Kinetik Presents CUET Inter University Programming Contest CodeStorm 1.0

https://toph.co/c/cuet-inter-university-codestorm-1-0

Schedule

The contest will run for **5h0m0s**.

Authors

The authors of this contest are Ahasan_1999, Apurba.884537, curly_braces, Iftekhar_Hakim, Ishtiaq, ludirm, Nirjhor, and Reyad18.

Rules

This contest is formatted as per the official rules of ICPC Regional Programming Contests.

You can use Bash 5.0, Brainf*ck, C# Mono 6.0, C++11 GCC 7.4, C++14 GCC 8.3, C++17 GCC 9.2, C++20 Clang 16.0, C++20 GCC 12.1, C11 GCC 12.1, C11 GCC 9.2, Common Lisp SBCL 2.0, D8 11.8, Erlang 22.3, Free Pascal 3.0, Go 1.18, Grep 3.7, Haskell 8.6, Java 1.8, Kotlin 1.1, Lua 5.4, Node.js 10.16, Perl 5.30, PHP 7.2, PyPy 7.1 (2.7), PyPy 7.1 (3.6), Python 2.7, Python 3.11, Python 3.7, Ruby 2.7, Ruby 3.2, Rust 1.57, Swift 5.3, and Whitespace in this contest.

Be fair, be honest. Plagiarism will result in disqualification. Judges' decisions will be final.

Notes

There are 11 challenges in this contest.

Please make sure this booklet contains all of the pages.

If you find any discrepencies between the printed copy and the problem statements in Toph Arena, please rely on the later.

A. Have Some Water

If you have been tired during the contest, just have some water, because this statement is fairly short.

Bears have got an array A of N positive integers. They want to split this array into parts in such a way that each element of the array belongs to exactly one of the parts, and each of the parts forms a consecutive sub-segment of the original array. Cost of a part is the *Least Common Multiple (LCM)* of its elements. Bears want that sum of the cost of their parts to be minimized. Tell them the minimum sum.

Least Common Multiple (LCM) of K positive *integers is defined as the smallest positive integer that is divisible all of those* K integers.

Input

The first line contains an integer T $(1 \leq T \leq 3 \times 10^5)$ — the number of test cases.

The first line of each test case contains an integer N $(1 \leq N \leq 3 \times 10^5)$ — the length of the given array A.

Second line of each test case contains N positive integers, denoting $A_1, A_2, \ldots, A_N$ $(1 \leq A_i \leq 3 \times 10^5)$.

Sum of N over all test cases does not exceed 3×10^5.

Output

For each test case, print an integer denoting the minimum sum in a line.

Samples

Input	Output
2 5 2 2 1 3 3 3 1 1 1	5 1

B. Break The Bar

You have a massive rectangular chocolate bar of height N and width M. So it consists of $N \times M$ squares. You want to share it with P friends, giving each of them a single 1×1 square piece. At each step, you can break any single rectangular piece either horizontally or vertically into two rectangular pieces. You can only break along the lines between the squares: that means, at each step, all the rectangular pieces must have integer heights and integer widths. You have to find the minimum number of steps needed to get P pieces of 1×1 sized single squares.

Input

The first line contains an integer T denoting the number of test cases.

Each of the following T lines contains three integers N, M, and P denoting the dimensions of the chocolate bar and the number of friends you have.

Constraints

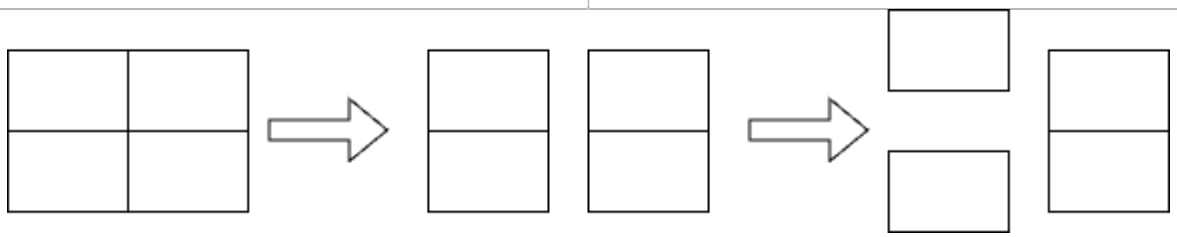
- $1 \leq T \leq 2 \times 10^5$
- $1 \leq N, M \leq 10^9$
- $1 \leq P \leq 2 \times 10^5$
- The sum of P across all test cases won't exceed 2×10^5.

