Codeforces Round #201 (Div. 1)

A. Alice and Bob

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

It is so boring in the summer holiday, isn't it? So Alice and Bob have invented a new game to play. The rules are as follows. First, they get a set of n distinct integers. And then they take turns to make the following moves. During each move, either Alice or Bob (the player whose turn is the current) can choose two distinct integers x and y from the set, such that the set doesn't contain their absolute difference $|x - y|$. Then this player adds integer $|x - y|$ to the set (so, the size of the set increases by one).

If the current player has no valid move, he (or she) loses the game. The question is who will finally win the game if both players play optimally. Remember that Alice always moves first.

Input

The first line contains an integer n ($2 \le n \le 100$) — the initial number of elements in the set. The second line contains n distinct space-separated integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the elements of the set.

Output

Print a single line with the winner's name. If Alice wins print "Alice", otherwise print "Bob" (without quotes).

Examples

input
2 2 3
output
Alice

input
2 5 3
output
Alice

input
3 5 6 7
output
Bob

Note

Consider the first test sample. Alice moves first, and the only move she can do is to choose 2 and 3, then to add 1 to the set. Next Bob moves, there is no valid move anymore, so the winner is Alice.

B. Lucky Common Subsequence

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

In mathematics, a *subsequence* is a sequence that can be derived from another sequence by deleting some elements without changing the order of the remaining elements. For example, the sequence BDF is a subsequence of ABCDEF. A *substring* of a string is a continuous subsequence of the string. For example, BCD is a substring of ABCDEF.

You are given two strings s_1, s_2 and another string called $virus$. Your task is to find the longest common subsequence of s_1 and s_2, such that it doesn't contain $virus$ as a substring.

Input

The input contains three strings in three separate lines: s_1, s_2 and $virus$ ($1 \leq |s_1|, |s_2|, |virus| \leq 100$). Each string consists only of uppercase English letters.

Output

Output the longest common subsequence of s_1 and s_2 without $virus$ as a substring. If there are multiple answers, any of them will be accepted.

If there is no valid common subsequence, output 0.

Examples

input
AJKEQSLOBSROFGZ OVGURWZLWVLUXTH OZ
output
ORZ

input
AA A A
output
0

C. Number Transformation II

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

You are given a sequence of positive integers $x_1, x_2, ..., x_n$ and two non-negative integers a and b. Your task is to transform a into b. To do that, you can perform the following moves:

- subtract 1 from the current a;
- subtract a mod x_i $(1 \le i \le n)$ from the current a.

Operation a mod x_i means taking the remainder after division of number a by number x_i.

Now you want to know the minimum number of moves needed to transform a into b.

Input

The first line contains a single integer n ($1 \le n \le 10^5$). The second line contains n space-separated integers $x_1, x_2, ..., x_n$ ($2 \le x_i \le 10^9$). The third line contains two integers a and b ($0 \le b \le a \le 10^9$, $a - b \le 10^6$).

Output

Print a single integer — the required minimum number of moves needed to transform number a into number b.

Examples

input
3 3 4 5 30 17
output
6

input
3 5 6 7 1000 200
output
206

D. Robot Control

time limit per test: 6 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The boss of the Company of Robot is a cruel man. His motto is "Move forward Or Die!". And that is exactly what his company's product do. Look at the behavior of the company's robot when it is walking in the directed graph. This behavior has been called "Three Laws of Robotics":

- Law 1. The Robot will destroy itself when it visits a vertex of the graph which it has already visited.
- Law 2. The Robot will destroy itself when it has no way to go (that is when it reaches a vertex whose out-degree is zero).
- Law 3. The Robot will move randomly when it has multiple ways to move (that is when it reach a vertex whose out-degree is more than one). Of course, the robot can move only along the directed edges of the graph.

Can you imagine a robot behaving like that? That's why they are sold at a very low price, just for those who are short of money, including mzry1992, of course. mzry1992 has such a robot, and she wants to move it from vertex s to vertex t in a directed graph safely without self-destruction. Luckily, she can send her robot special orders at each vertex. A special order shows the robot which way to move, if it has multiple ways to move (to prevent random moving of the robot according to Law 3). When the robot reaches vertex t, mzry1992 takes it off the graph immediately. So you can see that, as long as there exists a path from s to t, she can always find a way to reach the goal (whatever the vertex t has the outdegree of zero or not).

Sample 2

However, sending orders is expensive, so your task is to find the minimum number of orders mzry1992 needs to send in the worst case. Please note that mzry1992 can give orders to the robot **while it is walking** on the graph. Look at the first sample to clarify that part of the problem.

Input

The first line contains two integers n ($1 \le n \le 10^6$) — the number of vertices of the graph, and m ($1 \le m \le 10^6$) — the number of edges. Then m lines follow, each with two integers u_i and v_i ($1 \le u_i, v_i \le n$; $v_i \ne u_i$), these integers denote that there is a directed edge from vertex u_i to vertex v_i. The last line contains two integers s and t ($1 \le s, t \le n$).

It is guaranteed that there are no multiple edges and self-loops.

Output

If there is a way to reach a goal, print the required minimum number of orders in the worst case. Otherwise, print -1.

Examples

input
4 6 1 2 2 1 1 3 3 1 2 4 3 4 1 4
output
1

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

Note

Consider the first test sample. Initially the robot is on vertex 1. So, on the first step the robot can go to vertex 2 or 3. No matter what vertex the robot chooses, mzry1992 must give an order to the robot. This order is to go to vertex 4. If mzry1992 doesn't give an order to the robot at vertex 2 or 3, the robot can choose the "bad" outgoing edge (return to vertex 1) according Law 3. So, the answer is one.

E. Doodle Jump

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

In Doodle Jump the aim is to guide a four-legged creature called "The Doodler" up a never-ending series of platforms without falling. — Wikipedia.

It is a very popular game and xiaodao likes it very much. One day when playing the game she wondered whether there exists a platform that the doodler couldn't reach due to the limits of its jumping ability. Consider the following problem.

There are n platforms. The height of the x-th ($1 \le x \le n$) platform is $a \cdot x \bmod p$, where a and p are positive co-prime integers. The maximum possible height of a Doodler's jump is h. That is, it can jump from height h_1 to height h_2 ($h_1 < h_2$) if $h_2 - h_1 \le h$. Initially, the Doodler is on the ground, the height of which is 0. The question is whether it can reach the highest platform or not.

For example, when $a = 7, n = 4, p = 12, h = 2$, the heights of the platforms are $7, 2, 9, 4$ as in the picture below. With the first jump the Doodler can jump to the platform at height 2, with the second one the Doodler can jump to the platform at height 4, but then it can't jump to any of the higher platforms. So, it can't reach the highest platform.

User xiaodao thought about the problem for a long time but didn't solve it, so she asks you for help. Also, she has a lot of instances of the problem. Your task is solve all of these instances.

Input

The first line contains an integer t ($1 \le t \le 10^4$) — the number of problem instances. Each of the next t lines contains four integers a, n, p and h ($1 \le a \le 10^9$, $1 \le n < p \le 10^9$, $0 \le h \le 10^9$). It's guaranteed that a and p are co-prime.

Output

For each problem instance, if the Doodler can reach the highest platform, output "YES", otherwise output "NO".

Examples

input
3 7 4 12 2 7 1 9 4 7 4 12 3
output
NO NO YES