# Educational Codeforces Round 58 (Rated for Div. 2)

## A. Minimum Integer

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

You are given $q$ queries in the following form:

*Given three integers $l_i$, $r_i$ and $d_i$, find minimum **positive** integer $x_i$ such that it is divisible by $d_i$ and it does not belong to the segment $[l_i, r_i]$.*

Can you answer all the queries?

*Recall that a number $x$ belongs to segment $[l, r]$ if $l \le x \le r$.*

### Input
The first line contains one integer $q$ ($1 \le q \le 500$) — the number of queries.

Then $q$ lines follow, each containing a query given in the format $l_i$ $r_i$ $d_i$ ($1 \le l_i \le r_i \le 10^9$, $1 \le d_i \le 10^9$). $l_i$, $r_i$ and $d_i$ are integers.

### Output
For each query print one integer: the answer to this query.

**Example**

| input |
|---|
| 5<br>2 4 2<br>5 10 4<br>3 10 1<br>1 2 3<br>4 6 5 |
| **output** |
| 6<br>4<br>1<br>3<br>10 |

## B. Accordion

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

An *accordion* is a string (yes, in the real world accordions are musical instruments, but let's forget about it for a while) which can be represented as a concatenation of: an opening bracket (ASCII code $091$), a colon (ASCII code $058$), some (possibly zero) vertical line characters (ASCII code $124$), another colon, and a closing bracket (ASCII code $093$). The length of the accordion is the number of characters in it.

For example, `[::]`, `[:||:]` and `[:|||:]` are accordions having length $4$, $6$ and $7$. `(:|:)`, `{:||:}`, `[:]`, `]:||:[` are not accordions.

You are given a string $s$. You want to transform it into an accordion by removing some (possibly zero) characters from it. Note that you may not insert new characters or reorder existing ones. Is it possible to obtain an accordion by removing characters from $s$, and if so, what is the maximum possible length of the result?

### Input
The only line contains one string $s$ ($1 \le |s| \le 500000$). It consists of lowercase Latin letters and characters `[`, `]`, `:` and `|`.

### Output
If it is not possible to obtain an accordion by removing some characters from $s$, print $-1$. Otherwise print maximum possible length of the resulting accordion.

**Examples**

| input |
|---|
| \|[a:b:\|] |

| output |
| --- |
| 4 |

| input |
| --- |
| ]]:[]:] |
| **output** |
| -1 |

## C. Division and Union

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

There are $n$ segments $[l_i, r_i]$ for $1 \le i \le n$. You should divide all segments into two *non-empty* groups in such way that there is no pair of segments from different groups which have at least one common point, or say that it's impossible to do it. Each segment should belong to exactly one group.

To optimize testing process you will be given multitest.

### Input

The first line contains one integer $T$ ($1 \le T \le 50000$) — the number of queries. Each query contains description of the set of segments. Queries are independent.

First line of each query contains single integer $n$ ($2 \le n \le 10^5$) — number of segments. It is guaranteed that $\sum n$ over all queries does not exceed $10^5$.

The next $n$ lines contains two integers $l_i$, $r_i$ per line ($1 \le l_i \le r_i \le 2 \cdot 10^5$) — the $i$-th segment.

### Output

For each query print $n$ integers $t_1, t_2, \ldots, t_n$ ($t_i \in \{1, 2\}$) — for each segment (in the same order as in the input) $t_i$ equals $1$ if the $i$-th segment will belongs to the first group and $2$ otherwise.

If there are multiple answers, you can print any of them. If there is no answer, print $-1$.

#### Example

| input |
| --- |
| 3<br>2<br>5 5<br>2 3<br>3<br>3 5<br>2 3<br>2 3<br>3<br>3 3<br>4 4<br>5 5 |
| **output** |
| 2 1<br>-1<br>1 1 2 |

### Note

In the first query the first and the second segments should be in different groups, but exact numbers don't matter.

In the second query the third segment intersects with the first and the second segments, so they should be in the same group, but then the other group becomes empty, so answer is $-1$.

In the third query we can distribute segments in any way that makes groups non-empty, so any answer of $6$ possible is correct.

## D. GCD Counting

time limit per test: 4.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a tree consisting of $n$ vertices. A number is written on each vertex; the number on vertex $i$ is equal to $a_i$.

Let's denote the function $g(x, y)$ as the greatest common divisor of the numbers written on the vertices belonging to the simple path from vertex $x$ to vertex $y$ (including these two vertices). Also let's denote $dist(x, y)$ as the number of vertices on the simple

path between vertices $x$ and $y$, including the endpoints. $dist(x, x) = 1$ for every vertex $x$.

