Codeforces Global Round 21

A. NIT orz!

time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

NIT, the cleaver, is new in town! Thousands of people line up to orz him. To keep his orzers entertained, NIT decided to let them solve the following problem related to $\operatorname{or} z$. Can you solve this problem too?

You are given a 1-indexed array of n integers, a, and an integer z. You can do the following operation any number (possibly zero) of times:

- Select a positive integer i such that $1 \le i \le n$. Then, **simutaneously** set a_i to $(a_i \operatorname{or} z)$ and set z to $(a_i \operatorname{and} z)$. In other words, let x and y respectively be the current values of a_i and z. Then set a_i to $(x \operatorname{or} y)$ and set z to $(x \operatorname{and} y)$.

Here or and and denote the bitwise operations OR and AND respectively.

Find the maximum possible value of the maximum value in a after any number (possibly zero) of operations.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 100$). Description of the test cases follows.

The first line of each test case contains two integers n and z ($1 \le n \le 2000$, $0 \le z < 2^{30}$).

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

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

Output

For each test case, print one integer — the answer to the problem.

Example

input

```
5
2 3
3 4
5 5
0 2 4 6 8
1 9
10
5 7
7 15 30 29 27
3 39548743
10293834 10284344 13635445
```

output

```
7
13
11
31
48234367
```

Note

In the first test case of the sample, one optimal sequence of operations is:

- Do the operation with $i = 1$. Now a_1 becomes $(3 \operatorname{or} 3) = 3$ and z becomes $(3 \operatorname{and} 3) = 3$.
- Do the operation with $i = 2$. Now a_2 becomes $(4 \operatorname{or} 3) = 7$ and z becomes $(4 \operatorname{and} 3) = 0$.
- Do the operation with $i = 1$. Now a_1 becomes $(3 \operatorname{or} 0) = 3$ and z becomes $(3 \operatorname{and} 0) = 0$.

After these operations, the sequence a becomes $[3, 7]$, and the maximum value in it is 7. We can prove that the maximum value in a can never exceed 7, so the answer is 7.

In the fourth test case of the sample, one optimal sequence of operations is:

- Do the operation with $i = 1$. Now a_1 becomes $(7 \operatorname{or} 7) = 7$ and z becomes $(7 \operatorname{and} 7) = 7$.
- Do the operation with $i = 3$. Now a_3 becomes $(30 \operatorname{or} 7) = 31$ and z becomes $(30 \operatorname{and} 7) = 6$.
- Do the operation with $i = 5$. Now a_5 becomes $(27 \operatorname{or} 6) = 31$ and z becomes $(27 \operatorname{and} 6) = 2$.

B. NIT Destroys the Universe

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

For a collection of integers S, define $\mathrm{mex}(S)$ as the smallest non-negative integer that does not appear in S.

NIT, the cleaver, decides to destroy the universe. He is not so powerful as Thanos, so he can only destroy the universe by snapping his fingers several times.

The universe can be represented as a 1-indexed array a of length n. When NIT snaps his fingers, he does the following operation on the array:

- He selects positive integers l and r such that $1 \le l \le r \le n$. Let $w = \mathrm{mex}(\{a_l, a_{l+1}, \dots, a_r\})$. Then, for all $l \le i \le r$, set a_i to w.

We say the universe is destroyed if and only if for all $1 \le i \le n$, $a_i = 0$ holds.

Find the minimum number of times NIT needs to snap his fingers to destroy the universe. That is, find the minimum number of operations NIT needs to perform to make all elements in the array equal to 0.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). Description of the test cases follows.

The first line of each test case contains one integer n ($1 \le n \le 10^5$).

The second line of each test case contains n integers a_1, a_2, $\dots$, a_n ($0 \le a_i \le 10^9$).

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 one integer — the answer to the problem.

Example

input

```
4
4
0 0 0 0
5
0 1 2 3 4
7
0 2 3 0 1 2 0
1
1000000000
```

output

```
0
1
2
1
```

Note

In the first test case, we do 0 operations and all elements in the array are already equal to 0.

In the second test case, one optimal way is doing the operation with $l = 2$, $r = 5$.

In the third test case, one optimal way is doing the operation twice, respectively with $l = 4$, $r = 4$ and $l = 2$, $r = 6$.

In the fourth test case, one optimal way is doing the operation with $l = 1$, $r = 1$.

C. Fishingprince Plays With Array

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

Fishingprince is playing with an array $[a_1, a_2, \dots, a_n]$. He also has a magic number m.

