

A. Binary Literature, 1 second, 256 megabytes, standard input, standard output

A bitstring is a string that contains only the characters 0 and 1.

Koyomi Kanou is working hard towards her dream of becoming a writer. To practice, she decided to participate in the *Binary Novel Writing Contest*. The writing prompt for the contest consists of three bitstrings of length $2n$. A valid novel for the contest is a bitstring of length at most $3n$ that contains **at least two** of the three given strings as subsequences.

Koyomi has just received the three prompt strings from the contest organizers. Help her write a valid novel for the contest.

A string a is a subsequence of a string b if a can be obtained from b by deletion of several (possibly, zero) characters.

Input

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

The first line of each test case contains a single integer n ($1 \leq n \leq 10^5$).

Each of the following three lines contains a bitstring of length $2n$. It is guaranteed that these three strings are pairwise distinct.

It is guaranteed that the sum of n across all test cases does not exceed 10^5 .

Output

For each test case, print a single line containing a bitstring of length at most $3n$ that has at least two of the given bitstrings as subsequences.

It can be proven that under the constraints of the problem, such a bitstring always exists.

If there are multiple possible answers, you may output any of them.

input
2 1 00 11 01 3 011001 111010 010001
output
010 011001010

In the first test case, the bitstrings 00 and 01 are subsequences of the output string: 010 and 010. Note that 11 is not a subsequence of the output string, but this is not required.

In the second test case all three input strings are subsequences of the output string: 011001010, 011001010 and 011001010.

B. Almost Sorted, 2 seconds, 256 megabytes, standard input, standard output

Seiji Maki doesn't only like to observe relationships being unfolded, he also likes to observe sequences of numbers, especially permutations. Today, he has his eyes on *almost sorted* permutations.

A permutation a_1, a_2, \dots, a_n of $1, 2, \dots, n$ is said to be *almost sorted* if the condition $a_{i+1} \geq a_i - 1$ holds for all i between 1 and $n - 1$ inclusive.

Maki is considering the list of all almost sorted permutations of $1, 2, \dots, n$, given in lexicographical order, and he wants to find the k -th permutation in this list. Can you help him to find such permutation?

Permutation p is lexicographically smaller than a permutation q if and only if the following holds:

- in the first position where p and q differ, the permutation p has a smaller element than the corresponding element in q .

Input

The first line contains a single integer t ($1 \leq t \leq 1000$) — the number of test cases.

Each test case consists of a single line containing two integers n and k ($1 \leq n \leq 10^5, 1 \leq k \leq 10^{18}$).

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

Output

For each test case, print a single line containing the k -th almost sorted permutation of length n in lexicographical order, or -1 if it doesn't exist.

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

For the first and second test, the list of almost sorted permutations with $n = 1$ is $\{[1]\}$.

For the third and fifth test, the list of almost sorted permutations with $n = 3$ is $\{[1, 2, 3], [1, 3, 2], [2, 1, 3], [3, 2, 1]\}$.

C. Complete the MST, 3 seconds, 256 megabytes, standard input, standard output

As a teacher, Riko Hakozaki often needs to help her students with problems from various subjects. Today, she is asked a programming task which goes as follows.

You are given an undirected complete graph with n nodes, where some edges are pre-assigned with a positive weight while the rest aren't. You need to assign all unassigned edges with **non-negative weights** so that in the resulting fully-assigned complete graph the **XOR** sum of all weights would be equal to 0.

Define the *ugliness* of a fully-assigned complete graph the weight of its **minimum spanning tree**, where the weight of a spanning tree equals the sum of weights of its edges. You need to assign the weights so that the ugliness of the resulting graph is as small as possible.

As a reminder, an undirected complete graph with n nodes contains all edges (u, v) with $1 \leq u < v \leq n$; such a graph has $\frac{n(n-1)}{2}$ edges.

She is not sure how to solve this problem, so she asks you to solve it for her.

Input

The first line contains two integers n and m ($2 \leq n \leq 2 \cdot 10^5, 0 \leq m \leq \min(2 \cdot 10^5, \frac{n(n-1)}{2} - 1)$) — the number of nodes and the number of pre-assigned edges. The inputs are given so that there is at least one unassigned edge.

