Codeforces Round #283 (Div. 2)

A. Minimum Difficulty

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

Mike is trying rock climbing but he is awful at it.

There are n holds on the wall, i-th hold is at height a_i off the ground. Besides, let the sequence a_i increase, that is, $a_i < a_{i+1}$ for all i from 1 to $n - 1$; we will call such sequence a *track*. Mike thinks that the track $a_1, \ldots, a_n$ has *difficulty* $d = \max_{1 \le i \le n-1} (a_{i+1} - a_i)$. In other words, difficulty equals the maximum distance between two holds that are adjacent in height.

Today Mike decided to cover the track with holds hanging on heights $a_1, \ldots, a_n$. To make the problem harder, Mike decided to remove one hold, that is, remove one element of the sequence (for example, if we take the sequence $(1, 2, 3, 4, 5)$ and remove the third element from it, we obtain the sequence $(1, 2, 4, 5)$). However, as Mike is awful at climbing, he wants the final difficulty (i.e. the maximum difference of heights between adjacent holds after removing the hold) to be as small as possible among all possible options of removing a hold. The first and last holds **must** stay at their positions.

Help Mike determine the minimum difficulty of the track after removing one hold.

Input

The first line contains a single integer n ($3 \le n \le 100$) — the number of holds.

The next line contains n space-separated integers a_i ($1 \le a_i \le 1000$), where a_i is the height where the hold number i hangs. The sequence a_i is increasing (i.e. each element except for the first one is strictly larger than the previous one).

Output

Print a single number — the minimum difficulty of the track after removing a single hold.

Examples

input
3 1 4 6
output
5

input
5 1 2 3 4 5
output
2

input
5 1 2 3 7 8
output
4

Note

In the first sample you can remove only the second hold, then the sequence looks like $(1, 6)$, the maximum difference of the neighboring elements equals 5.

In the second test after removing every hold the difficulty equals 2.

In the third test you can obtain sequences $(1, 3, 7, 8)$, $(1, 2, 7, 8)$, $(1, 2, 3, 8)$, for which the difficulty is 4, 5 and 5, respectively. Thus, after removing the second element we obtain the optimal answer — 4.

B. Secret Combination

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

You got a box with a combination lock. The lock has a display showing n digits. There are two buttons on the box, each button changes digits on the display. You have quickly discovered that the first button adds 1 to all the digits (all digits 9 become digits 0), and the second button shifts all the digits on the display one position to the right (the last digit becomes the first one). For example, if the display is currently showing number 579, then if we push the first button, the display will show 680, and if after that we push the second button, the display will show 068.

You know that the lock will open if the display is showing the smallest possible number that can be obtained by pushing the buttons in some order. The leading zeros are ignored while comparing numbers. Now your task is to find the desired number.

Input

The first line contains a single integer n ($1 \le n \le 1000$) — the number of digits on the display.

The second line contains n digits — the initial state of the display.

Output

Print a single line containing n digits — the desired state of the display containing the smallest possible number.

Examples

input
3 579
output
024

input
4 2014
output
0142

C. Removing Columns

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

You are given an $n \times m$ rectangular table consisting of lower case English letters. In one operation you can completely remove one column from the table. The remaining parts are combined forming a new table. For example, after removing the second column from the table

abcd
edfg
hijk

we obtain the table:

acd
efg
hjk

A table is called *good* if its rows are ordered from top to bottom lexicographically, i.e. each row is lexicographically no larger than the following one. Determine the minimum number of operations of removing a column needed to make a given table good.

Input

The first line contains two integers — n and m ($1 \le n, m \le 100$).

Next n lines contain m small English letters each — the characters of the table.

Output

Print a single number — the minimum number of columns that you need to remove in order to make the table good.

Examples

input
1 10 codeforces
output
0

input
4 4 case care test code
output
2

input
5 4 code forc esco defo rces
output
4

Note

In the first sample the table is already good.

In the second sample you may remove the first and third column.

In the third sample you have to remove all the columns (note that the table where all rows are empty is considered good by definition).

Let strings s and t have equal length. Then, s is *lexicographically larger* than t if they are not equal and the character following the largest common prefix of s and t (the prefix may be empty) in s is alphabetically larger than the corresponding character of t.

D. Tennis Game

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

Petya and Gena love playing table tennis. A single match is played according to the following rules: a match consists of multiple sets, each set consists of multiple serves. Each serve is won by one of the players, this player scores one point. As soon as one of the players scores t points, he wins the set; then the next set starts and scores of both players are being set to 0. As soon as one of the players wins the total of s sets, he wins the match and the match is over. Here s and t are some positive integer numbers.

To spice it up, Petya and Gena choose new numbers s and t before every match. Besides, for the sake of history they keep a record of each match: that is, for each serve they write down the winner. Serve winners are recorded in the chronological order. In a record the set is over as soon as one of the players scores t points and the match is over as soon as one of the players wins s sets.

Petya and Gena have found a record of an old match. Unfortunately, the sequence of serves in the record isn't divided into sets and numbers s and t for the given match are also lost. The players now wonder what values of s and t might be. Can you determine all the possible options?

Input

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

The second line contains n space-separated integers a_i. If $a_i = 1$, then the i-th serve was won by Petya, if $a_i = 2$, then the i-th serve was won by Gena.

It is not guaranteed that at least one option for numbers s and t corresponds to the given record.

Output

In the first line print a single number k — the number of options for numbers s and t.

In each of the following k lines print two integers s_i and t_i — the option for numbers s and t. Print the options in the order of increasing s_i, and for equal s_i — in the order of increasing t_i.

Examples

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

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

input
4 1 2 1 2
output
0

input
8 2 1 2 1 1 1 1 1
output
3 1 6 2 3 6 1

E. Distributing Parts

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

You are an assistant director in a new musical play. The play consists of n musical parts, each part must be performed by exactly one actor. After the casting the director chose m actors who can take part in the play. Your task is to assign the parts to actors. However, there are several limitations.

First, each actor has a certain voice range and there are some parts that he cannot sing. Formally, there are two integers for each actor, c_i and d_i ($c_i \le d_i$) — the pitch of the lowest and the highest note that the actor can sing. There also are two integers for each part — a_j and b_j ($a_j \le b_j$) — the pitch of the lowest and the highest notes that are present in the part. The i-th actor can perform the j-th part if and only if $c_i \le a_j \le b_j \le d_i$, i.e. each note of the part is in the actor's voice range.

According to the contract, the i-th actor can perform at most k_i parts. Besides, you are allowed not to give any part to some actors (then they take part in crowd scenes).

The rehearsal starts in two hours and you need to do the assignment quickly!

Input

The first line contains a single integer n — the number of parts in the play ($1 \le n \le 10^5$).

Next n lines contain two space-separated integers each, a_j and b_j — the range of notes for the j-th part ($1 \le a_j \le b_j \le 10^9$).

The next line contains a single integer m — the number of actors ($1 \le m \le 10^5$).

Next m lines contain three space-separated integers each, c_i, d_i and k_i — the range of the i-th actor and the number of parts that he can perform ($1 \le c_i \le d_i \le 10^9$, $1 \le k_i \le 10^9$).

Output

If there is an assignment that meets all the criteria aboce, print a single word "YES" (without the quotes) in the first line.

In the next line print n space-separated integers. The i-th integer should be the number of the actor who should perform the i-th part. If there are multiple correct assignments, print any of them.

If there is no correct assignment, print a single word "NO" (without the quotes).

Examples

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

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