A. Min Or Sum

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

You are given an array a of size n.

You can perform the following operation on the array:

- Choose two different integers i, j $(1 \le i < j \le n)$, replace a_i with x and a_j with y. In order not to break the array, $a_i | a_j = x | y$ must be held, where $|$ denotes the bitwise OR operation. Notice that x and y are non-negative integers.

Please output the minimum sum of the array you can get after using the operation above any number of times.

Input
Each test contains multiple test cases. The first line contains the number of test cases t $(1 \le t \le 1000)$. Description of the test cases follows.

The first line of each test case contains an integer n $(2 \le n \le 100)$ — the size of array a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ $(0 \le a_i < 2^{30})$.

Output
For each test case, print one number in a line — the minimum possible sum of the array.

Example

input
4
3
1 3 2
5
1 2 4 8 16
2
6 6
3
3 5 6

output
3
31
6
7

Note
In the first example, you can perform the following operations to obtain the array $[1, 0, 2]$:

1. choose $i = 1, j = 2$, change $a_1 = 1$ and $a_2 = 2$, it's valid since $1|3 = 1|2$. The array becomes $[1, 2, 2]$.

2. choose $i = 2, j = 3$, change $a_2 = 0$ and $a_3 = 2$, it's valid since $2|2 = 0|2$. The array becomes $[1, 0, 2]$.

We can prove that the minimum sum is $1 + 0 + 2 = 3$

In the second example, We don't need any operations.

B. Avoid Local Maximums

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

You are given an array a of size n. Each element in this array is an integer between 1 and 10^9.

You can perform several operations to this array. During an operation, you can replace an element in the array with any integer between 1 and 10^9.

Output the minimum number of operations needed such that the resulting array doesn't contain any local maximums, and the resulting array after the operations.

An element a_i is a local maximum if it is strictly larger than both of its neighbors (that is, $a_i > a_{i-1}$ and $a_i > a_{i+1}$). Since a_1 and a_n have only one neighbor each, they will never be a local maximum.

Input

Each test contains multiple test cases. The first line will contain a single integer t $(1 \le t \le 10000)$ — the number of test cases. Then t test cases follow.

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

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 10^9)$, the elements of array.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, first output a line containing a single integer m — minimum number of operations required. Then ouput a line consist of n integers — the resulting array after the operations. Note that this array should differ in exactly m elements from the initial array.

If there are multiple answers, print any.

Example

input
5
3
2 1 2
4
1 2 3 1
5
1 2 1 2 1
9
1 2 1 3 2 3 1 2 1
9
2 1 3 1 3 1 3 1 3

output
0
2 1 2
1
1 3 3 1
1
1 2 2 2 1
2
1 2 3 3 2 3 3 2 1
2
2 1 3 3 3 1 1 1 3

Note

In the first example, the array contains no local maximum, so we don't need to perform operations.

In the second example, we can change a_2 to 3, then the array don't have local maximums.

C. Differential Sorting

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

You are given an array a of n elements.

Your can perform the following operation no more than n times: Select three indices x, y, z $(1 \le x < y < z \le n)$ and replace a_x with $a_y - a_z$. After the operation, $|a_x|$ need to be less than 10^{18}.

Your goal is to make the resulting array **non-decreasing**. If there are multiple solutions, you can output any. If it is impossible to achieve, you should report it as well.

Input

Each test contains multiple test cases. The first line will contain a single integer t $(1 \le t \le 10000)$ — the number of test cases. Then t test cases follow.

The first line of each test case contains a single integer n $(3 \le n \le 2 \cdot 10^5)$ — the size of the array a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ $(-10^9 \le a_i \le 10^9)$, the elements of a.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print -1 in a single line if there is no solution. Otherwise in the first line you should print a single integer m $(0 \le m \le n)$ — number of operations you performed.

Then the i-th of the following m lines should contain three integers x, y, z $(1 \le x < y < z \le n)$— description of the i-th operation.

If there are multiple solutions, you can output any. Note that you don't have to minimize the number of operations in this task.

Example

input
```
3
5
5 -4 2 -1 2
3
4 3 2
3
-3 -2 -1
```

output
```
2
1 2 3
3 4 5
-1
0
```

Note

In the first example, the array becomes

$[-6, -4, 2, -1, 2]$ after the first operation,

$[-6, -4, -3, -1, 2]$ after the second operation.

In the second example, it is impossible to make the array sorted after any sequence of operations.

In the third example, the array is already sorted, so we don't need to perform any operations.

D. Infinite Set

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

You are given an array a consisting of n **distinct** positive integers.

Let's consider an infinite integer set S which contains all integers x that satisfy at least one of the following conditions:

1. $x = a_i$ for some $1 \le i \le n$.
2. $x = 2y + 1$ and y is in S.
3. $x = 4y$ and y is in S.

For example, if $a = [1, 2]$ then the 10 smallest elements in S will be $\{1, 2, 3, 4, 5, 7, 8, 9, 11, 12\}$.

Find the number of elements in S that are strictly smaller than 2^p. Since this number may be too large, print it modulo $10^9 + 7$.

Input

The first line contains two integers n and p $(1 \le n, p \le 2 \cdot 10^5)$.

