A. Keanu Reeves

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

After playing Neo in the legendary "`Matrix`" trilogy, Keanu Reeves started doubting himself: maybe we really live in virtual reality? To find if this is true, he needs to solve the following problem.

Let's call a string consisting of only zeroes and ones **good** if it contains **different** numbers of zeroes and ones. For example, `1`, `101`, `0000` are good, while `01`, `1001`, and `111000` are not good.

We are given a string s of length n consisting of only zeroes and ones. We need to cut s into **minimal possible** number of substrings $s_1, s_2, \ldots, s_k$ such that **all** of them are good. More formally, we have to find **minimal** by number of strings sequence of good strings $s_1, s_2, \ldots, s_k$ such that their concatenation (joining) equals s, i.e. $s_1 + s_2 + \dots + s_k = s$.

For example, cuttings `110010` into `110` and `010` or into `11` and `0010` are valid, as `110`, `010`, `11`, `0010` are all good, and we can't cut `110010` to the smaller number of substrings as `110010` isn't good itself. At the same time, cutting of `110010` into `1100` and `10` isn't valid as both strings aren't good. Also, cutting of `110010` into `1`, `1`, `0010` isn't valid, as it isn't minimal, even though all 3 strings are good.

Can you help Keanu? We can show that the solution always exists. If there are multiple optimal answers, print any.

Input

The first line of the input contains a single integer n ($1 \le n \le 100$) — the length of the string s.

The second line contains the string s of length n consisting only from zeros and ones.

Output

In the first line, output a single integer k ($1 \le k$) — a **minimal** number of strings you have cut s into.

In the second line, output k strings $s_1, s_2, \ldots, s_k$ separated with spaces. The length of each string has to be positive. Their concatenation has to be equal to s and all of them have to be good.

If there are multiple answers, print any.

Examples

input
```
1
1
```
output
```
1
1
```

input
```
2
10
```
output
```
2
1 0
```

input
```
6
100011
```
output
```
2
100 011
```

Note

In the first example, the string `1` wasn't cut at all. As it is good, the condition is satisfied.

In the second example, `1` and `0` both are good. As `10` isn't good, the answer is indeed minimal.

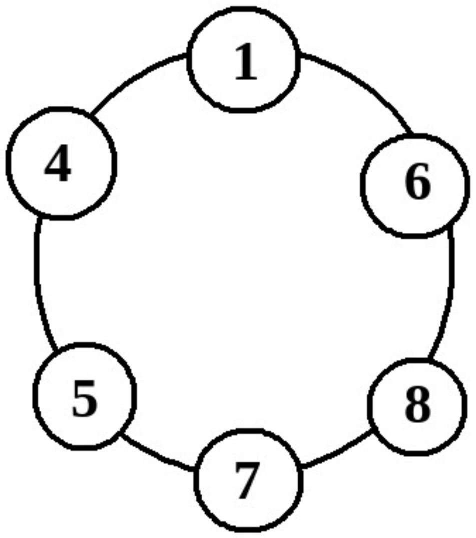
In the third example, `100` and `011` both are good. As `100011` isn't good, the answer is indeed minimal.

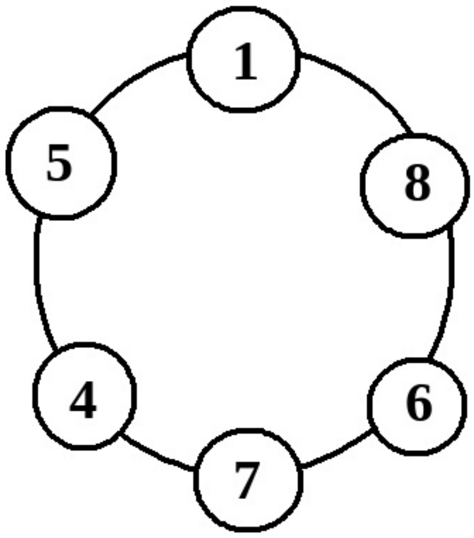
B. Number Circle

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

You are given n numbers $a_1, a_2, \dots, a_n$. Is it possible to arrange them in a circle in such a way that every number is **strictly** less than the sum of its neighbors?

