Codeforces Round #111 (Div. 2)

A. Twins

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

Imagine that you have a twin brother or sister. Having another person that looks exactly like you seems very unusual. It's hard to say if having something of an alter ego is good or bad. And if you do have a twin, then you very well know what it's like.

Now let's imagine a typical morning in your family. You haven't woken up yet, and Mom is already going to work. She has been so hasty that she has nearly forgotten to leave the two of her darling children some money to buy lunches in the school cafeteria. She fished in the purse and found some number of coins, or to be exact, n coins of arbitrary values $a_1, a_2, ..., a_n$. But as Mom was running out of time, she didn't split the coins for you two. So she scribbled a note asking you to split the money equally.

As you woke up, you found Mom's coins and read her note. "But why split the money equally?" — you thought. After all, your twin is sleeping and he won't know anything. So you decided to act like that: pick for yourself some subset of coins so that the sum of values of your coins is **strictly larger** than the sum of values of the remaining coins that your twin will have. However, you correctly thought that if you take too many coins, the twin will suspect the deception. So, you've decided to stick to the following strategy to avoid suspicions: you take the **minimum number of coins**, whose sum of values is strictly more than the sum of values of the remaining coins. On this basis, determine what **minimum** number of coins you need to take to divide them in the described manner.

Input
The first line contains integer n ($1 \le n \le 100$) — the number of coins. The second line contains a sequence of n integers $a_1, a_2, ..., a_n$ ($1 \le a_i \le 100$) — the coins' values. All numbers are separated with spaces.

Output
In the single line print the single number — the minimum needed number of coins.

Examples

input
2 3 3
output
2

input
3 2 1 2
output
2

Note
In the first sample you will have to take 2 coins (you and your twin have sums equal to $6, 0$ correspondingly). If you take 1 coin, you get sums $3, 3$. If you take 0 coins, you get sums $0, 6$. Those variants do not satisfy you as your sum should be strictly more that your twins' sum.

In the second sample one coin isn't enough for us, too. You can pick coins with values $1, 2$ or $2, 2$. In any case, the minimum number of coins equals 2.

B. Unlucky Ticket

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

Each of you probably has your personal experience of riding public transportation and buying tickets. After a person buys a ticket (which traditionally has an **even** number of digits), he usually checks whether the ticket is lucky. Let us remind you that a ticket is lucky if the sum of digits in its first half matches the sum of digits in its second half.

But of course, not every ticket can be lucky. Far from it! Moreover, sometimes one look at a ticket can be enough to say right away that the ticket is not lucky. So, let's consider the following *unluckiness criterion* that can definitely determine an unlucky ticket. We'll say that a ticket is definitely unlucky if each digit from the first half corresponds to some digit from the second half so that each digit from the first half is **strictly less** than the corresponding digit from the second one or each digit from the first half is **strictly more** than the corresponding digit from the second one. Each digit should be used exactly once in the comparisons. In other words, there is such *bijective correspondence* between the digits of the first and the second half of the ticket, that either each digit of the first half turns out **strictly less** than the corresponding digit of the second half or each digit of the first half turns out **strictly more** than the corresponding digit from the second half.

For example, ticket 2421 meets the following unluckiness criterion and will not be considered lucky (the sought correspondence is $2 > 1$ and $4 > 2$), ticket 0135 also meets the criterion (the sought correspondence is $0 < 3$ and $1 < 5$), and ticket 3754 does not meet the criterion.

You have a ticket in your hands, it contains $2n$ digits. Your task is to check whether it meets the unluckiness criterion.

Input

The first line contains an integer n ($1 \le n \le 100$). The second line contains a string that consists of $2n$ digits and defines your ticket.

Output

In the first line print "YES" if the ticket meets the unluckiness criterion. Otherwise, print "NO" (without the quotes).

Examples

input
2 2421
output
YES

input
2 0135
output
YES

input
2 3754
output
NO

C. Find Pair

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

You've got another problem dealing with arrays. Let's consider an arbitrary sequence containing n (not necessarily different) integers a_1, a_2, ..., a_n. We are interested in all possible pairs of numbers (a_i, a_j), ($1 \le i, j \le n$). In other words, let's consider all n^2 pairs of numbers, picked from the given array.

For example, in sequence $a = \{3, 1, 5\}$ are 9 pairs of numbers:
$(3, 3), (3, 1), (3, 5), (1, 3), (1, 1), (1, 5), (5, 3), (5, 1), (5, 5)$.

Let's sort all resulting pairs lexicographically by non-decreasing. Let us remind you that pair (p_1, q_1) is *lexicographically less* than pair (p_2, q_2) only if either $p_1 < p_2$, or $p_1 = p_2$ and $q_1 < q_2$.

Then the sequence, mentioned above, will be sorted like that: $(1, 1), (1, 3), (1, 5), (3, 1), (3, 3), (3, 5), (5, 1), (5, 3), (5, 5)$

Let's number all the pair in the sorted list from 1 to n^2. Your task is formulated like this: you should find the k-th pair in the ordered list of all possible pairs of the array you've been given.

