Codeforces Round #550 (Div. 3)

A. Diverse Strings

1 second, 256 megabytes

A string is called *diverse* if it contains consecutive (adjacent) letters of the Latin alphabet and each letter occurs exactly once. For example, the following strings are diverse: "`fced`", "`xyz`", "`r`" and "`dabcef`". The following string are **not** diverse: "`az`", "`aa`", "`bad`" and "`babc`". Note that the letters '`a`' and '`z`' are not adjacent.

Formally, consider positions of all letters in the string in the alphabet. These positions should form contiguous segment, i.e. they should come one by one without any gaps.

You are given a sequence of strings. For each string, if it is diverse, print "`Yes`". Otherwise, print "`No`".

Input

The first line contains integer n ($1 \le n \le 100$), denoting the number of strings to process. The following n lines contains strings, one string per line. Each string contains only lowercase Latin letters, its length is between 1 and 100, inclusive.

Output

Print n lines, one line per a string in the input. The line should contain "`Yes`" if the corresponding string is diverse and "`No`" if the corresponding string is not diverse. You can print each letter in any case (upper or lower). For example, "`YeS`", "`no`" and "`yES`" are all acceptable.

input

```
8
fced
xyz
r
dabcef
az
aa
bad
babc
```

output

```
Yes
Yes
Yes
Yes
No
No
No
No
```

B. Parity Alternated Deletions

2 seconds, 256 megabytes

Polycarp has an array a consisting of n integers.

He wants to play a game with this array. The game consists of several moves. On the first move he chooses any element and deletes it (after the first move the array contains $n-1$ elements). For each of the next moves he chooses any element with the only restriction: its parity should differ from the parity of the element deleted on the previous move. In other words, he alternates parities (even-odd-even-odd-... or odd-even-odd-even-...) of the removed elements. Polycarp stops if he can't make a move.

Formally:

- If it is the first move, he chooses any element and deletes it;
- If it is the second or any next move:
  - if the last deleted element was **odd**, Polycarp chooses any **even** element and deletes it;
  - if the last deleted element was **even**, Polycarp chooses any **odd** element and deletes it.
- If after some move Polycarp cannot make a move, the game ends.

Polycarp's goal is to **minimize** the sum of **non-deleted** elements of the array after end of the game. If Polycarp can delete the whole array, then the sum of **non-deleted** elements is zero.

Help Polycarp find this value.

Input

The first line of the input contains one integer n ($1 \le n \le 2000$) — the number of elements of a.

The second line of the input contains n integers $a_1, a_2, \dots, a_n$ ($0 \le a_i \le 10^6$), where a_i is the i-th element of a.

Output

Print one integer — the **minimum** possible sum of **non-deleted** elements of the array after end of the game.

input

```
5
1 5 7 8 2
```

output

```
0
```

input

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

output

```
0
```

input

```
2
1000000 1000000
```

output

```
1000000
```

C. Two Shuffled Sequences

2 seconds, 256 megabytes

Two integer sequences existed initially — one of them was **strictly** increasing, and the other one — **strictly** decreasing.

Strictly increasing sequence is a sequence of integers $[x_1 < x_2 < \dots < x_k]$. And strictly decreasing sequence is a sequence of integers $[y_1 > y_2 > \dots > y_l]$. Note that the empty sequence and the sequence consisting of one element can be considered as increasing or decreasing.

They were merged into one sequence a. After that sequence a got shuffled. For example, some of the possible resulting sequences a for an increasing sequence $[1, 3, 4]$ and a decreasing sequence $[10, 4, 2]$ are sequences $[1, 2, 3, 4, 4, 10]$ or $[4, 2, 1, 10, 4, 3]$.

This shuffled sequence a is given in the input.

Your task is to find **any** two suitable initial sequences. One of them should be **strictly** increasing and the other one — **strictly** decreasing. Note that the empty sequence and the sequence consisting of one element can be considered as increasing or decreasing.

If there is a contradiction in the input and it is impossible to split the given sequence a to increasing and decreasing sequences, print "`NO`".

Input

The first line of the input contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of elements in a.

The second line of the input contains n integers $a_1, a_2, \dots, a_n$ ($0 \le a_i \le 2 \cdot 10^5$), where a_i is the i-th element of a.

Output

If there is a contradiction in the input and it is impossible to split the given sequence a to increasing and decreasing sequences, print "NO" in the first line.

Otherwise print "YES" in the first line and **any** two suitable sequences. Note that the empty sequence and the sequence consisting of one element can be considered as increasing or decreasing.

