

A. Adjacent Digit Sums

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

You are given two numbers x, y . You need to determine if there exists an integer n such that $S(n) = x$, $S(n + 1) = y$.

Here, $S(a)$ denotes the sum of the digits of the number a in the decimal numeral system.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 500$). The description of the test cases follows.

The first line of each test case contains two integers x, y ($1 \leq x \leq 1000, 1 \leq y \leq 1000$).

Output

For each test case, print "NO" if a suitable n does not exist. Otherwise, output "YES".

You can output each letter in any case (for example, "YES", "Yes", "yes", "yEs", "yEs" will be recognized as a positive answer).

Standard Input	Standard Output
7	Yes
1 2	No
77 77	No
997 999	Yes
999 1	No
1000 1	No
1 11	Yes
18 1	

Note

In the first test case, for example, $n = 100$ works. $S(100) = 1$, $S(101) = 2$.

In the second test case, it can be shown that $S(n) \neq S(n + 1)$ for all n ; therefore, the answer is No.

In the fourth test case, $n = 10^{111} - 1$ works, which is a number consisting of 111 digits of 9.

B. Two Large Bags

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

You have two large bags of numbers. Initially, the first bag contains n numbers: a_1, a_2, \dots, a_n , while the second bag is empty. You are allowed to perform the following operations:

- Choose any number from the first bag and move it to the second bag.
- Choose a number from the first bag that is also present in the second bag and increase it by one.

You can perform an unlimited number of operations of both types, in any order. Is it possible to make the contents of the first and second bags identical?

Input

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

The first line of each test case contains an integer n ($2 \leq n \leq 1000$) — the length of the array a . It is guaranteed that n is an even number.

The second line of each test case contains n integers a_1, a_2, \dots, a_n ($1 \leq a_i \leq n$).

It is guaranteed that the sum of n^2 over all test cases does not exceed 10^6 .

Output

For each test case, print "YES" if it is possible to equalize the contents of the bags. Otherwise, output "NO".

You can output each letter in any case (for example, "YES", "Yes", "yes", "yEs", "yEs" will be recognized as a positive answer).

Standard Input	Standard Output
9	Yes
2	No
1 1	Yes
2	Yes
2 1	No
4	Yes
1 1 4 4	No
4	Yes
3 4 3 3	Yes
4	
2 3 4 4	
6	
3 3 4 5 3 3	
6	
2 2 2 4 4 4	
8	
1 1 1 1 1 1 1 4	

10	
9 9 9 10 10 10 10 10 10 10	

Note

Let's analyze the sixth test case: we will show the sequence of operations that leads to the equality of the bags. Initially, the first bag consists of the numbers (3, 3, 4, 5, 3, 3), and the second bag is empty.

1. In the first operation, move the number 3 from the first bag to the second. State: (3, 4, 5, 3, 3) and (3).
2. In the second operation, increase the number 3 from the first bag by one. This operation is possible because the second bag contains the number 3. State: (4, 4, 5, 3, 3) and (3).
3. In the third operation, move the number 4 from the first bag to the second. State: (4, 5, 3, 3) and (3, 4).
4. In the fourth operation, increase the number 4 from the first bag by one. State: (5, 5, 3, 3) and (3, 4).
5. In the fifth operation, move the number 5 from the first bag to the second. State: (5, 3, 3) and (3, 4, 5).
6. In the sixth operation, increase the number 3 from the first bag by one. State: (5, 3, 4) and (3, 4, 5).

As we can see, as a result of these operations, it is possible to make the contents of the bags equal, so the answer exists.

C. Devyatkin

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

You are given a positive integer n . In one operation, you can add to n any positive integer whose decimal representation contains only the digit 9, possibly repeated several times.

What is the minimum number of operations needed to make the number n contain at least one digit 7 in its decimal representation?

For example, if $n = 80$, it is sufficient to perform one operation: you can add 99 to n , after the operation $n = 179$, which contains the digit 7.

Input

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

The only line of each test case contains an integer n ($10 \leq n \leq 10^9$).

