A. Yet Another Colored Tree Problem

Time limit: 1s
Memory limit: 256MB
Input: stdin
Output: stdout

You are given a tree with n nodes. Each node has a color between 1 and k.

For each color i from 1 to k, print the number of simple paths that contain at least one node with color i.

A simple path is a sequence of edges $(u_1, u_2), (u_2, u_3), \ldots, (u_{k-1}, u_k)$ for which all u_i are pairwise distinct. Two paths are considered different if their corresponding sequences of edges are distinct.

Input

Each test contains multiple testcases. The first line of input contains a single integer t ($1 \leq t \leq 10^4$) — the number of testcases.

The first line of each testcase contains two integers n and k ($1 \leq k \leq n \leq 3 \cdot 10^5$) — the number of nodes and colors, respectively.

The second line contains n integers $c_1, c_2, \ldots, c_n$ ($1 \leq c_i \leq k$) — the colors of the n nodes.

Each of the next $n - 1$ lines contain two integers u_i and v_i ($1 \leq u_i, v_i \leq n, u_i \neq v_i$), meaning that an undirected edge exists between nodes u_i and v_i.

It is guaranteed that the given graph is a tree (an undirected connected graph without cycles).

It is also guaranteed that the sum of n across all testcases does not exceed $3 \cdot 10^5$.

Output

For each testcase, print k integers, the number of simple paths that contain at least one node with color i, for every color i from 1 to k.

Example

stdin

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

stdout

```
8 0 5
1
20 16 19
54 38 15 0 27 15 15
```

Note

Explanation of the first testcase:

There are 8 simple paths that contain at least one node with color 1:
$[1]$
$[2]$

$[1 \to 2]$
$[2 \to 1]$
$[1 \to 3]$
$[3 \to 1]$
$[2 \to 1 \to 3]$
$[3 \to 1 \to 2]$

Since there are no nodes with color 2, there are no paths that contain a node with color 2.

Finally, there are 5 simple paths that contain at least one node with color 3:
$[3]$
$[3 \to 1]$
$[1 \to 3]$
$[3 \to 1 \to 2]$
$[2 \to 1 \to 3]$

In the second testcase, the only path ($[1]$) contains one node with color 1.

B. Coins

Time limit: 1s
Memory limit: 256MB
Input: stdin
Output: stdout

Two players alternatively play the following game:

There are initially n coins. In one move, supposing that the current number of coins is x, the current player can either:

1. Take one coin
2. Take multiple coins, provided that the remaining number of coins is a divisor of x.

The game ends when there are no coins left. The player who took the last coin is considered the winner.

Given the number of coins n, your task is to find which player will win the game, supposing that both of them play optimally.

Input

Each test contains multiple testcases. The first line of input contains one integer t ($1 \leq t \leq 10^3$) — the number of testcases.

Each testcase consists of one line, containing n ($1 \leq n \leq 10^9$) — the number of coins.

Output

For each testcase, print either `First` or `Second`, depending on the player who will win the game.

Example

stdin

```
12
1
2
3
4
5
6
7
8
9
10
27
85653439
```

stdout

```
First
Second
First
First
Second
First
Second
First
Second
First
First
Second
```

C. Almost Tree Cut

Time limit: 1s
Memory limit: 256MB
Input: stdin
Output: stdout

You are given a connected undirected graph with n nodes **and** n **edges**. Each node i has an associated cost a_i.

A cut of the given graph is obtained by removing some edges in order to *split* the graph into **exactly** 2 disjoint connected subgraphs.

If the sum of the costs of the nodes from the first subgraph is s_1 and the sum of the costs of the nodes from the second subgraph is s_2, then the cost of the cut is equal to $|s_1 - s_2|$, where $|x|$ denotes the absolute value of x.

What is the smallest possible cost of a cut of the given graph?

Input

The first line of input contains a single integer n ($3 \leq n \leq 2 \cdot 10^5$) — the number of nodes **and edges** of the given graph.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$) — the associated costs of the n nodes.

