Codeforces Round #572 (Div. 1)

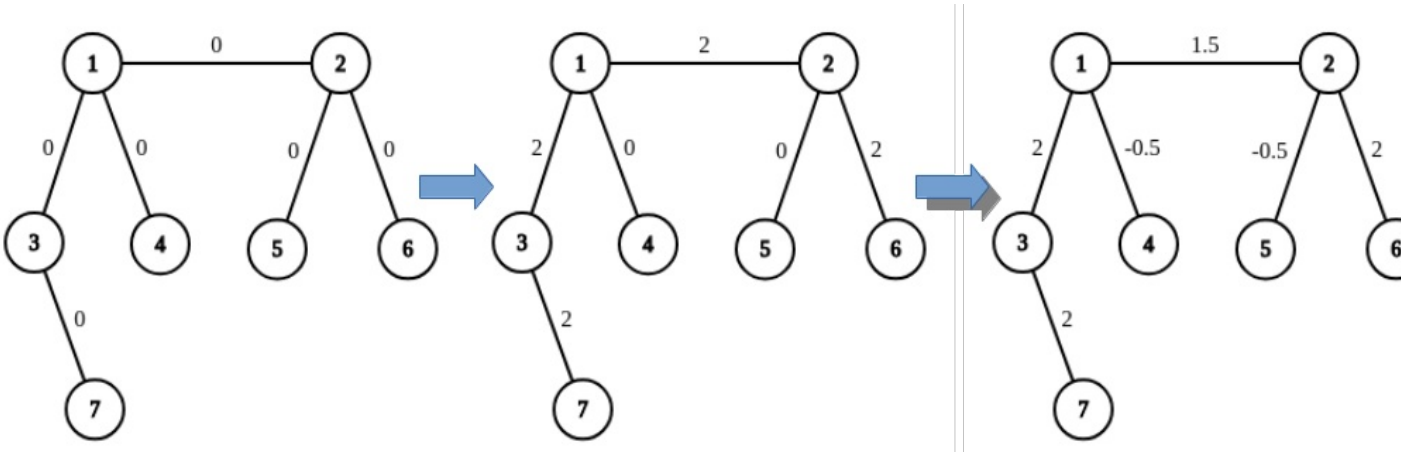
A1. Add on a Tree

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

Note that this is the first problem of the two similar problems. You can hack this problem only if you solve both problems.

You are given a tree with n nodes. In the beginning, 0 is written on all edges. In one operation, you can choose any 2 distinct **leaves** u, v and any **real** number x and add x to values written on all edges on the simple path between u and v.

For example, on the picture below you can see the result of applying two operations to the graph: adding 2 on the path from 7 to 6, and then adding -0.5 on the path from 4 to 5.



Is it true that for any configuration of real numbers written on edges, we can achieve it with a finite number of operations?

Leaf is a node of a tree of degree 1. Simple path is a path that doesn't contain any node twice.

Input

The first line contains a single integer n ($2 \le n \le 10^5$) — the number of nodes.

Each of the next $n-1$ lines contains two integers u and v ($1 \le u, v \le n$, $u \ne v$), meaning that there is an edge between nodes u and v. It is guaranteed that these edges form a tree.

Output

If there is a configuration of real numbers written on edges of the tree that we can't achieve by performing the operations, output "NO".

Otherwise, output "YES".

You can print each letter in any case (upper or lower).

Examples

input
2 1 2
output
YES

input
3 1 2 2 3
output
NO

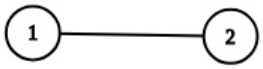
input
5 1 2 1 3 1 4 2 5
output
NO

input
6 1 2 1 3 1 4 2 5 2 6

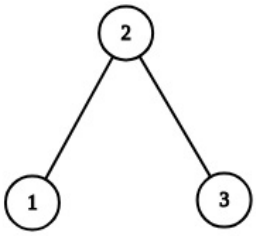
output
YES

Note

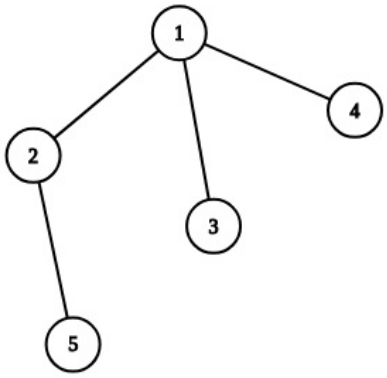
In the first example, we can add any real x to the value written on the only edge $(1, 2)$.