He can do the following two operations on it:

- Select $1 \le i \le n$ such that a_i is divisible by m (that is, there exists an integer t such that $m \cdot t = a_i$). Replace a_i with m **copies** of $\frac{a_i}{m}$. The order of the other elements doesn't change. For example, when $m = 2$ and $a = [2, 3]$ and $i = 1$, a changes into $[1, 1, 3]$.
- Select $1 \le i \le n - m + 1$ such that $a_i = a_{i+1} = \dots = a_{i+m-1}$. Replace these m elements with **a single** $m \cdot a_i$. The order of the other elements doesn't change. For example, when $m = 2$ and $a = [3, 2, 2, 3]$ and $i = 2$, a changes into $[3, 4, 3]$.

Note that the array length might change during the process. The value of n above is defined as the current length of the array (might differ from the n in the input).

Fishingprince has another array $[b_1, b_2, \ldots, b_k]$. Please determine if he can turn a into b using any number (possibly zero) of operations.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). Description of the test cases follows.

The first line of each test case contains two integers n and m ($1 \le n \le 5 \cdot 10^4$, $2 \le m \le 10^9$).

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 third line of each test case contains one integer k ($1 \le k \le 5 \cdot 10^4$).

The fourth line of each test case contains k integers $b_1, b_2, \ldots, b_k$ ($1 \le b_i \le 10^9$).

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

Output

For each testcase, print Yes if it is possible to turn a into b, and No otherwise. You can print each letter in any case (upper or lower).

Example

input

```
5
5 2
1 2 2 4 2
4
1 4 4 2
6 2
1 2 2 8 2 2
2
1 16
8 3
3 3 3 3 3 3 3 3
4
6 6 6 6
8 3
3 9 6 3 12 12 36 12
16
9 3 2 2 2 3 4 12 4 12 4 12 4 12 4 4
8 3
3 9 6 3 12 12 36 12
7
12 2 4 3 4 12 56
```

output

```
Yes
Yes
No
Yes
No
```

Note

In the first test case of the sample, we can do the second operation with $i = 2$: $[1, 2, 2, 4, 2] \to [1, 4, 4, 2]$.

In the second testcase of the sample, we can:

- do the second operation with $i = 2$: $[1, 2, 2, 8, 2, 2] \to [1, 4, 8, 2, 2]$.
- do the second operation with $i = 4$: $[1, 4, 8, 2, 2] \to [1, 4, 8, 4]$.
- do the first operation with $i = 3$: $[1, 4, 8, 4] \to [1, 4, 4, 4, 4]$.
- do the second operation with $i = 2$: $[1, 4, 4, 4, 4] \to [1, 8, 4, 4]$.
- do the second operation with $i = 3$: $[1, 8, 4, 4] \to [1, 8, 8]$.
- do the second operation with $i = 2$: $[1, 8, 8] \to [1, 16]$.

D. Permutation Graph

time limit per test: 2 seconds

memory limit per test: 512 megabytes

input: standard input

output: standard output

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).

You are given a permutation of $1, 2, \ldots, n$, $[a_1, a_2, \ldots, a_n]$. For integers i, j such that $1 \le i < j \le n$, define $\mathrm{mn}(i, j)$ as $\min\limits_{k=i}^{j} a_k$, and define $\mathrm{mx}(i, j)$ as $\max_{k=i}^{j} a_k$.

Let us build an undirected graph of n vertices, numbered 1 to n. For every pair of integers $1 \le i < j \le n$, if $\mathrm{mn}(i, j) = a_i$ and $\mathrm{mx}(i, j) = a_j$ both holds, or $\mathrm{mn}(i, j) = a_j$ and $\mathrm{mx}(i, j) = a_i$ both holds, add an undirected edge of length 1 between vertices i and j.

In this graph, find the length of the shortest path from vertex 1 to vertex n. We can prove that 1 and n will always be connected via some path, so a shortest path always exists.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 5 \cdot 10^4$). Description of the test cases follows.

The first line of each test case contains one integer n ($1 \le n \le 2.5 \cdot 10^5$).

The second line of each test case contains n integers a_1, a_2, $\ldots$, a_n ($1 \le a_i \le n$). It's guaranteed that a is a permutation of 1, 2, $\ldots$, n.

It is guaranteed that the sum of 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 length of the shortest path from 1 to n.