The following n lines contain the edges of the graph. Each line contains two integers, u and v ($1 \leq u, v \leq n, u \neq v$) — the endpoints of an edge.

It is guaranteed that there are no multiple edges and that the given graph is connected.

Output

Output a single integer, the minimum possible cost of a cut.

Example 1

```
stdin


3
1 7 3
1 2
2 3
3 1

stdout


3
```

Explanation

If we delete the edges $(1, 2)$ and $(2, 3)$, the resulting subgraphs will be $\{1, 3\}$ and $\{2\}$. The cost of this cut is equal to $|a_1 + a_3 - a_2| = |4 - 7| = 3$.

Example 2

```
stdin


5
1 7 3 3 6
3 5
5 4
1 4
4 2
2 5

stdout


2
```

Example 3

```
stdin
```

```
8
5 7 9 12 13 19 7 16
1 2
2 3
3 1
1 4
4 5
4 6
3 7
7 8
```

stdout

```
0
```

D. Doping 2

Time limit: 1s
Memory limit: 256MB
Input: stdin
Output: stdout

GrandPaPà calls $f(p)$ applied to a permutation $p_1, p_2, \ldots, p_n$ as the number of pairs (i, j) such that $i < j$ and $p_i + 1 = p_j$. For example, $f([1, 2, 3]) = 2$, $f([3, 1, 2]) = 1$ and $f([3, 2, 1]) = 0$.

Given n and a permutation p of length n, we define a permutation p' of length n to be k-special if and only if:

1. p' is lexicographically smaller than p, and
2. $f(p') = k$.

Your task is to count for each $0 \leq k \leq n - 1$ the number of k-special permutations modulo m.

A permutation is an array consisting of n distinct integers from 1 to n in arbitrary order. For example, $[2, 3, 1, 5, 4]$ is a permutation, but $[1, 2, 2]$ is not a permutation (2 appears twice in the array) and $[1, 3, 4]$ is also not a permutation ($n = 3$ but there is 4 in the array).

A permutation a of length n is lexicographically smaller than a permutation b of length n if and only if the following holds: in the first position where a and b differ, the permutation a has a smaller element than the corresponding element in b.

Input

The first line contains two integers n and m ($1 \leq n \leq 100$, $1 \leq m \leq 10^9$) — the length of the permutation and the required modulo.

The second line contains n distinct integers $p_1, p_2, \ldots, p_n$ ($1 \leq p_i \leq n$) — the permutation p.

Output

Print n integers, where the k-th integer is the number of $(k - 1)$-special permutations modulo m.

Example 1

```
stdin

4 666012
1 3 4 2

stdout

0 0 2 1
```

Note

The permutations that are lexicographically smaller than $[1, 3, 4, 2]$ are:

- $[1, 2, 3, 4]$, $f([1, 2, 3, 4]) = 3$;
- $[1, 2, 4, 3]$, $f([1, 2, 4, 3]) = 2$;
- $[1, 3, 2, 4]$, $f([1, 3, 2, 4]) = 2$.

Thus our answer is $[0, 0, 2, 1]$.

Example 2

```
stdin

10 60
10 9 8 7 6 5 4 3 2 1

stdout

0 53 20 32 14 14 32 20 53 1
```

E. Anton Would Approve This Problem

Time limit: 1s
Memory limit: 256MB
Input: stdin
Output: stdout

You are given a binary string of length n. A binary string is considered `good` if none of its substrings are equal to `101` or `010`.

What is the smallest number of characters that need to be deleted from the given string in order to make it `good`?

Input

Each test contains multiple testcases. The first line of input contains a single integer t ($1 \leq t \leq 2 \cdot 10^4$) — the number of testcases.

The first line of each testcase contains a single integer n ($1 \leq n \leq 3 \cdot 10^5$) — the length of the given string.

The second line of each testcase contains a binary string $\overline{s_1 s_2 \ldots s_n}$ ($s_i \in \{'0', '1'\}$).

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

Output

For each testcase, print smallest number of characters that need to be deleted from the given string in order to make it `good`.