The i -th of the following m lines contains three integers u_i, v_i , and w_i ($1 \leq u_i, v_i \leq n, u_i \neq v_i, 1 \leq w_i < 2^{30}$), representing the edge from u_i to v_i has been pre-assigned with the weight w_i . No edge appears in the input more than once.

Output

Print on one line one integer — the minimum ugliness of all assignments with XOR sum equal to 0.

🌐 online now: 0

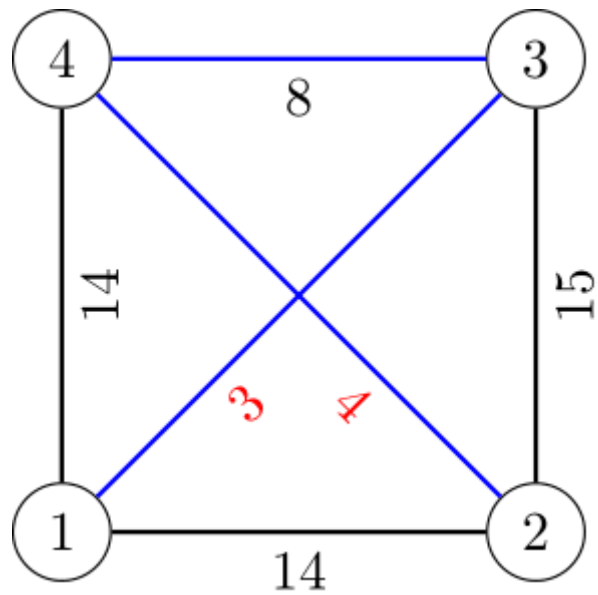
★ online now: 🌀

input
4 4 2 1 14 1 4 14 3 2 15 4 3 8
output
15

input
6 6 3 6 4 2 4 1 4 5 7 3 4 10 3 5 1 5 2 15
output
0

input
5 6 2 3 11 5 3 7 1 4 10 2 4 14 4 3 8 2 5 6
output
6

The following image showcases the first test case. The black weights are pre-assigned from the statement, the red weights are assigned by us, and the minimum spanning tree is denoted by the blue edges.



D. Swap Pass, 2 seconds, 256 megabytes, standard input, standard output

Based on a peculiar incident at basketball practice, Akari came up with the following competitive programming problem!

You are given n points on the plane, no three of which are collinear. The i -th point initially has a label a_i , in such a way that the labels a_1, a_2, \dots, a_n form a permutation of $1, 2, \dots, n$.

You are allowed to modify the labels through the following operation:

- Choose two distinct integers i and j between 1 and n .
- Swap the labels of points i and j , and finally
- Draw the segment between points i and j .

A sequence of operations is valid if after applying all of the operations in the sequence in order, the k -th point ends up having the label k for all k between 1 and n inclusive, and the drawn segments don't intersect each other internally. Formally, if two of the segments intersect, then they must do so at a common endpoint of both segments.

In particular, all drawn segments must be distinct.

Find any valid sequence of operations, or say that none exist.

Input

The first line contains an integer n ($3 \leq n \leq 2000$) — the number of points.

The i -th of the following n lines contains three integers x_i, y_i, a_i ($-10^6 \leq x_i, y_i \leq 10^6, 1 \leq a_i \leq n$), representing that the i -th point has coordinates (x_i, y_i) and initially has label a_i .

It is guaranteed that all points are distinct, no three points are collinear, and the labels a_1, a_2, \dots, a_n form a permutation of $1, 2, \dots, n$.

Output

If it is impossible to perform a valid sequence of operations, print -1 .

Otherwise, print an integer k ($0 \leq k \leq \frac{n(n-1)}{2}$) — the number of operations to perform, followed by k lines, each containing two integers i and j ($1 \leq i, j \leq n, i \neq j$) — the indices of the points chosen for the operation.