Example

input

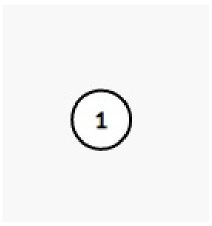
```
5
1
1
2
1 2
5
1 4 2 3 5
5
2 1 5 3 4
10
7 4 8 1 6 10 3 5 2 9
```

output

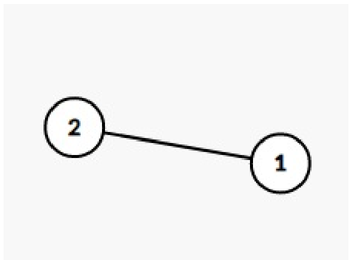
```
0
1
1
4
6
```

Note

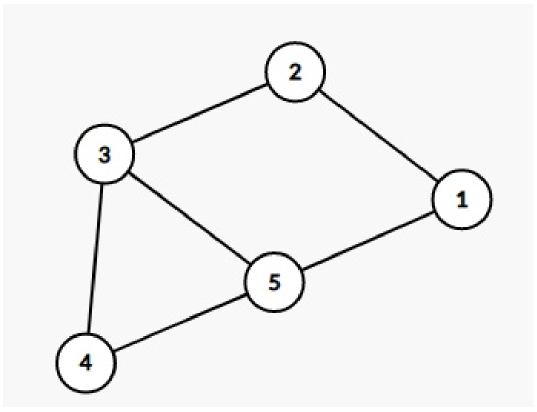
The following are illustrations of constructed graphs in example test cases.



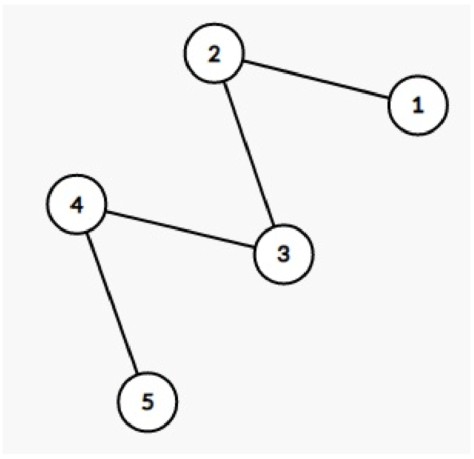
the constructed graph in test case 1



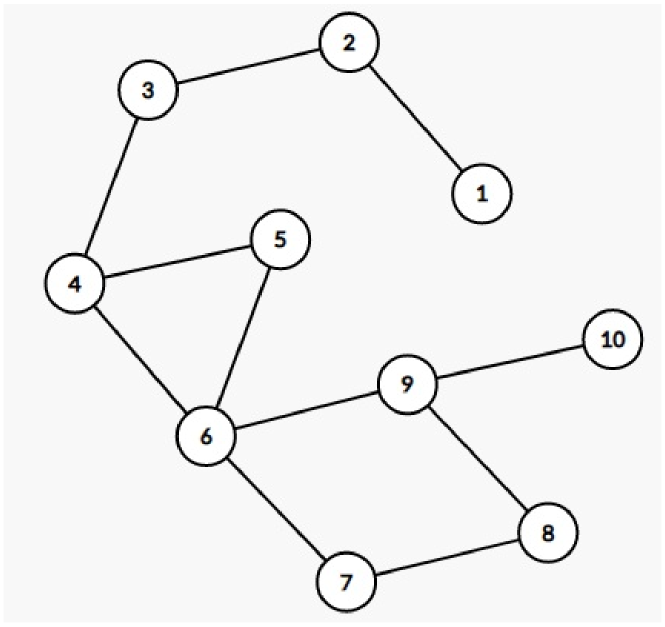
the constructed graph in test case 2



the constructed graph in test case 3



the constructed graph in test case 4



the constructed graph in test case 5

E. Placing Jinas

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

We say an infinite sequence $a_0, a_1, a_2, \ldots$ is **non-increasing** if and only if for all $i \geq 0$, $a_i \geq a_{i+1}$.

There is an infinite right and down grid. The upper-left cell has coordinates $(0,0)$. Rows are numbered 0 to infinity from top to bottom, columns are numbered from 0 to infinity from left to right.

There is also a **non-increasing** infinite sequence $a_0, a_1, a_2, \ldots$. You are given a_0, a_1, $\ldots$, a_n; for all $i > n$, $a_i = 0$. For every pair of x, y, the cell with coordinates (x, y) (which is located at the intersection of x-th row and y-th column) is white if $y < a_x$ and black otherwise.

Initially there is one doll named Jina on $(0,0)$. You can do the following operation.