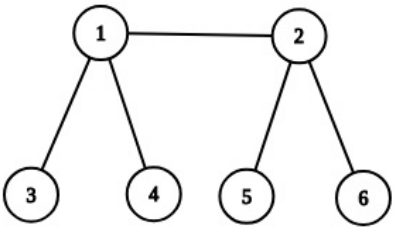


In the second example, one of configurations that we can't reach is 0 written on $(1, 2)$ and 1 written on $(2, 3)$.



Below you can see graphs from examples 3, 4:





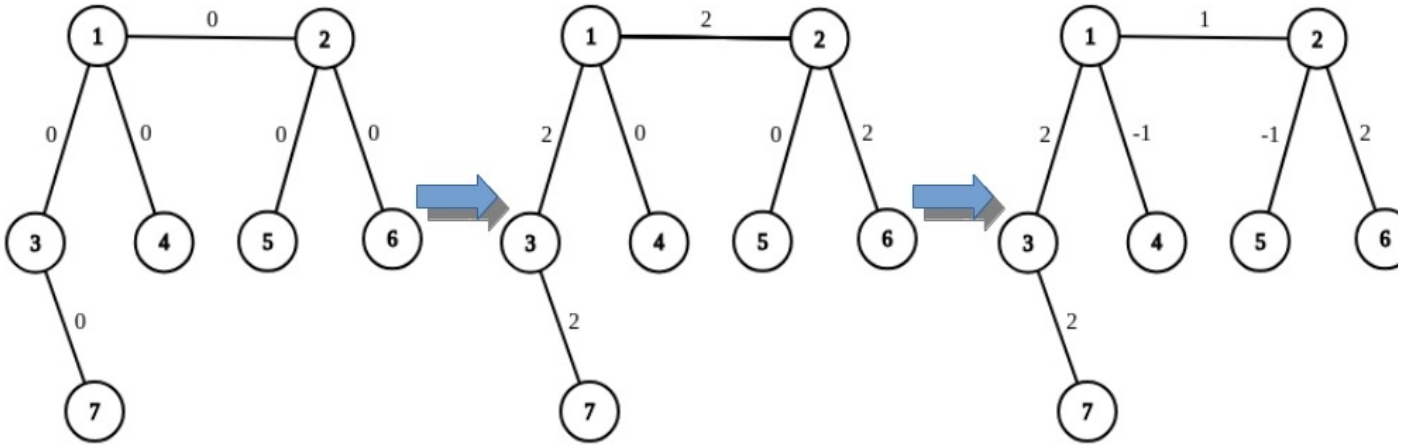
A2. Add on a Tree: Revolution

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

Note that this is the second problem of the two similar problems. You can hack this problem if you solve it. But you can hack the previous problem only if you solve both problems.

You are given a tree with n nodes. In the beginning, 0 is written on all edges. In one operation, you can choose any 2 distinct **leaves** u, v and any **integer** number x and add x to values written on all edges on the simple path between u and v. **Note that in previous subtask x was allowed to be any real, here it has to be integer**.

For example, on the picture below you can see the result of applying two operations to the graph: adding 2 on the path from 7 to 6, and then adding -1 on the path from 4 to 5.



You are given some configuration of **nonnegative integer pairwise different even** numbers, written on the edges. For a given configuration determine if it is possible to achieve it with these operations, and, if it is possible, output the sequence of operations that leads to the given configuration. Constraints on the operations are listed in the output format section.

Leave is a node of a tree of degree 1. Simple path is a path that doesn't contain any node twice.

Input

The first line contains a single integer n ($2 \le n \le 1000$) — the number of nodes in a tree.

Each of the next $n-1$ lines contains three integers u, v, val ($1 \le u, v \le n$, $u \ne v$, $0 \le val \le 10\,000$), meaning that there is an edge between nodes u and v with val written on it. It is guaranteed that these edges form a tree. It is guaranteed that all val numbers are **pairwise different and even**.

Output

If there aren't any sequences of operations which lead to the given configuration, output "NO".

If it exists, output "YES" in the first line. In the second line output m — number of operations you are going to apply ($0 \le m \le 10^5$). **Note that you don't have to minimize the number of the operations!**

In the next m lines output the operations in the following format:

u, v, x ($1 \le u, v \le n$, $u \ne v$, x — integer, $-10^9 \le x \le 10^9$), where u, v — leaves, x — number we are adding.

It is guaranteed that if there exists a sequence of operations producing given configuration, then there exists a sequence of operations producing given configuration, satisfying all the conditions above.