Your task is calculate the maximum value of $dist(x, y)$ among such pairs of vertices that $g(x, y) > 1$.

### Input
The first line contains one integer $n$ — the number of vertices $(1 \le n \le 2 \cdot 10^5)$.

The second line contains $n$ integers $a_1$, $a_2$, ..., $a_n$ $(1 \le a_i \le 2 \cdot 10^5)$ — the numbers written on vertices.

Then $n - 1$ lines follow, each containing two integers $x$ and $y$ $(1 \le x, y \le n, x \ne y)$ denoting an edge connecting vertex $x$ with vertex $y$. It is guaranteed that these edges form a tree.

### Output
If there is no pair of vertices $x, y$ such that $g(x, y) > 1$, print $0$. Otherwise print the maximum value of $dist(x, y)$ among such pairs.

**Examples**

| input |
| --- |
| 3<br>2 3 4<br>1 2<br>2 3 |
| **output** |
| 1 |

| input |
| --- |
| 3<br>2 3 4<br>1 3<br>2 3 |
| **output** |
| 2 |

| input |
| --- |
| 3<br>1 1 1<br>1 2<br>2 3 |
| **output** |
| 0 |

## E. Polycarp's New Job

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp has recently got himself a new job. He now earns so much that his old wallet can't even store all the money he has.

Berland bills somehow come in lots of different sizes. However, all of them are shaped as rectangles (possibly squares). All wallets are also produced in form of rectangles (possibly squares).

A bill $x \times y$ fits into some wallet $h \times w$ if either $x \le h$ and $y \le w$ or $y \le h$ and $x \le w$. Bills can overlap with each other in a wallet and an infinite amount of bills can fit into a wallet. That implies that all the bills Polycarp currently have fit into a wallet if every single one of them fits into it independently of the others.

Now you are asked to perform the queries of two types:

1. $+ \ x \ y$ — Polycarp earns a bill of size $x \times y$;
2. $? \ h \ w$ — Polycarp wants to check if all the bills he has earned to this moment fit into a wallet of size $h \times w$.

It is guaranteed that there is at least one query of type $1$ before the first query of type $2$ and that there is at least one query of type $2$ in the input data.

For each query of type $2$ print "YES" if all the bills he has earned to this moment fit into a wallet of given size. Print "NO" otherwise.

### Input
The first line contains a single integer $n$ $(2 \le n \le 5 \cdot 10^5)$ — the number of queries.

Each of the next $n$ lines contains a query of one of these two types:

1. $+ \ x \ y$ $(1 \le x, y \le 10^9)$ — Polycarp earns a bill of size $x \times y$;
2. $? \ h \ w$ $(1 \le h, w \le 10^9)$ — Polycarp wants to check if all the bills he has earned to this moment fit into a wallet of size $h \times w$.

It is guaranteed that there is at least one query of type $1$ before the first query of type $2$ and that there is at least one query of type $2$ in the input data.

### Output
For each query of type $2$ print "YES" if all the bills he has earned to this moment fit into a wallet of given size. Print "NO" otherwise.

**Example**

| input |
| --- |
| 9<br>+ 3 2<br>+ 2 3<br>? 1 20<br>? 3 3<br>? 2 3<br>+ 1 5<br>? 10 10<br>? 1 5<br>+ 1 1 |
| **output** |
| NO<br>YES<br>YES<br>YES<br>NO |

### Note
The queries of type $2$ of the example:

1. Neither bill fits;
2. Both bills fit (just checking that you got that bills can overlap);
3. Both bills fit (both bills are actually the same);
4. All bills fit (too much of free space in a wallet is not a problem);
5. Only bill $1 \times 5$ fit (all the others don't, thus it's "NO").

## F. Trucks and Cities

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

There are $n$ cities along the road, which can be represented as a straight line. The $i$-th city is situated at the distance of $a_i$ kilometers from the origin. All cities are situated in the same direction from the origin. There are $m$ trucks travelling from one city to another.

Each truck can be described by $4$ integers: starting city $s_i$, finishing city $f_i$, fuel consumption $c_i$ and number of possible refuelings $r_i$. The $i$-th truck will spend $c_i$ litres of fuel per one kilometer.

When a truck arrives in some city, it can be refueled (but refueling is impossible in the middle of nowhere). The $i$-th truck can be refueled at most $r_i$ times. Each refueling makes truck's gas-tank full. All trucks start with full gas-tank.