In the second line print n_i — the number of elements in the **strictly increasing** sequence. n_i can be zero, in this case the increasing sequence is empty.

In the third line print n_i integers $inc_1, inc_2, \dots, inc_{n_i}$ in the **increasing** order of its values ($inc_1 < inc_2 < \dots < inc_{n_i}$) — the **strictly increasing** sequence itself. You can keep this line empty if $n_i = 0$ (or just print the empty line).

In the fourth line print n_d — the number of elements in the **strictly decreasing** sequence. n_d can be zero, in this case the decreasing sequence is empty.

In the fifth line print n_d integers $dec_1, dec_2, \dots, dec_{n_d}$ in the **decreasing** order of its values ($dec_1 > dec_2 > \dots > dec_{n_d}$) — the **strictly decreasing** sequence itself. You can keep this line empty if $n_d = 0$ (or just print the empty line).

$n_i + n_d$ should be equal to n and the union of printed sequences should be a permutation of the given sequence (in case of "YES" answer).

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

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

input
```
5
1 1 2 1 2
```
output
```
NO
```

input
```
5
0 1 2 3 4
```
output
```
YES
0

5
4 3 2 1 0
```

D. Equalize Them All

2 seconds, 256 megabytes

You are given an array a consisting of n integers. You can perform the following operations arbitrary number of times (possibly, zero):

1. Choose a pair of indices (i, j) such that $|i - j| = 1$ (indices i and j are adjacent) and set $a_i := a_i + |a_i - a_j|$;
2. Choose a pair of indices (i, j) such that $|i - j| = 1$ (indices i and j are adjacent) and set $a_i := a_i - |a_i - a_j|$.

The value $|x|$ means the absolute value of x. For example, $|4| = 4$, $|-3| = 3$.

Your task is to find the minimum number of operations required to obtain the array of equal elements and print the order of operations to do it.

It is guaranteed that you always can obtain the array of equal elements using such operations.

Note that after each operation each element of the current array should not exceed 10^{18} by absolute value.

Input

The first line of the input contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of elements in a.

The second line of the input contains n integers $a_1, a_2, \dots, a_n$ ($0 \le a_i \le 2 \cdot 10^5$), where a_i is the i-th element of a.

Output

In the first line print one integer k — the minimum number of operations required to obtain the array of equal elements.

In the next k lines print operations itself. The p-th operation should be printed as a triple of integers (t_p, i_p, j_p), where t_p is either 1 or 2 (1 means that you perform the operation of the first type, and 2 means that you perform the operation of the second type), and i_p and j_p are indices of adjacent elements of the array such that $1 \le i_p, j_p \le n$, $|i_p - j_p| = 1$. See the examples for better understanding.

Note that after each operation each element of the current array should not exceed 10^{18} by absolute value.

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

input
```
3
2 8 10
```
output
```
2
2 2 1
2 3 2
```

input
```
4
1 1 1 1
```
output
```
0
```

E. Median String

2 seconds, 256 megabytes

You are given two strings s and t, both consisting of exactly k lowercase Latin letters, s is lexicographically less than t.

Let's consider list of all strings consisting of exactly k lowercase Latin letters, lexicographically not less than s and not greater than t (including s and t) in lexicographical order. For example, for $k = 2$, $s =$"az" and $t =$"bf" the list will be ["az", "ba", "bb", "bc", "bd", "be", "bf"].

Your task is to print the median (the middle element) of this list. For the example above this will be "bc".

It is guaranteed that there is an odd number of strings lexicographically not less than s and not greater than t.

Input

The first line of the input contains one integer k ($1 \le k \le 2 \cdot 10^5$) — the length of strings.

The second line of the input contains one string s consisting of exactly k lowercase Latin letters.

The third line of the input contains one string t consisting of exactly k lowercase Latin letters.

It is guaranteed that s is lexicographically less than t.

It is guaranteed that there is an odd number of strings lexicographically not less than s and not greater than t.

Output

Print one string consisting exactly of k lowercase Latin letters — the median (the middle element) of list of strings of length k lexicographically not less than s and not greater than t.

input
```
2
az
bf
```
output
```
bc
```

input
```
5
afogk
asdji
```
output
```
alvuw
```

input
```
6
nijfvj
tvqhwp
```
output
```
qoztvz
```

F. Graph Without Long Directed Paths

2 seconds, 256 megabytes

You are given a connected undirected graph consisting of n vertices and m edges. There are no self-loops or multiple edges in the given graph.

You have to direct its edges in such a way that the obtained directed graph does not contain any paths of length two or greater (where the length of path is denoted as the number of traversed edges).