Output

For each test case, print a single integer denoting the minimum number of steps to get P pieces of 1×1 sized squares, or print -1 if it's impossible to get an answer.

Samples

Input	Output
3 2 2 1 4 2 3 3 5 17	2 4 -1
For the first test case, we need two steps to get one 1×1 sized square.	

Input	Output
	

C. Three Arrays

Alex has a weird hobby. Whenever Alex has two integer arrays A and B, each of length n, he creates another integer array C of length n, where $C_i = \left\lfloor \frac{A_i}{B_i} \right\rfloor$ for each $1 \leq i \leq n$.

In this problem, you are given the array A $(1 \leq A_i \leq n)$ of length n, and you have to find the array B of length n, such that if Alex creates the array C using them as mentioned above, it follows the following rules in order:

1. $1 \leq B_i, C_i \leq n$ should hold for each $1 \leq i \leq n$.
2. Array C should be a non-decreasing array. Formally, $C_i \leq C_{i+1}$ should hold for each $1 \leq i < n$.
3. If there are multiple possible arrays B that fulfill the above two conditions, print the one with the smallest possible value when we take the bitwise **AND** of all the numbers in B.
4. If there are still multiple possible arrays B that fulfill the above three conditions, print the lexicographically smallest one.

You have to help Alex in T possible test scenarios.

Input

The first line contains the integer T $(1 \leq T \leq 10,000)$, denoting the number of test cases. Then follows the description of these T tests.

Each test case is described by an integer n $(1 \leq n \leq 10^5)$ denoting the length of the arrays. It is followed by n space-separated integers $A_1, A_2, \ldots, A_n$ $(1 \leq A_i \leq n)$ on the next line, denoting the array A.

Here, it is guaranteed that the sum of n over all T test cases does not exceed 10^5.

Output

On a single line, print n space-separated integers denoting the array B satisfying all the conditions mentioned above. This array always exists and is unique.

Trailing spaces are fine and they do not trigger a wrong answer.

Samples

Input	Output
5 1 1 2 1 1 3 3 1 3 4 1 2 2 3 5 1 2 1 3 1	1 1 1 2 1 1 1 2 1 1 1 2 1 2 1

For the first two tests, there is a unique array B that satisfies the first three conditions in the problem statement.

For the fourth test: the array $B = [1, 1, 1, 1]$ is the lexicographically smallest array making the array C sorted. However, the bitwise **AND** of the elements is 1. We can achieve a lower bitwise **AND** value of 0 using the array $B = [1, 2, 1, 1]$, which is lexicographically the smallest such array.

D. Max SMEX

The **MEX** (minimum excluded) of a set of integers is the smallest non-negative integer that does not belong to the set.

For a string S containing only lowercase Latin letters, if you put the frequencies of each character from `a` to `z` into a set, then the MEX of that set is called the **SMEX** of that string. Here, the frequency of a character is the number of times it appears in S.

Given a string S, your task is to print the maximum SMEX value among all of its substrings.

You have to deal with T possible test scenarios.

Input

Input starts with an integer T $(1 \leq T \leq 10^5)$ denoting the number of test cases.

The next T lines each contain one string S.

The sum of lengths of all the strings in an input file does not exceed 10^5.

Output

For each test case, print the answer in a single line.

Samples

Input	Output
2 abca bacabcb	3 3

E. Missing Integer

This is an **Interactive Problem**.

The judge has an integer A $(1 \leq A \leq 10^9)$. You can do two kinds of operations on A,

1. `MUL` X — Multiply A by X. X has to be an integer in range $[1, 10^9]$. After the multiplication, the value of A cannot exceed 10^{18}.
2. `DIV` Y — Divide A by Y. Y has to be an integer in range $[1, 10^9]$. To divide here refers to **integer division** or **floor**.

