Codeforces Round #656 (Div. 3)

A. Three Pairwise Maximums

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

You are given three positive (i.e. strictly greater than zero) integers x, y and z.

Your task is to find positive integers a, b and c such that $x = \max(a, b)$, $y = \max(a, c)$ and $z = \max(b, c)$, or determine that it is impossible to find such a, b and c.

You have to answer t independent test cases. Print required a, b and c in any (arbitrary) order.

Input

The first line of the input contains one integer t ($1 \le t \le 2 \cdot 10^4$) — the number of test cases. Then t test cases follow.

The only line of the test case contains three integers x, y, and z ($1 \le x, y, z \le 10^9$).

Output

For each test case, print the answer:

- "NO" in the only line of the output if a solution doesn't exist;
- or "YES" in the first line and **any** valid triple of positive integers a, b and c ($1 \le a, b, c \le 10^9$) in the second line. You can print a, b and c **in any order**.

Example

input
5 3 2 3 100 100 100 50 49 49 10 30 20 1 1000000000 1000000000
output
YES 3 2 1 YES 100 100 100 NO NO YES 1 1 1000000000

B. Restore the Permutation by Merger

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

A permutation of length n is a sequence of integers from 1 to n of length n containing each number exactly once. For example, $[1]$, $[4, 3, 5, 1, 2]$, $[3, 2, 1]$ are permutations, and $[1, 1]$, $[0, 1]$, $[2, 2, 1, 4]$ are not.

There was a permutation $p[1 \dots n]$. It was merged with itself. In other words, let's take two instances of p and insert elements of the second p into the first maintaining relative order of elements. The result is a sequence of the length $2n$.

For example, if $p = [3, 1, 2]$ some possible results are: $[3, 1, 2, 3, 1, 2]$, $[3, 3, 1, 1, 2, 2]$, $[3, 1, 3, 1, 2, 2]$. The following sequences are not possible results of a merging: $[1, 3, 2, 1, 2, 3]$, $[3, 1, 2, 3, 2, 1]$, $[3, 3, 1, 2, 2, 1]$.

For example, if $p = [2, 1]$ the possible results are: $[2, 2, 1, 1]$, $[2, 1, 2, 1]$. The following sequences are not possible results of a merging: $[1, 1, 2, 2]$, $[2, 1, 1, 2]$, $[1, 2, 2, 1]$.

Your task is to restore the permutation p by the given resulting sequence a. It is guaranteed that the answer **exists and is unique**.

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \le t \le 400$) — the number of test cases. Then t test cases follow.

The first line of the test case contains one integer n ($1 \le n \le 50$) — the length of permutation. The second line of the test case contains $2n$ integers $a_1, a_2, \ldots, a_{2n}$ ($1 \le a_i \le n$), where a_i is the i-th element of a. It is guaranteed that the array a represents the result of merging of some permutation p with the same permutation p.

Output

For each test case, print the answer: n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$), representing the initial permutation. It is guaranteed that the answer **exists and is unique**.

Example

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

C. Make It Good

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

You are given an array a consisting of n integers. You have to find the length of the smallest (shortest) *prefix* of elements you need to erase from a to make it a *good* array. Recall that the prefix of the array $a = [a_1, a_2, \ldots, a_n]$ is a subarray consisting several first elements: the prefix of the array a of length k is the array $[a_1, a_2, \ldots, a_k]$ ($0 \le k \le n$).

The array b of length m is called *good*, if you can obtain a **non-decreasing** array c ($c_1 \le c_2 \le \cdots \le c_m$) from it, repeating the following operation m times (initially, c is empty):

- select either the first or the last element of b, remove it from b, and append it to the end of the array c.

For example, if we do 4 operations: take b_1, then b_m, then b_{m-1} and at last b_2, then b becomes $[b_3, b_4, \ldots, b_{m-3}]$ and $c = [b_1, b_m, b_{m-1}, b_2]$.

Consider the following example: $b = [1, 2, 3, 4, 4, 2, 1]$. This array is **good** because we can obtain **non-decreasing** array c from it by the following sequence of operations:

1. take the first element of b, so $b = [2, 3, 4, 4, 2, 1]$, $c = [1]$;
2. take the last element of b, so $b = [2, 3, 4, 4, 2]$, $c = [1, 1]$;
3. take the last element of b, so $b = [2, 3, 4, 4]$, $c = [1, 1, 2]$;
4. take the first element of b, so $b = [3, 4, 4]$, $c = [1, 1, 2, 2]$;
5. take the first element of b, so $b = [4, 4]$, $c = [1, 1, 2, 2, 3]$;
6. take the last element of b, so $b = [4]$, $c = [1, 1, 2, 2, 3, 4]$;
7. take the only element of b, so $b = []$, $c = [1, 1, 2, 2, 3, 4, 4]$ — c is non-decreasing.