Output

For each test case, output the minimum number of operations required for the number n to contain the digit 7.

Standard Input	Standard Output
16	3
51	2
60	1
61	0
777	1
12345689	3
1000000000	5
2002	4
3001	0
977	7
989898986	1
80	2
800001	7
96	0
70	7
15	3
90	

Note

In the first test case, three operations are sufficient: $51 + 9 + 9 + 9 = 78$, which contains the digit 7. It can be shown that it is impossible to achieve the goal in one or two operations.

In the second test case, two operations are sufficient: $60 + 9 + 9 = 78$.

In the third test case, one operation is sufficient: $61 + 9 = 70$.

In the fourth test case, n already contains the digit 7, so no operations are required.

In the fifth test case, you can add 99 to n to obtain a number containing the digit 7.

D. Object Identification

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

This is an interactive problem.

You are given an array x_1, \dots, x_n of integers from 1 to n . The jury also has a fixed but hidden array y_1, \dots, y_n of integers from 1 to n . The elements of array y are **unknown** to you. Additionally, it is known that for all i , $x_i \neq y_i$, and all pairs (x_i, y_i) are distinct.

The jury has secretly thought of one of two objects, and you need to determine which one it is:

- **Object A:** A directed graph with n vertices numbered from 1 to n , and with n edges of the form $x_i \rightarrow y_i$.
- **Object B:** n points on a coordinate plane. The i -th point has coordinates (x_i, y_i) .

To guess which object the jury has thought of, you can make queries. In one query, you must specify two numbers i, j ($1 \leq i, j \leq n, i \neq j$). In response, you receive one number:

- If the jury has thought of **Object A**, you receive the length of the shortest path (in edges) from vertex i to vertex j in the graph, or 0 if there is no path.
- If the jury has thought of **Object B**, you receive the Manhattan distance between points i and j , that is $|x_i - x_j| + |y_i - y_j|$.

You have 2 queries to determine which of the objects the jury has thought of.

Input

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

Interaction

The interaction begins with reading n ($3 \leq n \leq 2 \cdot 10^5$) — the length of the arrays x and y at the start of each test case.

Next, read n integers: x_1, x_2, \dots, x_n ($1 \leq x_i \leq n$) — the elements of array x .

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

The array y_1, y_2, \dots, y_n is **fixed** for each test case. In other words, the interactor is not adaptive.

It is guaranteed that $x_i \neq y_i$ for all $1 \leq i \leq n$, and all pairs (x_i, y_i) are distinct.

To make a query, output "? i j " (without quotes, $1 \leq i, j \leq n, i \neq j$). Then you must read the response to the query, which is a non-negative integer.

To give an answer, output "! A" if **Object A** is hidden or "! B" if **Object B** is hidden (without quotes). Note that printing the answer does not count as one of the 2 queries.

After printing each query do not forget to output the end of line and flush* the output. Otherwise, you will get **Idleness limit exceeded** verdict. If, at any interaction step, you read -1 instead of valid data, your solution must exit immediately. This means that your solution will receive **Wrong answer** because of an

invalid query or any other mistake. Failing to exit can result in an arbitrary verdict because your solution will continue to read from a closed stream. Note that if the query is correct, the response will never be -1 .

Hacks

The first line must contain an integer t ($1 \leq t \leq 1000$) — the number of test cases.

The first line of each test case must contain an integer n ($3 \leq n \leq 2 \cdot 10^5$) — the length of the hidden array.

The second line of each test case must contain n integers — x_1, x_2, \dots, x_n ($1 \leq x_i \leq n$).

The third line of each test case must contain n integers — y_1, y_2, \dots, y_n ($1 \leq y_i \leq n$).

It must be ensured that $x_i \neq y_i$ for all $1 \leq i \leq n$, and all pairs (x_i, y_i) must be distinct.

The fourth line must contain one character: A or B, indicating which of the objects you want to hide.

The sum of n across all test cases must not exceed $2 \cdot 10^5$.