Example

```
stdin

7
6
101101
6
001010
1
0
8
00101011
15
001101010101101
2
01
30
010110101010110101011010101101

stdout

2
1
0
2
4
0
9
```

Explanation

In the first testcase, after deleting the second and fifth characters, s will become 1111.

In the second testcase, after deleting the fourth character, s will become 00110.

In the third testcase, we don't need to delete any characters.

In the fourth testcase, after deleting the third and fifth characters, s will become 000011.

F. Difference in Skill

Time limit: 1s
Memory limit: 256MB
Input: stdin
Output: stdout

In a company there are n employees. The skill level of the i-th employee is denoted by an integer a_i.

The company wants to start a new project, which requires some of the n employees. A project is considered `balanced` if the difference in skill between any two employees assigned to the project is at most k.

For every i from 1 to n, print the maximum number of employees that can be part of a `balanced` project which contains employee i.

Input

The first line of input contains two integers n and k ($1 \leq n \leq 2 \cdot 10^5, 0 \leq k \leq 10^9$).

The second line of input contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$) — the skill levels of the n employees.

Output

For every i from 1 to n, print the maximum number of employees that can be part of a `balanced` project which contains employee i.

Example 1

stdin

```
6 2
1 2 3 4 6 9
```

stdout

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

Example 2

stdin

```
15 78
98 190 175 67 109 139 297 175 789 162 109 87 165 243 72
```

stdout

```
8 6 8 7 8 8 2 8 1 8 8 7 8 5 7
```

Note

Explanation of the first testcase:

For the first employee, a balanced project can contain employees a_1, a_2 and a_3.
For the second employee, a balanced project can contain employees a_1, a_2 and a_3.
For the third employee, a balanced project can contain employees a_1, a_2 and a_3.
For the fourth employee, a balanced project can contain employees a_2, a_3 and a_4.
For the fifth employee, a balanced project can contain employees a_4 and a_5.
The sixth employee can be part of a balanced project only by themself.

G. Sign Flipping

Time limit: 1s
Memory limit: 256MB
Input: stdin
Output: stdout

You are given an array $a_1, a_2, ..., a_n$. In one move, you can select an element a_i and flip its sign (that is, perform $a_i := -a_i$).

Let $\text{diff}([x_1, x_2, ..., x_k])$ be the number of distinct elements in an array x of length k.

If we perform the aformentioned moves optimally, what is the maximum possible value of:

$$\sum_{i=1}^{n}\sum_{j=i}^{n}\text{diff}([a_i, a_{i+1}, ..., a_j])$$

Note: You do not have to minimize the number of moves.

Input

The first line of input contains a single integer n ($1 \leq n \leq 3 \cdot 10^5$) — the length of array a.

The second line of input contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \leq a_i \leq 10^9$) — the elements of array a.

Output

The first line of output contains the maximum possible value of $\sum\text{diff}([a_i, a_{i+1}, ..., a_j])$, after performing the sign flipping operations optimally.

Example 1

stdin

```
4
1 1 0 -1
```

stdout

```
19
```

Example 2

stdin

```
3
0 0 0
```

stdout

```
6
```

Example 3

stdin

```
15
2 -2 0 1 -3 0 0 3 1 -1 -4 0 -1 -2 3
```

stdout

```
510
```

Note

In the first testcase, if we flip the signs of a_2 and a_4, the resulting array will be $[1, -1, 0, 1]$. Therefore:

- $\text{diff}([1]) = 1$
- $\text{diff}([1, -1]) = 2$
- $\text{diff}([1, -1, 0]) = 3$
- $\text{diff}([1, -1, 0, 1]) = 3$
- $\text{diff}([-1]) = 1$
- $\text{diff}([-1, 0]) = 2$
- $\text{diff}([-1, 0, 1]) = 3$
- $\text{diff}([0]) = 1$
- $\text{diff}([0, 1]) = 2$
- $\text{diff}([1]) = 1$

The sum of all diff's is equal to $1 + 2 + 3 + 3 + 1 + 2 + 3 + 1 + 2 + 1 = 19$, which is the maximum possible value.