For example, for the array $[1, 4, 5, 6, 7, 8]$, the arrangement on the left is valid, while arrangement on the right is not, as $5 \ge 4 + 1$ and $8 > 1 + 6$.





Input

The first line contains a single integer n ($3 \le n \le 10^5$) — the number of numbers.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^9$) — the numbers. The given numbers are not necessarily distinct (i.e. duplicates are allowed).

Output

If there is no solution, output "`NO`" in the first line.

If there is a solution, output "YES" in the first line. In the second line output n numbers — elements of the array in the order they will stay in the circle. The first and the last element you output are considered neighbors in the circle. If there are multiple solutions, output any of them. You can print the circle starting with any element.

Examples

input

```
3
2 4 3
```

output

```
YES
4 2 3
```

input

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

output

```
YES
4 4 2 1 3
```

input

```
3
13 8 5
```

output

```
NO
```

input

```
4
1 10 100 1000
```

output

```
NO
```

Note

One of the possible arrangements is shown in the first example:

$4 < 2 + 3$;

$2 < 4 + 3$;

$3 < 4 + 2$.

One of the possible arrangements is shown in the second example.

No matter how we arrange $13, 8, 5$ in a circle in the third example, 13 will have 8 and 5 as neighbors, but $13 \ge 8 + 5$.

There is no solution in the fourth example.

C. Candies!

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

Consider a sequence of digits of length 2^k $[a_1, a_2, \ldots, a_{2^k}]$. We perform the following operation with it: replace pairs (a_{2i+1}, a_{2i+2}) with $(a_{2i+1} + a_{2i+2}) \bmod 10$ for $0 \le i < 2^{k-1}$. For every i where $a_{2i+1} + a_{2i+2} \ge 10$ we get a candy! As a result, we will get a sequence of length 2^{k-1}.

Less formally, we partition sequence of length 2^k into 2^{k-1} pairs, each consisting of 2 numbers: the first pair consists of the first and second numbers, the second of the third and fourth $\ldots$, the last pair consists of the ($2^k - 1$)-th and (2^k)-th numbers. For every pair such that sum of numbers in it is at least 10, we get a candy. After that, we replace every pair of numbers with a remainder of the division of their sum by 10 (and don't change the order of the numbers).

Perform this operation with a resulting array until it becomes of length 1. Let $f([a_1, a_2, \ldots, a_{2^k}])$ denote the number of candies we get in this process.

For example: if the starting sequence is $[8, 7, 3, 1, 7, 0, 9, 4]$ then:

After the first operation the sequence becomes $[(8 + 7) \bmod 10, (3 + 1) \bmod 10, (7 + 0) \bmod 10, (9 + 4) \bmod 10] = [5, 4, 7, 3]$, and we get 2 candies as $8 + 7 \ge 10$ and $9 + 4 \ge 10$.

After the second operation the sequence becomes $[(5 + 4) \bmod 10, (7 + 3) \bmod 10] = [9, 0]$, and we get one more candy as $7 + 3 \ge 10$.

After the final operation sequence becomes $[(9 + 0) \bmod 10] = [9]$.

Therefore, $f([8, 7, 3, 1, 7, 0, 9, 4]) = 3$ as we got 3 candies in total.

You are given a sequence of digits of length n $s_1, s_2, \ldots s_n$. You have to answer q queries of the form (l_i, r_i), where for i-th query you have to output $f([s_{l_i}, s_{l_i+1}, \ldots, s_{r_i}])$. It is guaranteed that $r_i - l_i + 1$ is of form 2^k for some nonnegative integer k.

Input

The first line contains a single integer n ($1 \le n \le 10^5$) — the length of the sequence.

The second line contains n digits $s_1, s_2, \ldots, s_n$ ($0 \le s_i \le 9$).

The third line contains a single integer q ($1 \le q \le 10^5$) — the number of queries.