After each operation, the judge will respond with a string: <, >, =, or `WA`.

The response is `WA` if you did an invalid operation or exceeded the number of allowed operations. Your program should terminate after this kind of response.

Otherwise, the judge will respond with the relation between the current value of A, and the initial value of A. For example, if the current value of A is 64 and the initial value of A was 23, the judge will reply with >.

Your task is to find the initial value of A by doing **at most** 64 operations of type 1 and 2 **in total**.

When you're ready with the answer, print `VERIFY` A. Here, A is the initial value that the judge chose.

After this, the judge will respond with `AC` or `WA`. The response is `AC` if your result was correct, and you should continue to the next test case in this case. Otherwise, it is `WA`, and your program should terminate.

Input

The first line contains a single integer T $(1 \leq T \leq 1000)$ — the number of test cases.

Interaction Details

For each query, print the query in a line and flush the output. The query should be in the format mentioned above.

After each query, read a string: the response from the judge.

After a `VERIFY` query, move on to the next test case if there is any, terminate your program otherwise.

To flush the output stream, you may use:

- `fflush(stdout)` in C/C++
- `stdout.flush()` in Python

Output

A sample interaction is given below for a test with one test case and $A = 5$.

```
>> 1
<< DIV 6
>> <
<< MUL 100
>> <
<< VERIFY 60
>> WA
```

Another sample with a single test case and for $A = 23$.

```
>> 1
<< DIV 10
>> <
<< MUL 10
>> <
<< MUL 2
>> >
<< VERIFY 23
>> AC
```

Here `>>` indicates what your program reads and `<<` indicates what your program writes. These symbols are here to make it easy to understand. You do not have to print such symbols from your program.

F. Solo Leveling

Introducing a new game, named **Solo Leveling**, where a player is tasked with defeating N monsters. The i^{th} $(1 \leq i \leq N)$ monster has B_i hit points. As the player embarks on this challenging journey, he begins with an initial set of resources: P hit points, X elixirs and A amulets. When confronted with a monster, the game presents the player with the following possible ways to defeat the monster:

1. If the player's current hit points are equal to or greater than the monster's hit points, the player successfully defeats the monster, taking damage equal to the monster's hit points, which is then subtracted from the player's current hit points.
2. If the player's current hit points are less than the monster's hit points, the player can use elixirs to increase his current hit points. Each elixir used adds **one** unit to the player's hit points. Then defeat the monster according to the first way that is mentioned above.
3. The player can deploy an amulet which allows the player to instantly defeat the monster without taking any damage, and it doesn't reduce the player's current hit points.

The player can encounter the i^{th} monster only if he has defeated the $(i-1)^{th}$ monster. Now, find the **maximum** number of monsters the player can successfully defeat by making optimal use of elixirs and amulets. Navigate through the game wisely, considering the available hit points, elixirs, and amulets to defeat as many monsters as possible!

Input

The first line contains an integer T $(1 \leq T \leq 10)$, the number of test cases.

The first line of each test case contains four integers N $\left(1 \leq N \leq 5 \cdot 10^5\right)$ — the number of monsters, P $\left(0 \leq P \leq 10^9\right)$ — the initial hit points of the player, X $\left(0 \leq X \leq 10^9\right)$ — the initial number of elixirs and A $(0 \leq A \leq N)$ — the initial number of amulets.

The second line of each test case contains N integers $B_1, B_2, \ldots, B_N$ $\left(0 \leq B_i \leq 10^9\right)$, the hit points of the monsters.

Output

For each test case, print the **maximum** number of monsters the player can successfully defeat by making optimal use of elixirs and amulets.

Samples

Input	Output
2 5 3 4 1 2 5 3 2 6 2 2 2 0 2 2	4 2

For first test case, the player initially has 3 hit points, 4 elixirs and 1 amulet. The player can defeat maximum 4 monsters by following steps:

1. The first monster has less hit points than the current hit points of the player. The player defeats the monster taking damage of 2 hit points and advances with 1 hit point.
2. To defeat the second monster, player uses an amulet.
3. The third monster has more hit points than the current hit points of the player. To defeat this monster, player uses 2 elixirs. After defeating the monster, the player advances with 0 hit point.
4. The fourth monster has more hit points than the current hit points of the player. To defeat this monster, player uses 2 elixirs. After defeating the monster, the player advances with 0 hit point.
5. The fifth monster has more hit points than the current hit points of the player. The player has no more elixir and amulet. So, the player can not defeat the monster.