Note that the array consisting of one element is *good*.

Print the length of the shortest prefix of a to delete (erase), to make a to be a *good* array. Note that the required length can be 0.

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \le t \le 2 \cdot 10^4$) — the number of test cases. Then t test cases follow.

The first line of the test case contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the length of a. The second line of the test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 2 \cdot 10^5$), where a_i is the i-th element of a.

It is guaranteed that the sum of n does not exceed $2 \cdot 10^5$ ($\sum n \le 2 \cdot 10^5$).

Output

For each test case, print the answer: the length of the shortest *prefix* of elements you need to erase from a to make it a *good* array.

Example

input

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

output

```
0
4
0
2
3
```

Note

In the first test case of the example, the array a is already good, so we don't need to erase any prefix.

In the second test case of the example, the initial array a is not good. Let's erase first 4 elements of a, the result is $[4, 5, 2]$. The resulting array is good. You can prove that if you erase fewer number of first elements, the result will not be good.

D. a-Good String

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

You are given a string $s[1 \ldots n]$ consisting of lowercase Latin letters. It is guaranteed that $n = 2^k$ for some integer $k \geq 0$.

The string $s[1 \ldots n]$ is called *c-good* if **at least one** of the following three conditions is satisfied:

- The length of s is 1, and it consists of the character c (i.e. $s_1 = c$);
- The length of s is greater than 1, the first half of the string consists of only the character c (i.e. $s_1 = s_2 = \cdots = s_{\frac{n}{2}} = c$) and the second half of the string (i.e. the string $s_{\frac{n}{2}+1} s_{\frac{n}{2}+2} \ldots s_n$) is a $(c+1)$-*good* string;
- The length of s is greater than 1, the second half of the string consists of only the character c (i.e. $s_{\frac{n}{2}+1} = s_{\frac{n}{2}+2} = \cdots = s_n = c$) and the first half of the string (i.e. the string $s_1 s_2 \ldots s_{\frac{n}{2}}$) is a $(c+1)$-*good* string.

For example: "aabc" is *'a'-good*, "ffgheeee" is *'e'-good*.

In one move, you can choose one index i from 1 to n and replace s_i with any lowercase Latin letter (any character from 'a' to 'z').

Your task is to find the minimum number of moves required to obtain an *'a'-good* string from s (i.e. *c-good string* for $c =$ 'a'). It is guaranteed that the answer always exists.

You have to answer t independent test cases.

Another example of an *'a'-good* string is as follows. Consider the string $s =$"cdbbaaaa". It is an *'a'-good* string, because:

- the second half of the string ("aaaa") consists of only the character 'a';
- the first half of the string ("cdbb") is *'b'-good* string, because:
  - the second half of the string ("bb") consists of only the character 'b';
  - the first half of the string ("cd") is *'c'-good* string, because:
    - the first half of the string ("c") consists of only the character 'c';
    - the second half of the string ("d") is *'d'-good* string.

Input

The first line of the input contains one integer t ($1 \leq t \leq 2 \cdot 10^4$) — the number of test cases. Then t test cases follow.

The first line of the test case contains one integer n ($1 \leq n \leq 131\,072$) — the length of s. It is guaranteed that $n = 2^k$ for some integer $k \geq 0$. The second line of the test case contains the string s consisting of n lowercase Latin letters.

It is guaranteed that the sum of n does not exceed $2 \cdot 10^5$ ($\sum n \leq 2 \cdot 10^5$).

Output

For each test case, print the answer — the minimum number of moves required to obtain an *'a'-good* string from s (i.e. *c-good string* with $c =$ 'a'). It is guaranteed that the answer exists.

Example

input

```
6
8
```

bbdcaaaa
8
asdfghjk
8
ceaaaabb
8
bbaaddcc
1
z
2
ac

output

0
7
4
5
1
1

E. Directing Edges

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

You are given a graph consisting of n vertices and m edges. It is not guaranteed that the given graph is connected. Some edges are already directed and you can't change their direction. Other edges are undirected and you have to choose some direction for all these edges.

You have to direct undirected edges in such a way that the resulting graph is directed and acyclic (i.e. the graph with all edges directed and having no directed cycles). Note that you have to direct **all** undirected edges.

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \le t \le 2 \cdot 10^4$) — the number of test cases. Then t test cases follow.

The first line of the test case contains two integers n and m ($2 \le n \le 2 \cdot 10^5$, $1 \le m \le min(2 \cdot 10^5, \frac{n(n-1)}{2})$) — the number of vertices and the number of edges in the graph, respectively.