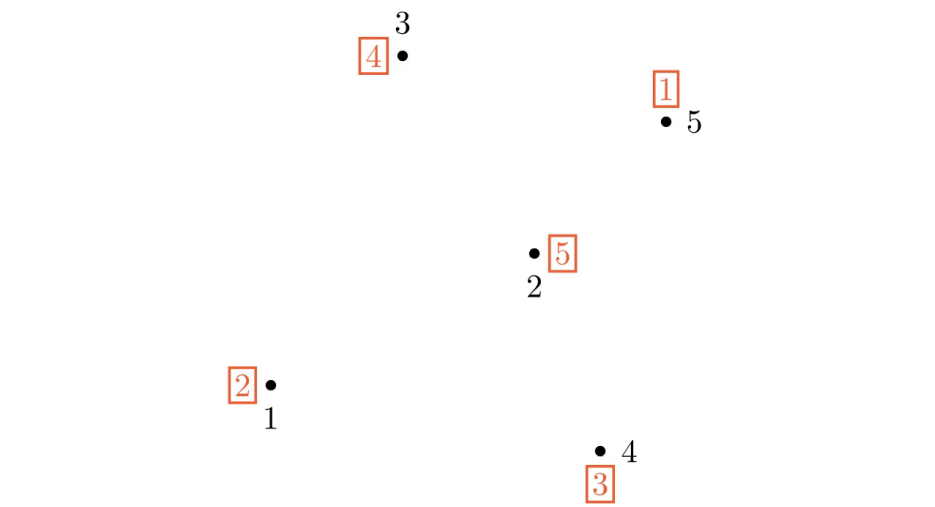
Note that you are **not** required to minimize or maximize the value of k .

If there are multiple possible answers, you may print any of them.

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

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

The following animation showcases the first sample test case. The black numbers represent the indices of the points, while the boxed orange numbers represent their labels.



In the second test case, all labels are already in their correct positions, so no operations are necessary.

E. Tree Calendar, 2 seconds, 512 megabytes, standard input, standard output

Yuu Koito and Touko Nanami are newlyweds! On the wedding day, Yuu gifted Touko a directed tree with n nodes and rooted at 1, and a labeling a which is *some* DFS order of the tree. Every edge in this tree is directed away from the root.

After calling `dfs(1)` the following algorithm returns a as a DFS order of a tree rooted at 1 :

```
order := 0
a := array of length n

function dfs(u):
    order := order + 1
    a[u] := order
    for all v such that there is a directed edge (u -> v):
        dfs(v)
```

Note that there may be different DFS orders for a given tree.

Touko likes the present so much she decided to play with it! On each day following the wedding day, Touko performs this procedure once:

- Among all directed edges $u \rightarrow v$ such that $a_u < a_v$, select the edge $u' \rightarrow v'$ with the lexicographically smallest pair $(a_{u'}, a_{v'})$.
- Swap $a_{u'}$ and $a_{v'}$.

Days have passed since their wedding, and Touko has somehow forgotten which date the wedding was and what was the original labeling a ! Fearing that Yuu might get angry, Touko decided to ask you to derive these two pieces of information using the current labeling.

Being her good friend, you need to find the number of days that have passed since the wedding, and the original labeling of the tree. However, there is a chance that Touko might have messed up her procedures, which result in the current labeling being impossible to obtain from some original labeling; in that case, please inform Touko as well.

Input

The first line of the input contains an integer n ($2 \leq n \leq 3 \cdot 10^5$) — the number of nodes on the tree.

The second line contains n integers a_1, a_2, \dots, a_n ($1 \leq a_i \leq n$, all a_i are distinct) — the current labeling of the tree.

Each of the next $n - 1$ lines contains two integers u_i and v_i ($1 \leq u, v \leq n, u \neq v$), describing an directed edge from u_i to v_i . The edges form a directed tree rooted at 1.

Output

If the current labeling is impossible to arrive at from any DFS order, print NO.

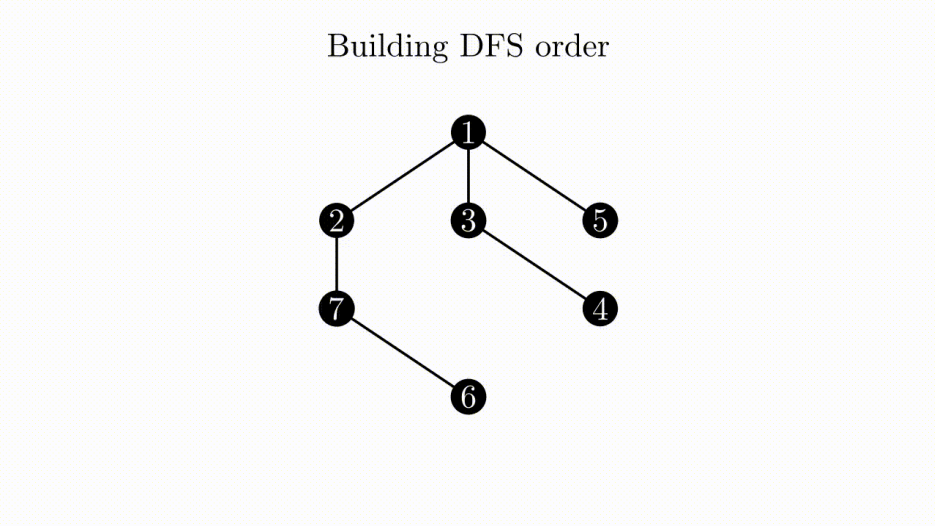
Else, on the first line, print YES. On the second line, print a single integer denoting the number of days since the wedding. On the third line, print n numbers space-separated denoting the original labeling of the tree.