In the second testcase, regardless of how many operations we will perform, a will remain equal to $[0, 0, 0]$.

Therefore, $\mathrm{diff}([a_i, a_{i+1}, ..., a_j])$ is equal to 1, for all $1 \leq i \leq j \leq n$. Since there are 6 subarrays, the answer is equal to $6 \cdot 1 = 6$.

H. The Binary Matrix of All Time

Time limit: 1s
Memory limit: 256MB
Input: stdin
Output: stdout

A binary matrix with n rows and m columns is considered **good** if it has no 1×3 or 3×1 submatrices consisting of equal elements.

For example, $\begin{bmatrix}1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1\end{bmatrix}$ and $\begin{bmatrix}0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1\end{bmatrix}$ are **good**, while $\begin{bmatrix}1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1\end{bmatrix}$ and $\begin{bmatrix}1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0\end{bmatrix}$ are not.

Given two integers n and m, what is the maximum number of 1's in a **good** matrix with n rows and m columns?

Input

Each test contains multiple testcases. The first line of input contains one integer t ($1 \le t \le 10^5$) — the number of testcases.

The first line of each testcase contains two integers n and m ($3 \le n, m \le 10^9$) — the height and width of the matrix.

Output

For each testcase, print one integer, the maximum number of 1's in a **good** matrix with n rows and m columns.

Example

stdin

```
6
3 3
3 4
126 74
169 208
1000000000 987654320
976875237 848936273
```

stdout

```
6
8
6216
23435
658436213333333334
552869881923181134
```

I. Weird Divisibility

Time limit: 1.5s
Memory limit: 256MB
Input: stdin
Output: stdout

You are given an integer a. Find the smallest positive integer b ($b \geq 1$) such that $a + b$ is a divisor of $a \cdot b$.

It can be proven that, under the given constraints, a solution always exists.

Input

Each test contains multiple testcases. The first line of each testcase contains an integer t ($1 \leq t \leq 10^4$) — the number of testcases.

The first line of each testcase contains one integer a ($2 \leq a \leq 10^9$).

Output

For each testcase, print the smallest integer $b \geq 1$ such that $a + b$ is a divisor of $a \cdot b$.

Example

stdin

```
14
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

stdout

```
2
6
4
20
3
42
8
18
10
110
4
156
14
10
```

J. Triple Reverse Sort

Time limit: 1s
Memory limit: 256MB
Input: stdin
Output: stdout

You are given a permutation a of length n.

On this permutation, we can perform the following operation multiple times:

1. Select an index i such that $1 \leq i \leq n-2$;
2. Reverse the subarray $[a_i, a_{i+1}, a_{i+2}]$.

Is it possible to sort the permutation?

A permutation of length n is an array consisting of n distinct integers from 1 to n in arbitrary order. For example, $[2, 3, 1, 5, 4]$ is a permutation, but $[1, 2, 2]$ is not a permutation (2 appears twice in the array), and $[1, 3, 4]$ is also not a permutation ($n = 3$ but there is 4 in the array).

Input

Each test contains multiple testcases. The first line of each testcase contains an integer t ($1 \leq t \leq 10^4$) — the number of testcases.

The first line of each testcase contains one integer n ($1 \leq n \leq 2 \cdot 10^5$) — the length of permutation a.

The second line of each testcase contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq n$, all a_i are pairwise distinct) — the elements of permutation a.

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

Output

For each testcase, print `YES` if a can be sorted, or `NO` otherwise.

Example

stdin

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

stdout