The next m lines describe edges of the graph. The i-th edge is described with three integers t_i, x_i and y_i ($t_i \in [0; 1]$, $1 \le x_i, y_i \le n$) — the type of the edge ($t_i = 0$ if the edge is undirected and $t_i = 1$ if the edge is directed) and vertices this edge connects (the undirected edge connects vertices x_i and y_i and directed edge is going from the vertex x_i to the vertex y_i). It is guaranteed that the graph do not contain self-loops (i.e. edges from the vertex to itself) and multiple edges (i.e. for each pair (x_i, y_i) there are no other pairs (x_i, y_i) or (y_i, x_i)).

It is guaranteed that both sum n and sum m do not exceed $2 \cdot 10^5$ ($\sum n \le 2 \cdot 10^5$; $\sum m \le 2 \cdot 10^5$).

Output

For each test case print the answer — "NO" if it is impossible to direct undirected edges in such a way that the resulting graph is directed and acyclic, otherwise print "YES" on the first line and m lines describing edges of the resulted directed acyclic graph (**in any order**). Note that you cannot change the direction of the already directed edges. If there are several answers, you can print any.

Example

input

4
3 1
0 1 3
5 5
0 2 1
1 1 5
1 5 4
0 5 2
1 3 5
4 5
1 1 2
0 4 3
1 3 1
0 2 3
1 2 4
4 5
1 4 1
1 1 3
0 1 2
1 2 4
1 3 2

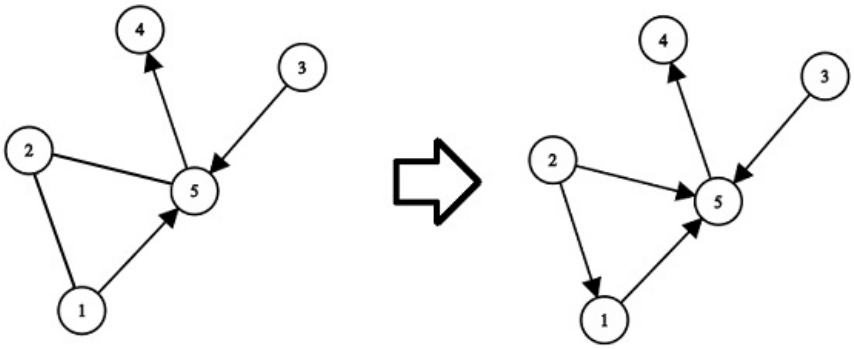
output

YES

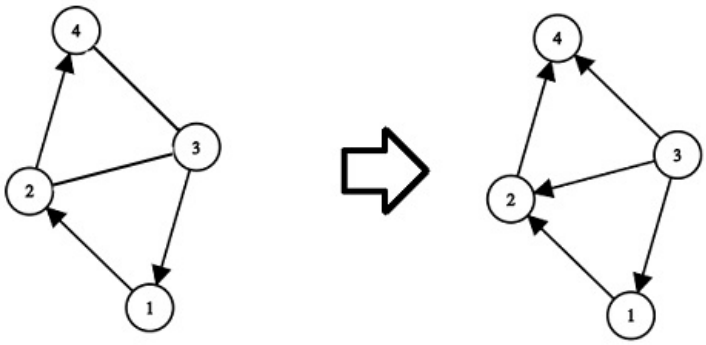
```
3 1
YES
2 1
1 5
5 4
2 5
3 5
YES
1 2
3 4
3 1
3 2
2 4
NO
```

Note

Explanation of the second test case of the example:



Explanation of the third test case of the example:



F. Removing Leaves

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

You are given a tree (connected graph without cycles) consisting of n vertices. The tree is unrooted — it is just a connected undirected graph without cycles.

In one move, you can choose exactly k leaves (leaf is such a vertex that is connected to only one another vertex) connected **to the same vertex** and remove them with edges incident to them. I.e. you choose such leaves $u_1, u_2, \ldots, u_k$ that there are edges (u_1, v), (u_2, v), ..., (u_k, v) and remove these leaves and these edges.

Your task is to find the **maximum** number of moves you can perform if you remove leaves optimally.

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \le t \le 2 \cdot 10^4$) — the number of test cases. Then t test cases follow.

The first line of the test case contains two integers n and k ($2 \le n \le 2 \cdot 10^5$; $1 \le k < n$) — the number of vertices in the tree and the number of leaves you remove in one move, respectively. The next $n - 1$ lines describe edges. The i-th edge is represented as two integers x_i and y_i ($1 \le x_i, y_i \le n$), where x_i and y_i are vertices the i-th edge connects. It is guaranteed that the given set of edges forms a tree.

It is guaranteed that the sum of n does not exceed $2 \cdot 10^5$ ($\sum n \le 2 \cdot 10^5$).