Each of the next q lines contains two integers l_i, r_i ($1 \le l_i \le r_i \le n$) — i-th query. It is guaranteed that $r_i - l_i + 1$ is a nonnegative integer power of 2.

Output

Output q lines, in i-th line output single integer — $f([s_{l_i}, s_{l_i+1}, \ldots, s_{r_i}])$, answer to the i-th query.

Examples

input

```
8
8 7 3 1 7 0 9 4
3
1 8
2 5
7 7
```

output

```
3
1
0
```

input

```
6
0 1 2 3 3 5
3
1 2
1 4
3 6
```

output

```
0
0
1
```

Note

The first example illustrates an example from the statement.

$f([7, 3, 1, 7]) = 1$: sequence of operations is $[7, 3, 1, 7] \to [(7 + 3) \bmod 10, (1 + 7) \bmod 10] = [0, 8]$ and one candy as $7 + 3 \ge 10 \to [(0 + 8) \bmod 10] = [8]$, so we get only 1 candy.

$f([9]) = 0$ as we don't perform operations with it.

D1. Add on a Tree

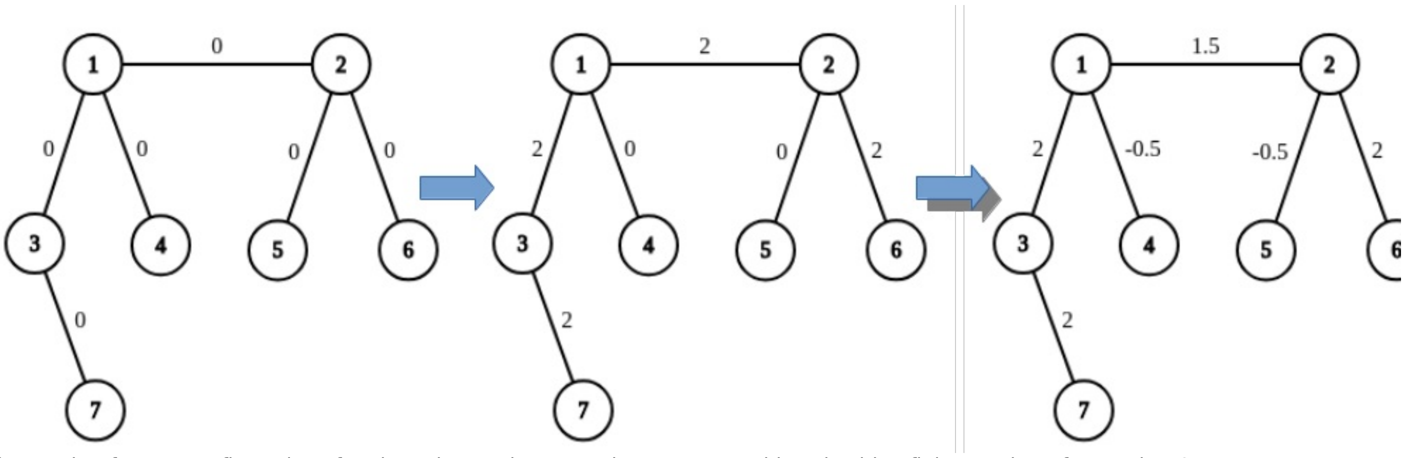