Input

The first line contains two integers n and k ($1 \le n \le 10^5, 1 \le k \le n^2$). The second line contains the array containing n integers a_1, a_2, ..., a_n ($-10^9 \le a_i \le 10^9$). The numbers in the array can coincide. All numbers are separated with spaces.

Please do not use the `%lld` specificator to read or write 64-bit integers in C++. It is preferred to use `cin`, `cout`, streams or the `%I64d` specificator instead.

Output

In the single line print two numbers — the sought k-th pair.

Examples

input
2 4 2 1
output
2 2

input
3 2 3 1 5
output
1 3

Note

In the first sample the sorted sequence for the given array looks as: $(1, 1), (1, 2), (2, 1), (2, 2)$. The 4-th of them is pair $(2, 2)$.

The sorted sequence for the array from the second sample is given in the statement. The 2-nd pair there is $(1, 3)$.

D. Edges in MST

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

You are given a connected weighted undirected graph without any loops and multiple edges.

Let us remind you that a graph's *spanning tree* is defined as an acyclic connected subgraph of the given graph that includes all of the graph's vertexes. The *weight* of a tree is defined as the sum of weights of the edges that the given tree contains. The *minimum spanning tree* (**MST**) of a graph is defined as the graph's spanning tree having the minimum possible weight. For any connected graph obviously exists the minimum spanning tree, but in the general case, a graph's minimum spanning tree is not unique.

Your task is to determine the following for each edge of the given graph: whether it is either included in **any** MST, or included **at least in one** MST, or **not included in any** MST.

Input

The first line contains two integers n and m ($2 \le n \le 10^5$, $n - 1 \le m \le min(10^5, \frac{n(n-1)}{2})$) — the number of the graph's vertexes and edges, correspondingly. Then follow m lines, each of them contains three integers — the description of the graph's edges as "a_i b_i w_i" ($1 \le a_i, b_i \le n, 1 \le w_i \le 10^6, a_i \ne b_i$), where a_i and b_i are the numbers of vertexes connected by the i-th edge, w_i is the edge's weight. It is guaranteed that the graph is connected and doesn't contain loops or multiple edges.

Output

Print m lines — the answers for all edges. If the i-th edge is included in any MST, print "any"; if the i-th edge is included at least in one MST, print "at least one"; if the i-th edge isn't included in any MST, print "none". Print the answers for the edges in the order in which the edges are specified in the input.

Examples

input
4 5 1 2 101 1 3 100 2 3 2 2 4 2 3 4 1
output
none any at least one at least one any

input
3 3 1 2 1 2 3 1 1 3 2
output
any any none

input
3 3 1 2 1 2 3 1 1 3 1
output
at least one at least one at least one

Note

In the second sample the MST is unique for the given graph: it contains two first edges.

In the third sample any two edges form the MST for the given graph. That means that each edge is included at least in one MST.

E. Buses and People

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

The main Bertown street is represented by a straight line. There are 10^9 bus stops located on the line. The stops are numbered with integers from 1 to 10^9 in the order in which they follow on the road. The city has n buses. Every day the i-th bus drives from stop number s_i to stop number f_i ($s_i < f_i$), it stops on all intermediate stops and returns only at night. The bus starts driving at time t_i and drives so fast that it finishes driving also at time t_i. The time t_i is different for all buses. The buses have infinite capacity.

Bertown has m citizens. Today the i-th person should get from stop number l_i to stop number r_i ($l_i < r_i$); the i-th citizen comes to his initial stop (l_i) at time b_i. Each person, on the one hand, wants to get to the destination point as quickly as possible, and on the other hand, definitely does not want to change the buses as he rides. More formally: the i-th person chooses bus j, with minimum time t_j, such that $s_j \le l_i$, $r_i \le f_j$ and $b_i \le t_j$.

Your task is to determine for each citizen whether he can ride to the destination point today and if he can, find the number of the bus on which the citizen will ride.

Input

The first line contains two integers n and m ($1 \le n, m \le 10^5$) — the number of buses and the number of people.

Then n lines follow, each of them contains three integers: s_i, f_i, t_i ($1 \le s_i, f_i, t_i \le 10^9$, $s_i < f_i$) — the description of the buses. It is guaranteed that all t_i-s are different.

Then m lines follow, each of them contains three integers: l_i, r_i, b_i ($1 \le l_i, r_i, b_i \le 10^9$, $l_i < r_i$) — the Bertown citizens' description. Some b_i-s could coincide.

Output

In the first line print m space-separated integers: the i-th number should be equal either to -1, if the person number i can't get to the destination point, or to the number of the bus that will ride the person number i. The buses are numbered with integers from 1 to n in the input order.

Examples

input
4 3 1 10 10 5 6 2 6 7 3 5 7 4 5 7 1 1 2 1 1 10 11
output
4 1 -1

input
1 1 1 1000000000 1000000000 1 1000000000 1000000000
output
1