```
YES
NO
YES
NO
NO
YES
```

Note

In the first two testcases, no operations can be performed since $n < 3$. $[1]$ is already sorted, while $[2, 1]$ is not.

In the third testcase, the permutation can be sorted by choosing $i = 1$ and reversing $[a_1, a_2, a_3]$.

In the fourth and fifth testcases, it can be proven that a cannot be sorted.

K. Distinctness Queries

Time limit: 2s
Memory limit: 256MB
Input: stdin
Output: stdout

You are given a matrix with n rows and m columns. You will have to answer q queries of the following type:

- `i1 j1 i2 j2` — in the submatrix whose top-left corner is $(i1, j1)$ and its bottom-right corner is $(i2, j2)$, are all of the elements distinct?

Input

The first line of input contains two integers n and m ($1 \leq n, m, n \cdot m \leq 10^5$) — the height, and width, of the matrix, respectively.

Each of the next n lines of input contain m integers $a_{i,1}, a_{i,2}, \ldots, a_{i,m}$ ($1 \leq a_{i,j} \leq n \cdot m$) — the elements of the i-th row of the given matrix.

The next line contains a single integer q ($1 \leq q \leq 10^5$) — the number of queries.

Each of the following q lines contain 4 integers i_1, j_1, i_2, j_2 ($1 \leq i_1 \leq i_2 \leq n, 1 \leq j_1 \leq j_2 \leq m$) — the submatrix's corner cells' coordinates.

Output

For each query, print `YES`, if all of the elements of the given submatrix are distinct, or `NO`, otherwise.

Example 1

stdin

```
2 2
2 1
3 2
9
1 1 1 1
1 2 1 2
2 1 2 1
2 2 2 2
1 1 1 2
2 1 2 2
1 1 2 1
1 2 2 2
1 1 2 2
```

stdout

```
YES
YES
YES
YES
YES
YES
YES
YES
NO
```

Example 2

stdin

```
4 6
1 4 7 1 2 3
2 5 8 6 5 4
3 6 9 7 8 9
1 2 3 4 5 6
10
1 1 3 3
1 4 3 6
1 1 4 6
1 2 4 3
2 1 3 4
4 1 4 6
1 4 4 4
2 5 4 5
2 1 2 6
4 6 4 6
```

stdout

```
YES
YES
NO
YES
NO
YES
YES
NO
NO
YES
```

L. Best or Worst

Time limit: 1s
Memory limit: 256MB
Input: stdin
Output: stdout

George dreamt about an *unusual* permutation $p_1, p_2, \ldots, p_n$.

A permutation is considered *unusual* if, for each element p_i, **exactly one** of the following conditions is true:

1. $p_i = min(p_1, p_2, \ldots, p_i)$
2. $p_i = max(p_1, p_2, \ldots, p_i)$

For example, $[1]$, $[2, 1, 3, 4]$ and $[1, 2, 3, 4, 5]$ are *unusual*, while $[3, 1, 2]$ and $[3, 2, 5, 4, 1]$ are not.

Unfortunately, after waking up, George remembers only some of the elements from p. As such, he now wonders how many *unusual* permutations containing the remaining numbers exist.

Since the answer can be very large, George only wants to know its remainder modulo $10^9 + 7$.

Input

Each test contains multiple test cases. The first line of input contains a single integer t ($1 \leq t \leq 2 \cdot 10^4$) — the number of testcases.

The first line of each test case contains a single integer n ($1 \leq n \leq 2 \cdot 10^5$) — the number of elements in the permutation.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \leq a_i \leq n$). If $a_i = 0$, then George forgot the value of p_i. Otherwise, George knows that p_i must be equal to a_i. It is guaranteed that there are no two indices $1 \leq i < j \leq n$ for which $0 \neq a_i = a_j$.

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

Output

For each test case, output a single integer, denoting the number of *unusual* permutations containing the remaining numbers. Since the answer can be very large, print its remainder modulo $10^9 + 7$.

Example

stdin

```
4
4
2 0 0 4
3
3 1 2
5
0 0 0 4 0
1
0
```

stdout

```
2
0
4
1
```

Note

In the first testcase, there are two possible permutations: $[2, 1, 3, 4]$ and $[2, 3, 1, 4]$, both of which are *unusual*.

In the second testcase, the only possible permutation is $[3, 1, 2]$, which is not *unusual*.

In the third testcase, the 4 *unusual* permutations are $[1, 2, 3, 4, 5]$, $[2, 1, 3, 4, 5]$, $[2, 3, 1, 4, 5]$ and $[3, 2, 1, 4, 5]$.

In the fourth testcase, the only possible permutation is $[1]$, which is *unusual*.