Examples

input
5 1 2 2

2 3 4
3 4 10
3 5 18

output

NO

input

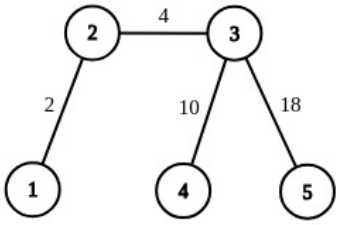
6
1 2 6
1 3 8
1 4 12
2 5 2
2 6 4

output

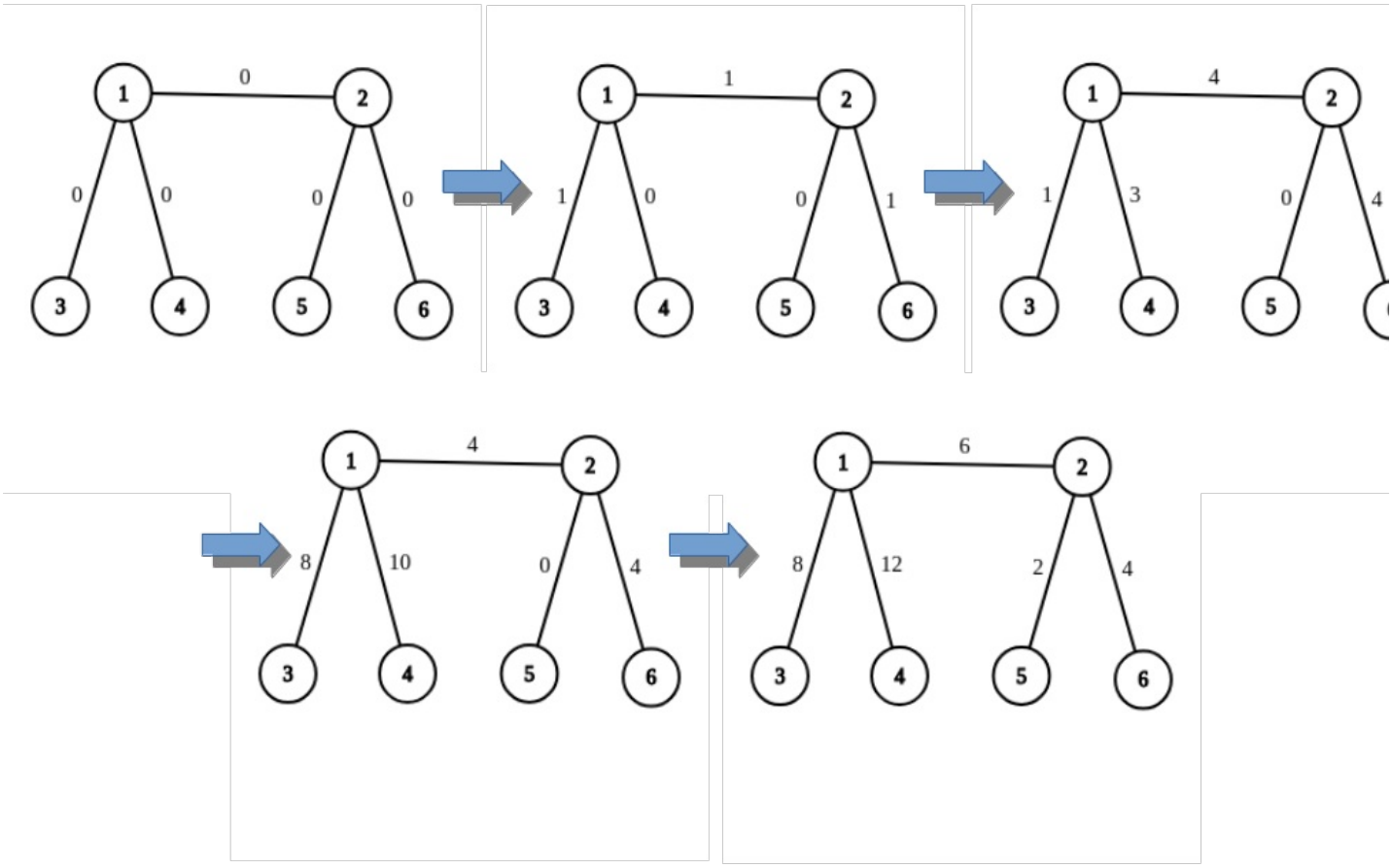
YES
4
3 6 1
4 6 3
3 4 7
4 5 2

Note

The configuration from the first sample is drawn below, and it is impossible to achieve.



The sequence of operations from the second sample is illustrated below.



B. Count Pairs

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

You are given a **prime** number p, n integers $a_1, a_2, \ldots, a_n$, and an integer k.

Find the number of pairs of indexes (i, j) ($1 \le i < j \le n$) for which $(a_i + a_j)(a_i^2 + a_j^2) \equiv k \bmod p$.

