## A. k-Amazing Numbers

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

You are given an array $a$ consisting of $n$ integers numbered from $1$ to $n$.

Let's define the $k$-amazing number of the array as the minimum number that occurs in all of the subsegments of the array having length $k$ (recall that a subsegment of $a$ of length $k$ is a contiguous part of $a$ containing exactly $k$ elements). If there is no integer occuring in all subsegments of length $k$ for some value of $k$, then the $k$-amazing number is $-1$.

For each $k$ from $1$ to $n$ calculate the $k$-amazing number of the array $a$.

### Input

The first line contains one integer $t$ ($1 \le t \le 1000$) — the number of test cases. Then $t$ test cases follow.

The first line of each test case contains one integer $n$ ($1 \le n \le 3 \cdot 10^5$) — the number of elements in the array. The second line contains $n$ integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le n$) — the elements of the array.

It is guaranteed that the sum of $n$ over all test cases does not exceed $3 \cdot 10^5$.

### Output

For each test case print $n$ integers, where the $i$-th integer is equal to the $i$-amazing number of the array.

#### Example

**input**

```
3
5
1 2 3 4 5
5
4 4 4 4 2
6
1 3 1 5 3 1
```

**output**

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

## B. Make Them Equal

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

You are given an array $a$ consisting of $n$ **positive** integers, numbered from $1$ to $n$. You can perform the following operation no more than $3n$ times:

1. choose three integers $i$, $j$ and $x$ ($1 \le i, j \le n$; $0 \le x \le 10^9$);
2. assign $a_i := a_i - x \cdot i$, $a_j := a_j + x \cdot i$.

After each operation, all elements of the array should be **non-negative**.

Can you find a sequence of no more than $3n$ operations after which all elements of the array are equal?

### Input

The first line contains one integer $t$ ($1 \le t \le 10^4$) — the number of test cases. Then $t$ test cases follow.

The first line of each test case contains one integer $n$ ($1 \le n \le 10^4$) — the number of elements in the array. The second line contains $n$ integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^5$) — the elements of the array.

It is guaranteed that the sum of $n$ over all test cases does not exceed $10^4$.

### Output

For each test case print the answer to it as follows:

- if there is no suitable sequence of operations, print $-1$;

- otherwise, print one integer $k$ ($0 \le k \le 3n$) — the number of operations in the sequence. Then print $k$ lines, the $m$-th of which should contain three integers $i$, $j$ and $x$ ($1 \le i, j \le n$; $0 \le x \le 10^9$) for the $m$-th operation.

If there are multiple suitable sequences of operations, print any of them. Note that you don't have to minimize $k$.

**Example**

**input**

```
3
4
2 16 4 18
6
1 2 3 4 5 6
5
11 19 1 1 3
```

**output**

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

# C. XOR Inverse

time limit per test: 2 seconds

memory limit per test: 512 megabytes

input: standard input

output: standard output

You are given an array $a$ consisting of $n$ non-negative integers. You have to choose a non-negative integer $x$ and form a new array $b$ of size $n$ according to the following rule: for all $i$ from $1$ to $n$, $b_i = a_i \oplus x$ ($\oplus$ denotes the operation bitwise XOR).

An inversion in the $b$ array is a pair of integers $i$ and $j$ such that $1 \le i < j \le n$ and $b_i > b_j$.

You should choose $x$ in such a way that the number of inversions in $b$ is minimized. If there are several options for $x$ — output the smallest one.

## Input

First line contains a single integer $n$ ($1 \le n \le 3 \cdot 10^5$) — the number of elements in $a$.

Second line contains $n$ space-separated integers $a_1$, $a_2$, ..., $a_n$ ($0 \le a_i \le 10^9$), where $a_i$ is the $i$-th element of $a$.

## Output

Output two integers: the minimum possible number of inversions in $b$, and the minimum possible value of $x$, which achieves those number of inversions.

**Examples**

**input**

```
4
0 1 3 2
```

**output**

```
1 0
```

**input**

```
9
10 7 9 10 7 5 5 3 5
```

**output**

```
4 14
```

**input**

```
3
8 10 3
```

**output**

```
0 8
```

## Note

In the first sample it is optimal to leave the array as it is by choosing $x = 0$.

In the second sample the selection of $x = 14$ results in $b$: $[4, 9, 7, 4, 9, 11, 11, 13, 11]$. It has $4$ inversions:

- $i = 2$, $j = 3$;

- $i = 2, j = 4$;
- $i = 3, j = 4$;
- $i = 8, j = 9$.

In the third sample the selection of $x = 8$ results in $b$: $[0, 2, 11]$. It has no inversions.

# D. Graph and Queries

time limit per test: 1.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an undirected graph consisting of $n$ vertices and $m$ edges. Initially there is a single integer written on every vertex: the vertex $i$ has $p_i$ written on it. All $p_i$ are distinct integers from $1$ to $n$.

You have to process $q$ queries of two types:

- $1\ v$ — among all vertices reachable from the vertex $v$ using the edges of the graph (including the vertex $v$ itself), find a vertex $u$ with the largest number $p_u$ written on it, print $p_u$ and replace $p_u$ with $0$;
- $2\ i$ — delete the $i$-th edge from the graph.

Note that, in a query of the first type, it is possible that all vertices reachable from $v$ have $0$ written on them. In this case, $u$ is not explicitly defined, but since the selection of $u$ does not affect anything, you can choose any vertex reachable from $v$ and print its value (which is $0$).