Output

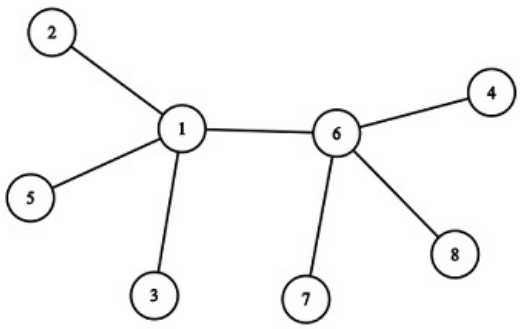
For each test case, print the answer — the **maximum** number of moves you can perform if you remove leaves optimally.

Example

input
4 8 3 1 2 1 5 7 6 6 8 3 1 6 4 6 1 10 3 1 2 1 10 2 3 1 5 1 6 2 4 7 10 10 9 8 10 7 2 3 1 4 5 3 6 7 4 1 2 1 4 5 1 1 2 2 3 4 3 5 3
output
2 3 3 4

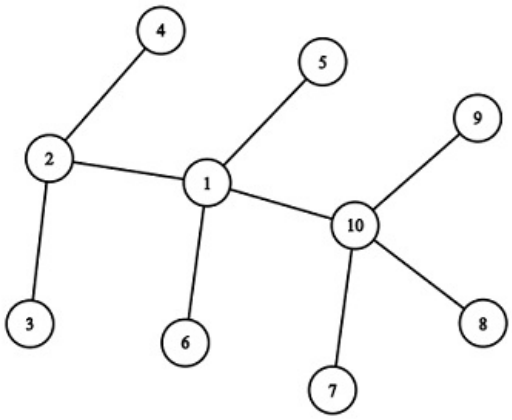
Note

The picture corresponding to the first test case of the example:



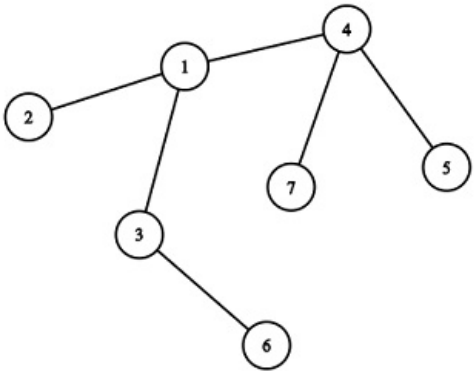
There you can remove vertices 2, 5 and 3 during the first move and vertices 1, 7 and 4 during the second move.

The picture corresponding to the second test case of the example:



There you can remove vertices 7, 8 and 9 during the first move, then vertices 5, 6 and 10 during the second move and vertices 1, 3 and 4 during the third move.

The picture corresponding to the third test case of the example:



There you can remove vertices 5 and 7 during the first move, then vertices 2 and 4 during the second move and vertices 1 and 6 during the third move.

G. Columns Swaps

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

You are given a table a of size $2 \times n$ (i.e. two rows and n columns) consisting of integers from 1 to n.

In one move, you can choose some **column** j ($1 \le j \le n$) and swap values $a_{1,j}$ and $a_{2,j}$ in it. Each column can be chosen **no more than once**.

Your task is to find the **minimum** number of moves required to obtain permutations of size n in both first and second rows of the table or determine if it is impossible to do that.

You have to answer t independent test cases.

Recall that the permutation of size n is such an array of size n that contains *each integer* from 1 to n exactly once (the order of elements doesn't matter).

Input

The first line of the input contains one integer t ($1 \le t \le 2 \cdot 10^4$) — the number of test cases. Then t test cases follow.

The first line of the test case contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of columns in the table. The second line of the test case contains n integers $a_{1,1}, a_{1,2}, \ldots, a_{1,n}$ ($1 \le a_{1,i} \le n$), where $a_{1,i}$ is the i-th element of the first row of the table. The third line of the test case contains n integers $a_{2,1}, a_{2,2}, \ldots, a_{2,n}$ ($1 \le a_{2,i} \le n$), where $a_{2,i}$ is the i-th element of the second row of the table.

It is guaranteed that the sum of n does not exceed $2 \cdot 10^5$ ($\sum n \le 2 \cdot 10^5$).

Output

For each test case print the answer: -1 if it is impossible to obtain permutation of size n in both first and the second rows of the table, or one integer k in the first line, where k is the **minimum** number of moves required to obtain permutations in both rows, and k *distinct* integers $pos_1, pos_2, \ldots, pos_k$ in the second line ($1 \le pos_i \le n$) in *any order* — indices of columns in which you need to swap values to obtain permutations in both rows. If there are several answers, you can print any.

Example

input
6 4 1 2 3 4 2 3 1 4 5 5 3 5 1 4 1 2 3 2 4 3 1 2 1 3 3 2 4 1 2 2 1 3 4 3 4 4 4 3 1 4 3 2 2 1 3 1 1 2 3 2 2
output
0

```
2
2 3
1
1
2
3 4
2
3 4
-1
```