- Select one doll on (x, y). Remove it and place a doll on $(x, y+1)$ and place a doll on $(x+1, y)$.

Note that multiple dolls can be present at a cell at the same time; in one operation, you remove only one. Your goal is to make all white cells contain 0 dolls.

What's the minimum number of operations needed to achieve the goal? Print the answer modulo $10^9 + 7$.

Input

The first line of input contains one integer n ($1 \leq n \leq 2 \cdot 10^5$).

The second line of input contains $n + 1$ integers $a_0, a_1, \ldots, a_n$ ($0 \leq a_i \leq 2 \cdot 10^5$).

It is guaranteed that the sequence a is **non-increasing**.

Output

Print one integer — the answer to the problem, modulo $10^9 + 7$.

Examples

input

```
2
2 2 0
```

output

```
5
```

input

```
10
12 11 8 8 6 6 6 5 3 2 1
```

output

```
2596
```

Note

Consider the first example. In the given grid, cells $(0, 0), (0, 1), (1, 0), (1, 1)$ are white, and all other cells are black. Let us use triples to describe the grid: triple (x, y, z) means that there are z dolls placed on cell (x, y). Initially the state of the grid is $(0, 0, 1)$.

One of the optimal sequence of operations is as follows:

- Do the operation with $(0, 0)$. Now the state of the grid is $(1, 0, 1), (0, 1, 1)$.
- Do the operation with $(0, 1)$. Now the state of the grid is $(1, 0, 1), (1, 1, 1), (0, 2, 1)$.
- Do the operation with $(1, 0)$. Now the state of the grid is $(1, 1, 2), (0, 2, 1), (2, 0, 1)$.
- Do the operation with $(1, 1)$. Now the state of the grid is $(1, 1, 1), (0, 2, 1), (2, 0, 1), (1, 2, 1), (2, 1, 1)$.
- Do the operation with $(1, 1)$. Now the state of the grid is $(0, 2, 1), (2, 0, 1), (1, 2, 2), (2, 1, 2)$.

Now all white cells contain 0 dolls, so we have achieved the goal with 5 operations.

F. Tree Recovery

time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

Fishingprince loves trees. A tree is a connected undirected graph without cycles.

Fishingprince has a tree of n vertices. The vertices are numbered 1 through n. Let $d(x, y)$ denote the shortest distance on the tree from vertex x to vertex y, assuming that the length of each edge is 1.

However, the tree was lost in an accident. Fortunately, Fishingprince still remembers some information about the tree. More specifically, for every triple of integers x, y, z ($1 \leq x < y \leq n$, $1 \leq z \leq n$) he remembers whether $d(x, z) = d(y, z)$ or not.

Help him recover the structure of the tree, or report that no tree satisfying the constraints exists.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 200$). Description of the test cases follows.

The first line of each test case contains an integer n ($2 \leq n \leq 100$) — the number of vertices in the tree.

Then $n - 1$ lines follow. The i-th line of these $n - 1$ lines contains $n - i$ strings of length n consisting of `0` and `1`. If the k-th character in the j-th string of the i-th line is `0`, it means that $d(i, k) \neq d(i + j, k)$; if the k-th character in the j-th string of the i-th line is `1`, it means that $d(i, k) = d(i + j, k)$.

It is guaranteed that in one input file,

- there are at most 2 test cases that have $n > 50$;
- there are at most 5 test cases that have $n > 20$.

Output

For each test case:

- if no answer exists, output No;
- otherwise, on the first line output Yes. Then output $n-1$ lines. Each line should contain two integers x, y ($1 \le x, y \le n$), denoting an edge between vertices x and y of the tree. If there are multiple solutions, print any.

When printing Yes and No, you can print each letter in any case (upper or lower).

Example

input

```
5
2
00
2
10
3
001 000
000
3
001 010
000
5
00000 01001 00000 01100
00000 10000 00000
00000 11010
00000
```

output

```
Yes
1 2
No
Yes
1 3
2 3
No
Yes
1 2
1 4
2 3
2 5
```

G. Fishingprince Plays With Array Again

time limit per test: 6 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

Suppose you are given a 1-indexed sequence a of non-negative integers, whose length is n, and two integers x, y. In consecutive t seconds (t can be any positive real number), you can do one of the following operations:

- Select $1 \le i < n$, decrease a_i by $x \cdot t$, and decrease a_{i+1} by $y \cdot t$.
- Select $1 \le i < n$, decrease a_i by $y \cdot t$, and decrease a_{i+1} by $x \cdot t$.