time limit per test: 1 second

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
```

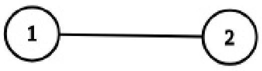
input
```
3
1 2
2 3
```
output
```
NO
```

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

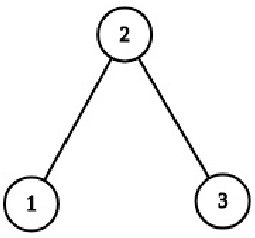
input
```
6
1 2
1 3
1 4
2 5
2 6
```
output
```
YES
```

Note

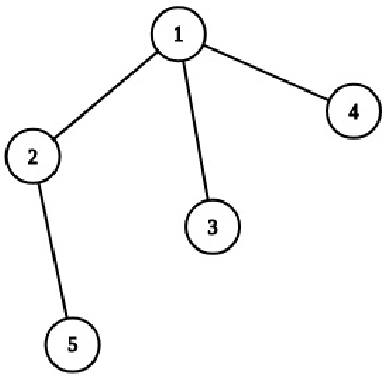
In the first example, we can add any real x to the value written on the only edge $(1, 2)$.

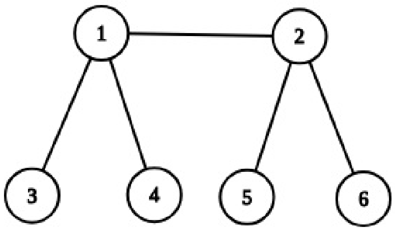


In the second example, one of configurations that we can't reach is 0 written on $(1, 2)$ and 1 written on $(2, 3)$.



Below you can see graphs from examples 3, 4:





D2. Add on a Tree: Revolution

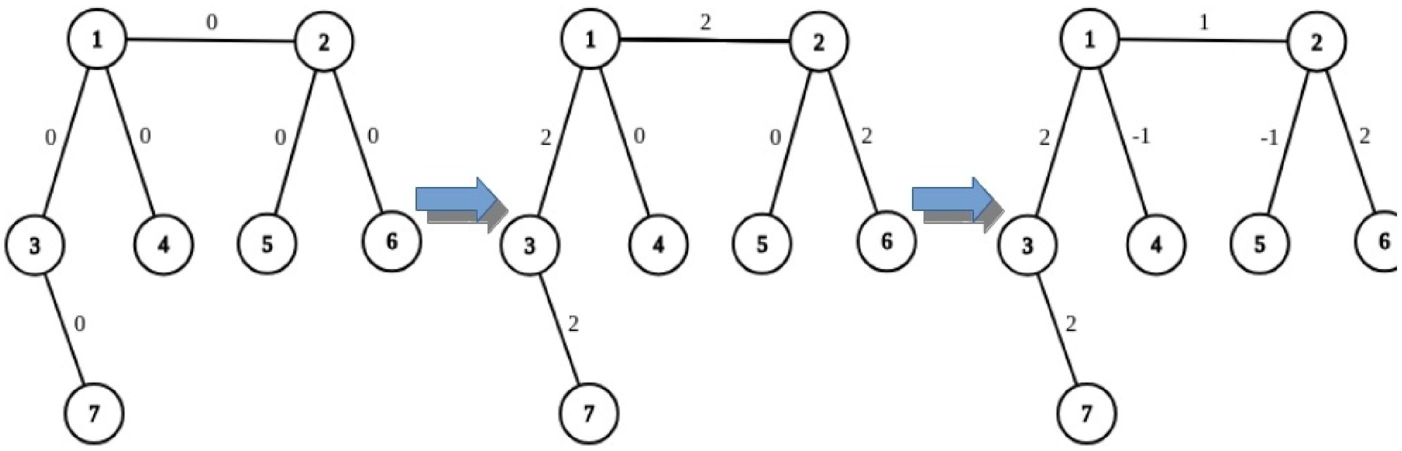
time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input