Input

The first line contains two integer numbers n and m ($2 \le n \le 2 \cdot 10^5$, $n - 1 \le m \le 2 \cdot 10^5$) — the number of vertices and edges, respectively.

The following m lines contain edges: edge i is given as a pair of vertices u_i, v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$). There are no multiple edges in the given graph, i. e. for each pair (u_i, v_i) there are no other pairs (u_i, v_i) and (v_i, u_i) in the list of edges. It is also guaranteed that the given graph is connected (there is a path between any pair of vertex in the given graph).

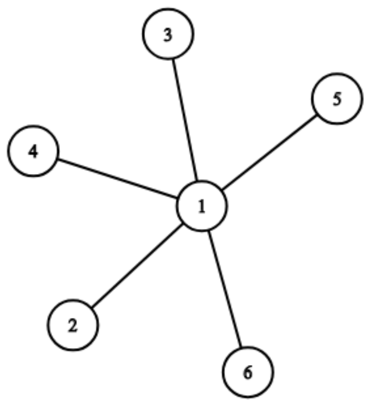
Output

If it is impossible to direct edges of the given graph in such a way that the obtained directed graph does not contain paths of length at least two, print "NO" in the first line.

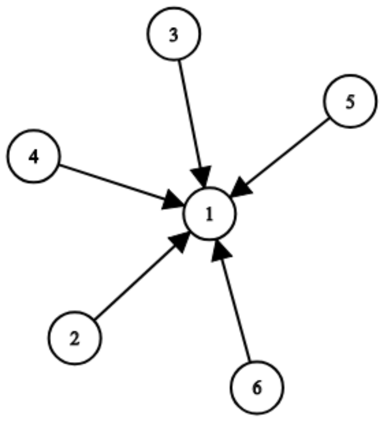
Otherwise print "YES" in the first line, and then print **any** suitable orientation of edges: a binary string (the string consisting only of '0' and '1') of length m. The i-th element of this string should be '0' if the i-th edge of the graph should be directed from u_i to v_i, and '1' otherwise. Edges are numbered in the order they are given in the input.

input
```
6 5
1 5
2 1
1 4
3 1
6 1
```
output
```
YES
10100
```

The picture corresponding to the first example:



And one of possible answers:



G. Two Merged Sequences

2 seconds, 256 megabytes

Two integer sequences existed initially, one of them was **strictly** increasing, and another one — **strictly** decreasing.

Strictly increasing sequence is a sequence of integers $[x_1 < x_2 < \dots < x_k]$. And strictly decreasing sequence is a sequence of integers $[y_1 > y_2 > \dots > y_l]$. Note that the empty sequence and the sequence consisting of one element can be considered as increasing or decreasing.

Elements of increasing sequence were inserted between elements of the decreasing one (and, possibly, before its first element and after its last element) **without changing the order**. For example, sequences $[1, 3, 4]$ and $[10, 4, 2]$ can produce the following resulting sequences: $[10, \mathbf{1}, \mathbf{3}, 4, 2, \mathbf{4}]$, $[\mathbf{1}, \mathbf{3}, \mathbf{4}, 10, 4, 2]$. The following sequence cannot be the result of these insertions: $[\mathbf{1}, 10, \mathbf{4}, 4, \mathbf{3}, 2]$ because the order of elements in the increasing sequence was changed.

Let the obtained sequence be a. This sequence a is given in the input. Your task is to find **any** two suitable initial sequences. One of them should be **strictly** increasing, and another one — **strictly** decreasing. **Note that the empty sequence and the sequence consisting of one element can be considered as increasing or decreasing.**

If there is a contradiction in the input and it is impossible to split the given sequence a into one increasing sequence and one decreasing sequence, print "NO".

Input

The first line of the input contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of elements in a.

The second line of the input contains n integers $a_1, a_2, \dots, a_n$ ($0 \le a_i \le 2 \cdot 10^5$), where a_i is the i-th element of a.

Output

If there is a contradiction in the input and it is impossible to split the given sequence a into one increasing sequence and one decreasing sequence, print "NO" in the first line.

Otherwise print "YES" in the first line. In the second line, print a sequence of n integers $res_1, res_2, \dots, res_n$, where res_i should be either 0 or 1 for each i from 1 to n. The i-th element of this sequence should be 0 if the i-th element of a belongs to the increasing sequence, and 1 otherwise. **Note that the empty sequence and the sequence consisting of one element can be considered as increasing or decreasing.**

input

```
9
5 1 3 6 8 2 9 0 10
```

output

```
YES
1 0 0 0 0 1 0 1 0
```

input

```
5
1 2 4 0 2
```

output

```
NO
```