If there are multiple correct outputs, print any. This means: you are allowed to output any pair (DFS order, number of days), such that we get the current configuration from the DFS order you provided in exactly the number of days you provided.

input
7 4 5 2 1 7 6 3 1 5 7 6 1 2 2 7 3 4 1 3
output
YES 5 1 4 2 3 7 6 5

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

The following animation showcases the first sample test case. The white label inside the node represents the index of the node i , while the boxed orange label represents the value a_i .



F. Optimal Encoding, 7 seconds, 1024 megabytes, standard input, standard output

Touko's favorite sequence of numbers is a permutation a_1, a_2, \dots, a_n of $1, 2, \dots, n$, and she wants some collection of permutations that are similar to her favorite permutation.

She has a collection of q intervals of the form $[l_i, r_i]$ with $1 \leq l_i \leq r_i \leq n$. To create permutations that are similar to her favorite permutation, she coined the following definition:

- A permutation b_1, b_2, \dots, b_n allows an interval $[l', r']$ to *holds its shape* if for any pair of integers (x, y) such that $l' \leq x < y \leq r'$, we have $b_x < b_y$ if and only if $a_x < a_y$.
- A permutation b_1, b_2, \dots, b_n is *k-similar* if b allows all intervals $[l_i, r_i]$ for all $1 \leq i \leq k$ to hold their shapes.

Yuu wants to figure out all k -similar permutations for Touko, but it turns out this is a very hard task; instead, Yuu will *encode* the set of all k -similar permutations with directed acyclic graphs (DAG). Yuu also coined the following definitions for herself:

- A permutation b_1, b_2, \dots, b_n *satisfies* a DAG G' if for all edge $u \rightarrow v$ in G' , we must have $b_u < b_v$.
- A *k-encoding* is a DAG G_k on the set of vertices $1, 2, \dots, n$ such that a permutation b_1, b_2, \dots, b_n satisfies G_k if and only if b is k -similar.

Since Yuu is free today, she wants to figure out the minimum number of edges among all k -encodings for each k from 1 to q .

Input

The first line contains two integers n and q ($1 \leq n \leq 25\,000$, $1 \leq q \leq 100\,000$).

The second line contains n integers a_1, a_2, \dots, a_n which form a permutation of $1, 2, \dots, n$.

The i -th of the following q lines contains two integers l_i and r_i . ($1 \leq l_i \leq r_i \leq n$).

Output

Print q lines. The k -th of them should contain a single integer — The minimum number of edges among all k -encodings.


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


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

input
10 10 10 5 1 2 7 3 9 4 6 8 2 2 4 5 6 8 4 10 4 4 2 7 2 2 7 8 3 7 2 10
output
0 1 3 6 6 9 9 9 9 8

For the first test case:

- All 1-similar permutations must allow the interval $[1, 3]$ to hold its shape. Therefore, the set of all 1-similar permutations is $\{[3, 4, 2, 1], [3, 4, 1, 2], [2, 4, 1, 3], [2, 3, 1, 4]\}$. The optimal encoding of these permutations is


- All 2-similar permutations must allow the intervals $[1, 3]$ and $[2, 4]$ to hold their shapes. Therefore, the set of all 2-similar permutations is $\{[3, 4, 1, 2], [2, 4, 1, 3]\}$. The optimal encoding of these permutations is


- All 3-similar permutations must allow the intervals $[1, 3]$, $[2, 4]$, and $[1, 4]$ to hold their shapes. Therefore, the set of all 3-similar permutations only includes $[2, 4, 1, 3]$. The optimal encoding of this permutation is