All trucks will have gas-tanks of the same size $V$ litres. You should find minimum possible $V$ such that all trucks can reach their destinations without refueling more times than allowed.

### Input
First line contains two integers $n$ and $m$ ($2 \le n \le 400$, $1 \le m \le 250000$) — the number of cities and trucks.

The second line contains $n$ integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$, $a_i < a_{i+1}$) — positions of cities in the ascending order.

Next $m$ lines contains $4$ integers each. The $i$-th line contains integers $s_i$, $f_i$, $c_i$, $r_i$ ($1 \le s_i < f_i \le n$, $1 \le c_i \le 10^9$, $0 \le r_i \le n$) — the description of the $i$-th truck.

### Output
Print the only integer — minimum possible size of gas-tanks $V$ such that all trucks can reach their destinations.

**Example**

| input |
| --- |
| 7 6<br>2 5 7 10 14 15 17<br>1 3 10 0<br>1 7 12 7<br>4 5 13 3<br>4 7 10 1<br>4 7 10 1<br>1 5 11 2 |
| **output** |
| 55 |

### Note
Let's look at queries in details:

1. the $1$-st truck must arrive at position $7$ from $2$ without refuelling, so it needs gas-tank of volume at least $50$.
2. the $2$-nd truck must arrive at position $17$ from $2$ and can be refueled at any city (if it is on the path between starting point and ending point), so it needs gas-tank of volume at least $48$.
3. the $3$-rd truck must arrive at position $14$ from $10$, there is no city between, so it needs gas-tank of volume at least $52$.
4. the $4$-th truck must arrive at position $17$ from $10$ and can be refueled only one time: it's optimal to refuel at $5$-th city (position $14$) so it needs gas-tank of volume at least $40$.
5. the $5$-th truck has the same description, so it also needs gas-tank of volume at least $40$.
6. the $6$-th truck must arrive at position $14$ from $2$ and can be refueled two times: first time in city $2$ or $3$ and second time in city $4$ so it needs gas-tank of volume at least $55$.

## G. (Zero XOR Subset)-less

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

You are given an array $a_1, a_2, \ldots, a_n$ of integer numbers.

Your task is to divide the array into the maximum number of segments in such a way that:

- each element is contained in **exactly one** segment;
- each segment contains at least one element;
- there doesn't exist a non-empty subset of segments such that bitwise XOR of the numbers from them is equal to $0$.

Print the maximum number of segments the array can be divided into. Print $-1$ if no suitable division exists.

### Input
The first line contains a single integer $n$ ($1 \le n \le 2 \cdot 10^5$) — the size of the array.

The second line contains $n$ integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$).

### Output
Print the maximum number of segments the array can be divided into while following the given constraints. Print $-1$ if no suitable division exists.

**Examples**

| input |
| --- |
| 4<br>5 5 7 2 |
| **output** |
| 2 |

| input |
| --- |
| 3<br>1 2 3 |
| **output** |
| -1 |

| input |
| --- |
| 3<br>3 1 10 |
| **output** |
| 3 |

### Note
In the first example $2$ is the maximum number. If you divide the array into $\{[5], [5, 7, 2]\}$, the XOR value of the subset of only the second segment is $5 \oplus 7 \oplus 2 = 0$. $\{[5, 5], [7, 2]\}$ has the value of the subset of only the first segment being $5 \oplus 5 = 0$. However, $\{[5, 5, 7], [2]\}$ will lead to subsets $\{[5, 5, 7]\}$ of XOR $7$, $\{[2]\}$ of XOR $2$ and $\{[5, 5, 7], [2]\}$ of XOR $5 \oplus 5 \oplus 7 \oplus 2 = 5$.

Let's take a look at some division on $3$ segments — $\{[5], [5, 7], [2]\}$. It will produce subsets:

- $\{[5]\}$, XOR $5$;
- $\{[5, 7]\}$, XOR $2$;
- $\{[5], [5, 7]\}$, XOR $7$;
- $\{[2]\}$, XOR $2$;
- $\{[5], [2]\}$, XOR $7$;

- $\{[5, 7], [2]\}$, XOR $0$;
- $\{[5], [5, 7], [2]\}$, XOR $5$;

As you can see, subset $\{[5, 7], [2]\}$ has its XOR equal to $0$, which is unacceptable. You can check that for other divisions of size $3$ or $4$, non-empty subset with $0$ XOR always exists.

The second example has no suitable divisions.

The third example array can be divided into $\{[3], [1], [10]\}$. No subset of these segments has its XOR equal to $0$.

---