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 10^9)$.

It is guaranteed that all the numbers in a are distinct.

Output

Print a single integer, the number of elements in S that are strictly smaller than 2^p. Remember to print it modulo $10^9 + 7$.

Examples

input
```
2 4
6 1
```

output
```
9
```

input
```
4 7
20 39 5 200
```

output
```
14
```

input

output

Note

In the first example, the elements smaller than 2^4 are $\{1, 3, 4, 6, 7, 9, 12, 13, 15\}$.

In the second example, the elements smaller than 2^7 are $\{5, 11, 20, 23, 39, 41, 44, 47, 79, 80, 83, 89, 92, 95\}$.

E. Cars

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

There are n cars on a coordinate axis OX. Each car is located at an integer point initially and no two cars are located at the same point. Also, each car is oriented either left or right, and they can move at any constant positive speed in that direction at any moment.

More formally, we can describe the i-th car with a letter and an integer: its orientation ori_i and its location x_i. If $ori_i = L$, then x_i is decreasing at a constant rate with respect to time. Similarly, if $ori_i = R$, then x_i is increasing at a constant rate with respect to time.

We call two cars **irrelevant** if they never end up in the same point regardless of their speed. In other words, they won't share the same coordinate at any moment.

We call two cars **destined** if they always end up in the same point regardless of their speed. In other words, they must share the same coordinate at some moment.

Unfortunately, we lost all information about our cars, but we do remember m relationships. There are two types of relationships:

$1\ i\ j$ — i-th car and j-th car are **irrelevant**.

$2\ i\ j$ — i-th car and j-th car are **destined**.

Restore the orientations and the locations of the cars satisfying the relationships, or report that it is impossible. If there are multiple solutions, you can output any.

Note that if two cars share the same coordinate, they will intersect, but at the same moment they will continue their movement in their directions.

Input

The first line contains two integers, n and m $(2 \le n \le 2 \cdot 10^5; 1 \le m \le min(2 \cdot 10^5, \frac{n(n-1)}{2}))$ — the number of cars and the number of restrictions respectively.

Each of the next m lines contains three integers, $type$, i, and j $(1 \le type \le 2; 1 \le i, j \le n; i \ne j)$.

If $type = 1$, i-th car and j-th car are **irrelevant**. Otherwise, i-th car and j-th car are **destined**.

It is guaranteed that for each pair of cars, there are at most 1 relationship between.

Output

In the first line, print either "YES" or "NO" (in any case), whether it is possible to restore the orientations and the locations of the cars satisfying the relationships.

If the answer is "YES", print n lines each containing a symbol and an integer: ori_i and x_i $(ori_i \in \{L, R\}; -10^9 \le x_i \le 10^9)$ — representing the information of the i-th car.

If the orientation is left, then $ori_i = L$. Otherwise $ori_i = R$.

x_i is the where the i-th car is located. Note that all x_i should be **distinct**.

We can prove that if there exists a solution, then there must be a solution satisfying the constraints on x_i.

Examples

input

```
4 4
1 1 2
1 2 3
2 3 4
2 4 1
```

output

```
YES
R 0
L -3
```

```
R 5
L 6
```

F. Closest Pair

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

There are n weighted points on the OX-axis. The coordinate and the weight of the i-th point is x_i and w_i, respectively. All points have distinct coordinates and positive weights. Also, $x_i < x_{i+1}$ holds for any $1 \le i < n$.

The weighted distance between i-th point and j-th point is defined as $|x_i - x_j| \cdot (w_i + w_j)$, where $|val|$ denotes the absolute value of val.

You should answer q queries, where the i-th query asks the following: Find the **minimum** weighted distance among all pairs of distinct points among the points in subarray $[l_i, r_i]$.

Input

The first line contains 2 integers n and q $(2 \le n \le 3 \cdot 10^5; 1 \le q \le 3 \cdot 10^5)$ — the number of points and the number of queries.

Then, n lines follows, the i-th of them contains two integers x_i and w_i $(-10^9 \le x_i \le 10^9; 1 \le w_i \le 10^9)$ — the coordinate and the weight of the i-th point.

It is guaranteed that the points are given in the increasing order of x.

Then, q lines follows, the i-th of them contains two integers l_i and r_i $(1 \le l_i < r_i \le n)$ — the given subarray of the i-th query.

Output

For each query output one integer, the **minimum** weighted distance among all pair of distinct points in the given subarray.

Example

input

```
5 5
-2 2
0 10
1 1
9 2
12 7
1 3
2 3
1 5
3 5
2 4
```

output

```
9
11
9
24
11
```

Note

For the first query, the minimum weighted distance is between points 1 and 3, which is equal to $|x_1 - x_3| \cdot (w_1 + w_3) = |-2 - 1| \cdot (2 + 1) = 9$.

For the second query, the minimum weighted distance is between points 2 and 3, which is equal to $|x_2 - x_3| \cdot (w_2 + w_3) = |0 - 1| \cdot (10 + 1) = 11$.

For the fourth query, the minimum weighted distance is between points 3 and 4, which is equal to $|x_3 - x_4| \cdot (w_3 + w_4) = |1 - 9| \cdot (1 + 2) = 24$.