### Input

The first line contains three integers $n$, $m$ and $q$ ($1 \le n \le 2 \cdot 10^5$; $1 \le m \le 3 \cdot 10^5$; $1 \le q \le 5 \cdot 10^5$).

The second line contains $n$ distinct integers $p_1, p_2, \ldots, p_n$, where $p_i$ is the number initially written on vertex $i$ ($1 \le p_i \le n$).

Then $m$ lines follow, the $i$-th of them contains two integers $a_i$ and $b_i$ ($1 \le a_i, b_i \le n$, $a_i \ne b_i$) and means that the $i$-th edge connects vertices $a_i$ and $b_i$. It is guaranteed that the graph does not contain multi-edges.

Then $q$ lines follow, which describe the queries. Each line is given by one of the following formats:

- $1\ v$ — denotes a query of the first type with a vertex $v$ ($1 \le v \le n$).
- $2\ i$ — denotes a query of the second type with an edge $i$ ($1 \le i \le m$). For each query of the second type, it is guaranteed that the corresponding edge is not deleted from the graph yet.

### Output

For every query of the first type, print the value of $p_u$ written on the chosen vertex $u$.

#### Example

**input**

```
5 4 6
1 2 5 4 3
1 2
2 3
1 3
4 5
1 1
2 1
2 3
1 1
1 2
1 2
```

**output**

```
5
1
2
0
```

# E. Split

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

One day, *BThero* decided to play around with arrays and came up with the following problem:

You are given an array $a$, which consists of $n$ positive integers. The array is numerated $1$ through $n$. You execute the following procedure **exactly once**:

- You create a new array $b$ which consists of $2n$ **positive** integers, where for each $1 \le i \le n$ the condition $b_{2i-1} + b_{2i} = a_i$ holds. For example, for the array $a = [6, 8, 2]$ you can create $b = [2, 4, 4, 4, 1, 1]$.

- You merge consecutive equal numbers in $b$. For example, $b = [2, 4, 4, 4, 1, 1]$ becomes $b = [2, 4, 1]$.

Find and print *the minimum possible* value of $|b|$ (size of $b$) which can be achieved at the end of the procedure. It can be shown that under the given constraints there is at least one way to construct $b$.

### Input

The first line of the input file contains a single integer $T$ ($1 \le T \le 5 \cdot 10^5$) denoting the number of test cases. The description of $T$ test cases follows.

The first line of each test contains a single integer $n$ ($1 \le n \le 5 \cdot 10^5$).

The second line contains $n$ space-separated integers $a_1$, $a_2$, ..., $a_n$ ($2 \le a_i \le 10^9$).

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

### Output

For each test case, print a single line containing one integer — the minimum possible value of $|b|$.

#### Example

**input**

```
3
3
6 8 2
1
4
3
5 6 6
```

**output**

```
3
1
2
```

## F. Showing Off

time limit per test: 6 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

Another dull quarantine day was going by when *BThero* decided to start researching matrices of size $n \times m$. The rows are numerated $1$ through $n$ from top to bottom, and the columns are numerated $1$ through $m$ from left to right. The cell in the $i$-th row and $j$-th column is denoted as $(i, j)$.

For each cell $(i, j)$ *BThero* had two values:

1. The cost of the cell, which is a single **positive** integer.
2. The direction of the cell, which is one of characters L, R, D, U. Those characters correspond to transitions to adjacent cells $(i, j - 1)$, $(i, j + 1)$, $(i + 1, j)$ or $(i - 1, j)$, respectively. No transition pointed outside of the matrix.

Let us call a cell $(i_2, j_2)$ *reachable* from $(i_1, j_1)$, if, starting from $(i_1, j_1)$ and repeatedly moving to the adjacent cell according to our current direction, we will, sooner or later, visit $(i_2, j_2)$.

*BThero* decided to create another matrix from the existing two. For a cell $(i, j)$, let us denote $S_{i,j}$ as a set of all reachable cells from it (including $(i, j)$ itself). Then, the value at the cell $(i, j)$ in the new matrix will be equal to the sum of costs of all cells in $S_{i,j}$.

After quickly computing the new matrix, *BThero* immediately sent it to his friends. However, he did not save any of the initial matrices! Help him to restore any two valid matrices, which produce the current one.

### Input

The first line of input file contains a single integer $T$ ($1 \le T \le 100$) denoting the number of test cases. The description of $T$ test cases follows.

First line of a test case contains two integers $n$ and $m$ ($1 \le n \cdot m \le 10^5$).

Each of the following $n$ lines contain exactly $m$ integers — the elements of the produced matrix. Each element belongs to the segment $[2, 10^9]$.

It is guaranteed that $\sum (n \cdot m)$ over all test cases does not exceed $10^5$.

### Output

For each test case, if an answer does not exist, print a single word NO. Otherwise, print YES and both matrices in the same format as in the input.

- The first matrix should be the *cost matrix* and the second matrix should be the *direction matrix*.
- All integers in the *cost matrix* should be positive.
- All characters in the *direction matrix* should be valid. No direction should point outside of the matrix.

**Example**

**input**

```
2
3 4
7 6 7 8
5 5 4 4
5 7 4 4
1 1
5
```

**output**

```
YES
1 1 1 1
2 1 1 1
3 2 1 1
R D L L
D R D L
U L R U
NO
```