- *To flush, use:
- `fflush(stdout)` or `cout.flush()` in C++;
 - `sys.stdout.flush()` in Python;
 - see the documentation for other languages.

Standard Input	Standard Output
2	? 2 3
3	
2 2 3	? 1 2
1	! A
0	? 1 5
5	? 5 1
5 1 4 2 3	
	! B
4	
4	

Note

In the first test case, $x = [2, 2, 3]$, $y = [1, 3, 1]$ and Object A is guessed.

In the second test case, $x = [5, 1, 4, 2, 3]$, $y = [3, 3, 2, 4, 1]$ and Object B is guessed.

E. White Magic

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

We call a sequence a_1, a_2, \dots, a_n *magical* if for all $1 \leq i \leq n - 1$ it holds that:
 $\min(a_1, \dots, a_i) \geq \text{mex}(a_{i+1}, \dots, a_n)$. In particular, any sequence of length 1 is considered *magical*.

The minimum excluded (MEX) of a collection of integers a_1, a_2, \dots, a_k is defined as the smallest non-negative integer t which does not occur in the collection a .

You are given a sequence a of n non-negative integers. Find the maximum possible length of a *magical* subsequence* of the sequence a .

*A sequence a is a subsequence of a sequence b if a can be obtained from b by the deletion of several (possibly, zero or all) element from arbitrary positions.

Input

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

The first line of each test case contains an integer n ($1 \leq n \leq 2 \cdot 10^5$) — the length of the sequence a .

The second line of each test case contains n integers a_1, a_2, \dots, a_n ($0 \leq a_i \leq 10^9$) — the elements of the sequence a .

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

Output

For each test case, output a single number — the maximum possible length of a *magical* subsequence of the sequence a .

Standard Input	Standard Output
8	5
5	5
4 3 2 1 0	3
6	1
4 3 3 2 1 0	4
4	2
2 0 1 2	2
1	3
777	
4	
1000000000 1 7 9	
2	
0 1	
2	
1 2	
4	

0 1 0 1	
---------	--

Note

In the first test case, the sequence $[4, 3, 2, 1, 0]$ is *magical*, since:

- $\min(4) = 4, \text{mex}(3, 2, 1, 0) = 4. 4 \geq 4$
- $\min(4, 3) = 3, \text{mex}(2, 1, 0) = 3. 3 \geq 3$
- $\min(4, 3, 2) = 2, \text{mex}(1, 0) = 2. 2 \geq 2$
- $\min(4, 3, 2, 1) = 1, \text{mex}(0) = 1. 1 \geq 1$

In the second test case, the sequence $[4, 3, 3, 2, 1, 0]$ is not *magical*, since

$\min(4, 3) = 3, \text{mex}(3, 2, 1, 0) = 4, 3 < 4$. However, the subsequence $[4, 3, 2, 1, 0]$ of this sequence is *magical*, so the answer is 5.

F. Bitwise Slides

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

You are given an array a_1, a_2, \dots, a_n . Also, you are given three variables P, Q, R , initially equal to zero.

You need to process all the numbers a_1, a_2, \dots, a_n , **in the order from 1 to n** . When processing the next a_i , you must perform **exactly** one of the three actions of your choice:

1. $P := P \oplus a_i$
2. $Q := Q \oplus a_i$
3. $R := R \oplus a_i$

\oplus denotes the bitwise XOR operation.

When performing actions, you must follow the *main rule*: it is necessary that after each action, all three numbers P, Q, R are **not** pairwise distinct.

There are a total of 3^n ways to perform all n actions. How many of them do not violate the *main rule*? Since the answer can be quite large, find it modulo $10^9 + 7$.

Input

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

The first line of each test case contains an integer n ($1 \leq n \leq 2 \cdot 10^5$) — the length of the array a .

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

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

Output

For each test case, output the number of ways to perform all n actions without violating the *main rule*, modulo $10^9 + 7$.

