vertices. For each query, you need to determine whether the corresponding set of vertices is **passable**.

Input

The first line of input contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — number of vertices.

Following $n-1$ lines a description of the tree..

Each line contains two integers u and v ($1 \le u, v \le n$, $u \ne v$) — indices of vertices connected by an edge.

Following line contains single integer q ($1 \le q \le 10^5$) — number of queries.

The following $2 \cdot q$ lines contain descriptions of sets.

The first line of the description contains an integer k ($1 \le k \le n$) — the size of the set.

The second line of the description contains k of distinct integers $p_1, p_2, \ldots, p_k$ ($1 \le p_i \le n$) — indices of the vertices of the set.

It is guaranteed that the sum of k values for all queries does not exceed $2 \cdot 10^5$.

Output

Output q lines, each of which contains the answer to the corresponding query. As an answer, output "YES" if the set is **passable**, and "NO" otherwise.

You can output the answer in any case (for example, the strings "yEs", "yes", "Yes" and "YES" will be recognized as a positive answer).

Examples

input

```
5
1 2
2 3
2 4
4 5
5
3
3 2 5
5
1 2 3 4 5
2
1 4
3
1 3 5
3
1 5 4
```

output

```
YES
NO
YES
NO
YES
```

input

```
5
1 2
3 2
2 4
5 2
4
2
3 1
3
3 4 5
3
2 3 5
1
1
```

output

```
YES
NO
YES
YES
```