Input

The first line contains integers n, p, k ($2 \le n \le 3 \cdot 10^5$, $2 \le p \le 10^9$, $0 \le k \le p - 1$). p is guaranteed to be prime.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le p - 1$). It is guaranteed that all elements are different.

Output

Output a single integer — answer to the problem.

Examples

input
3 3 0 0 1 2
output
1

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

Note

In the first example:

$(0+1)(0^2+1^2) = 1 \equiv 1 \bmod 3$.

$(0+2)(0^2+2^2) = 8 \equiv 2 \bmod 3$.

$(1+2)(1^2+2^2) = 15 \equiv 0 \bmod 3$.

So only 1 pair satisfies the condition.

In the second example, there are 3 such pairs: $(1, 5)$, $(2, 3)$, $(4, 6)$.

C. Array Beauty

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Let's call beauty of an array $b_1, b_2, \ldots, b_n$ $(n > 1)$ — $\min\limits_{1 \le i < j \le n} |b_i - b_j|$.

You're given an array $a_1, a_2, \ldots a_n$ and a number k. Calculate the sum of beauty over all subsequences of the array of length exactly k. As this number can be very large, output it modulo 998244353.

A sequence a is a subsequence of an array b if a can be obtained from b by deletion of several (possibly, zero or all) elements.

Input
The first line contains integers n, k $(2 \le k \le n \le 1000)$.

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(0 \le a_i \le 10^5)$.

Output
Output one integer — the sum of beauty over all subsequences of the array of length exactly k. As this number can be very large, output it modulo 998244353.

Examples

input
4 3 1 7 3 5
output
8

input
5 5 1 10 100 1000 10000
output
9

Note
In the first example, there are 4 subsequences of length 3 — $[1, 7, 3]$, $[1, 3, 5]$, $[7, 3, 5]$, $[1, 7, 5]$, each of which has beauty 2, so answer is 8.

In the second example, there is only one subsequence of length 5 — the whole array, which has the beauty equal to $|10 - 1| = 9$.

D. Make Equal

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

You are given n numbers $a_1, a_2, \ldots, a_n$. In one operation we can add to any one of those numbers a nonnegative integer power of 2.

What is the smallest number of operations we need to perform to make all n numbers equal? It can be proved that under given constraints it doesn't exceed 10^{18}.

Input
The first line contains a single integer n $(1 \le n \le 10^5)$.

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(0 \le a_i \le 10^{17})$.

Output
Output exactly one integer — the smallest number of operations we need to perform to make all n numbers equal.

Examples

input
4 228 228 228 228
output
0

input
3 2 2 8
output
3

Note
In the first example, all numbers are already equal. So the needed number of operation is 0.

In the second example, we can apply the operation 3 times: add 8 to first 2, add 8 to second 2, add 2 to 8, making all numbers equal to 10. It can be proved that we can't make all numbers equal in less than 3 operations.

E. Problem from Red Panda

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

At Moscow Workshops ICPC team gets a balloon for each problem they solved first. Team MSU Red Panda got so many balloons that they didn't know how to spend them. So they came up with a problem with them.

There are several balloons, not more than 10^6 in total, each one is colored in one of k colors. We can perform the following operation: choose $k-1$ balloons such that they are of $k-1$ different colors, and recolor them all into remaining color. We can perform this operation any finite number of times (for example, we can only perform the operation if there are at least $k-1$ different colors among current balls).

How many different balloon configurations can we get? Only number of balloons of each color matters, configurations differing only by the order of balloons are counted as equal. As this number can be very large, output it modulo 998244353.

Input
The first line contains a single integer k ($2 \le k \le 10^5$) —the number of colors.

The second line contains k integers $a_1, a_2, \ldots, a_k$ ($0 \le a_i$) —initial configuration of balloons. a_i is number of balloons of color i. The total number of balloons doesn't exceed 10^6. In other words,

$a_1 + a_2 + a_3 + \ldots + a_k \le 10^6$.

Output
Output number of possible configurations modulo 998244353.

Examples

input
3 0 1 2
output
3

input
4 1 1 1 1
output
5

input
5 0 0 1 2 3
output
1

input
3 2 2 8
output
31

Note
In the first example, there are 3 configurations we can get: $[0, 1, 2]$, $[2, 0, 1]$, $[1, 2, 0]$.

In the second example, we can apply the operation not more than once, and possible configurations are: $[1, 1, 1, 1]$, $[0, 0, 0, 4]$, $[0, 0, 4, 0]$, $[0, 4, 0, 0]$, $[4, 0, 0, 0]$.

In the third example, we can't apply any operations, so the only achievable configuration is the starting one.

The only programming contests Web 2.0 platform