Define the minimum amount of time (it might be a real number) required to make all elements in the sequence less than or equal to 0 as $f(a)$.

For example, when $x=1$, $y=2$, it takes 3 seconds to deal with the array $[3,1,1,3]$. We can:

- In the first 1.5 seconds do the second operation with $i=1$.
- In the next 1.5 seconds do the first operation with $i=3$.

We can prove that it's not possible to make all elements less than or equal to 0 in less than 3 seconds, so $f([3,1,1,3])=3$.

Now you are given a 1-indexed sequence b of positive integers, whose length is n. You are also given positive integers x, y. Process q queries of the following two types:

- 1 k v: change b_k to v.
- 2 l r: print $f([b_l, b_{l+1}, \dots, b_r])$.

Input

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

The second line of input contains two integers x and y ($1 \le x, y \le 10^6$).

The third line of input contains n integers $b_1, b_2, \dots, b_n$ ($1 \le b_i \le 10^6$).

This is followed by q lines. Each of these q lines contains three integers. The first integer op is either 1 or 2.

- If it is 1, it is followed by two integers k, v ($1 \le k \le n$, $1 \le v \le 10^6$). It means that you should change b_k to v.
- If it is 2, it is followed by two integers l, r ($1 \le l < r \le n$). It means that you should print $f([b_l, b_{l+1}, \dots, b_r])$.

Output

For each query of type 2, print one real number — the answer to the query. Your answer is considered correct if its absolute error or relative error does not exceed 10^{-9}.

Example

input

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

output

```
3.500000000000000
1.000000000000000
```

Note

Let's analyse the sample.

In the first query, we are asked to compute $f([3,1,1,4])$. The answer is 3.5. One optimal sequence of operations is:

- In the first 1.5 seconds do the second operation with $i = 1$.
- In the next 2 seconds do the first operation with $i = 3$.

In the third query, we are asked to compute $f([1,1,1])$. The answer is 1. One optimal sequence of operations is:

- In the first 0.5 seconds do the second operation with $i = 1$.
- In the next 0.5 seconds do the first operation with $i = 2$.

H. Maximum Product?

time limit per test: 1.5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given a positive integer k. For a multiset of integers S, define $f(S)$ as the following.

- If the number of elements in S is less than k, $f(S) = 0$.
- Otherwise, define $f(S)$ as the maximum product you can get by choosing exactly k integers from S.

More formally, let $|S|$ denote the number of elements in S. Then,

- If $|S| < k$, $f(S) = 0$.
- Otherwise, $f(S) = \max\limits_{T \subseteq S, |T| = k} \left(\prod\limits_{i \in T} i \right)$.

You are given a multiset of integers, A. Compute $\sum\limits_{B \subseteq A} f(B)$ modulo $10^9 + 7$.

Note that in this problem, **we distinguish the elements by indices instead of values**. That is, a multiset consisting of n elements always has 2^n distinct subsets regardless of whether some of its elements are equal.

Input

The first line of input contains two integers n and k, where n is the number of elements in A ($1 \le k \le n \le 600$).

The second line of input contains n integers $a_1, a_2, \ldots, a_n$, describing the elements in A ($-10^9 \le a_i \le 10^9$).

Output

Output $\sum\limits_{B \subseteq A} f(B)$ modulo $10^9 + 7$.

Examples

input

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

output

```
10
```

input

```
3 1
1 1 1
```

output

```
7
```

input

```
10 4
-24 -41 9 -154 -56 14 18 53 -7 120
```

output

```
225905161
```

input

```
15 5
0 0 2 -2 2 -2 3 -3 -3 4 5 -4 -4 4 5
```

output

```
18119684
```

Note

Consider the first sample. From the definitions we know that

- $f(\varnothing) = 0$
- $f(\{-1\}) = 0$
- $f(\{2\}) = 0$
- $f(\{4\}) = 0$
- $f(\{-1, 2\}) = -2$
- $f(\{-1, 4\}) = -4$
- $f(\{2, 4\}) = 8$
- $f(\{-1, 2, 4\}) = 8$

So we should print $(0 + 0 + 0 + 0 - 2 - 4 + 8 + 8) \bmod (10^9 + 7) = 10$.

In the second example, note that although the multiset consists of three same values, it still has 8 distinct subsets: $\varnothing, \{1\}, \{1\}, \{1\}, \{1, 1\}, \{1, 1\}, \{1, 1\}, \{1, 1, 1\}$.