Standard Input	Standard Output
5 3 1 7 9 4 179 1 1 179 5 1 2 3 3 2 12 8 2 5 3 9 1 8 12 9 9 9 4 1 1000000000	3 9 39 123 3

Note

In the first test case, there are 3 valid sequences of operations: **PPP**, **QQQ**, **RRR**.

In the second test case, there are 9 valid sequences of operations: **PPPP**, **PPPQ**, **PPPR**, **QQQP**, **QQQQ**, **QQQR**, **RRRP**, **RRRQ**, **RRRR**.

G. Tropical Season

Input file: standard input
Output file: standard output
Time limit: 4 seconds
Memory limit: 1024 megabytes

You have n barrels of infinite capacity. The i -th barrel initially contains a_i kilograms of water. In this problem, we assume that all barrels weigh the same.

You know that **exactly** one of the barrels has a small amount of tropical poison distributed on its surface, with a total weight of 0.179 kilograms. However, you do not know which barrel contains the poison. Your task is to identify this *poisonous* barrel.

All the barrels are on scales. Unfortunately, the scales do not show the exact weight of each barrel. Instead, for each pair of barrels, they show the result of a comparison between the weights of those barrels. Thus, for any two barrels, you can determine whether their weights are equal, and if not, which barrel is heavier. The poison and water are included in the weight of the barrel.

The scales are always turned on, and the information from them can be used an unlimited number of times.

You also have the ability to pour water. You can pour water from any barrel into any other barrel in any amounts.

However, to pour water, you must physically handle the barrel from which you are pouring, so if that happens to be the poisonous barrel, you will die. This outcome must be avoided.

However, you can pour water into the poisonous barrel without touching it.

In other words, you can choose the numbers i, j, x ($i \neq j, 1 \leq i, j \leq n, 0 < x \leq a_i$, the barrel numbered i is **not** poisonous) and execute $a_i := a_i - x, a_j := a_j + x$. Where x is not necessarily an integer.

Is it possible to guarantee the identification of which barrel contains the poison and remain alive using pouring and the information from the scales? You know that the poison is located on **exactly** one of the barrels.

Additionally, we ask you to process q queries. In each query, either one of the existing barrels is removed, or an additional barrel with a certain amount of water is added. After each query, you need to answer whether it is possible to guarantee the identification of the poisonous barrel, given that there is exactly one.

Input

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

The second line contains n integers a_1, a_2, \dots, a_n ($1 \leq a_i \leq 10^6$) — the volumes of water in the existing barrels.

The next q lines contain queries, one per line. Each query is either of the form $+ x$ or $- x$, which means, respectively, adding and removing a barrel with x kilograms of water. It is guaranteed that when the query $- x$ is made, one of the existing barrels has a volume of x , and that there is always at least one barrel remaining throughout the queries. In all queries, $1 \leq x \leq 10^6$.

Output

Output $q + 1$ lines, the answer to the problem before all queries, and after each query. If it is possible to identify the *poisonous* barrel, output "Yes"; otherwise, output "No". You can output the answer in any case

(upper or lower). For example, the strings "yEs", "yes", "Yes", and "YES" will be recognized as positive responses.

Standard Input	Standard Output
4 7 2 2 4 11 - 2 + 4 + 30 + 40 - 4 + 2 + 2	Yes No Yes No Yes No No Yes
6 7 5000 1000 400 400 100 99 + 1 - 5000 - 1 - 400 - 400 - 100 - 99	No Yes Yes Yes No No No Yes

Note

In the first test, before any queries, the weights of the barrels are $[2, 2, 4, 11]$. You can compare the values of the first and second barrels. If the result is not equal, you can definitively conclude that the barrel with the greater weight is poisonous. If the result is equal, both barrels are non-poisonous. Then you can pour all the contents of the first barrel into the second. After that, both the second and third barrels will contain 4 kilograms of water. Now let's compare the values of the second and third barrels. If the result is not equal, the barrel with the greater weight is poisonous. Otherwise, both barrels are non-poisonous. The only barrel that could be poisonous is barrel 4. Thus, with this strategy, we can safely determine the poisonous barrel.