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

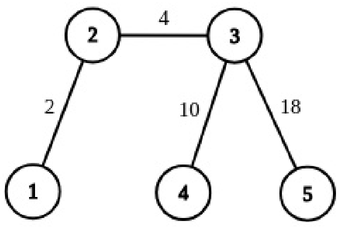
```
6
1 2 6
1 3 8
1 4 12
2 5 2
2 6 4
```

output

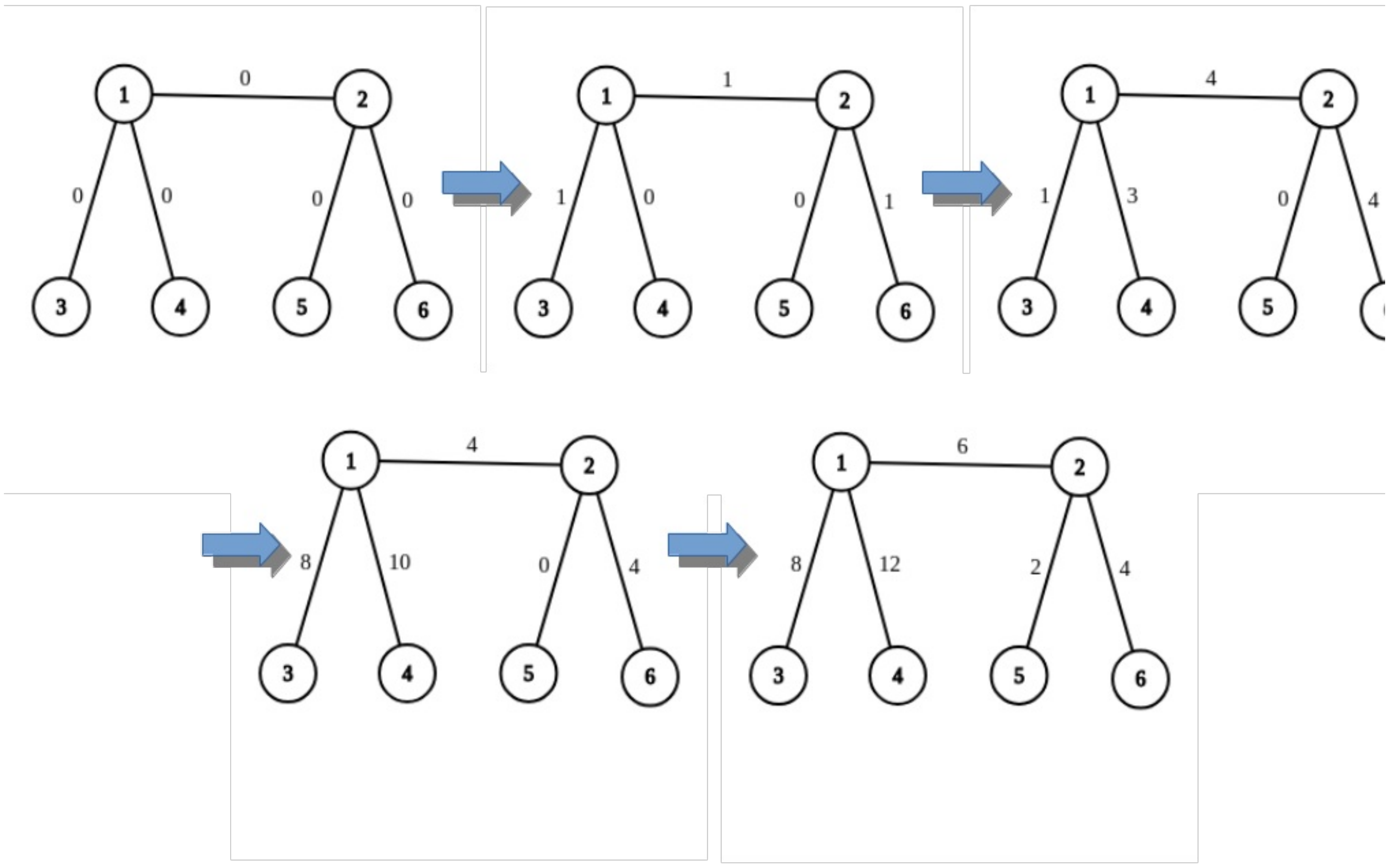
```
YES
4
3 6 1
4 6 3
3 4 7
4 5 2
```

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.



E. 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: $(1,5)$, $(2,3)$, $(4,6)$.

F. 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 $b_1, b_2, \ldots, b_n$ ($n > 1$) — $\min\limits_{1 \leq i < j \leq n} |b_i - b_j|$.

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 ($2 \leq k \leq n \leq 1000$).

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

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 $|10-1| = 9$.

Codeforces (c) Copyright 2010-2022 Mike Mirzayanov
The only programming contests Web 2.0 platform