G. Demon Slayer

Introducing another new game, named **Demon Slayer**, that is presented in an infinite 2D plane in Cartesian coordinate system. Here, the player is tasked with eliminating demons using laser beams. The game is set with N laser beam emitters and M demons. For each i, where $(1 \leq i \leq N)$, there is a laser beam emitter along the *X*-axis at point $(i, 0)$. The player can activate all laser beam emitters only once at the same time. Each laser beam emitter has the artificial intelligence ability to independently rotate to the angle where the most demons can be eliminated. The angle may not necessarily be an integer. The emitted laser travels in a straight line to infinity, and each emitter can eliminate all demons located on the straight line passing through it.

A demon's location during the laser beam activation is represented by coordinates (X, Y) in the plane. A demon is considered eliminated if its coordinates lies within the path of a laser beam. Each demon is at unique coordinates.

Now, your task is to determine the **maximum** number of demons that can be eliminated by each laser beam emitter. A demon can be eliminated by multiple laser beam emitters.

Input

The first line contains two integers N and M $\left(1 \leq N \leq 3 \cdot 10^5,\ 1 \leq M \leq 500\right)$, the number of laser beam emitters and the number of demons respectively.

The next M lines represent the coordinates of the demons. The i-th line contains two integers X_i and Y_i $\left(1 \leq X_i, Y_i \leq 10^9\right)$, the i-th demon's coordinates.

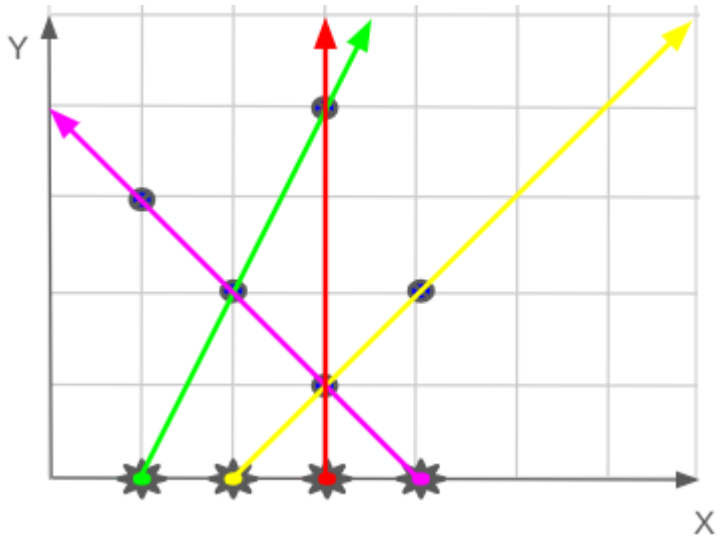
Output

Print a line containing N space separated integers, where the i-th integer will represent the **maximum** number of demons that can be eliminated by the laser beam emitter located at the $(i, 0)$ coordinates.

Samples

Input	Output
4 5 1 3 2 2	2 2 2 3

Input	Output
3 1 4 2 3 4	



Input	Output
1 2 1 1 2 1	1

H. Turn Them Off

Imagine you get an internship to manage lights in Chittagong Port. The lights are numbered from 1 to N. Initially all the lights are turned off.

For Q days, you will be given to do one of the following three tasks:

- $1\ L\ R$: Turn off all lights in the range from L to R.
- $2\ L\ R$: Turn on all lights in the range from L to R.
- 3: Find the number of magic steps that would be needed, if you want to turn off all the currently turned-on lights. Or report that it is impossible. Note that the actual state of the lights does not change at this third task.

In each magic step, you have to toggle the P^{th}, light where P is the total number of currently turned-on lights.

Input

The first line of the input contains two integers N and Q $(1 \leq N \leq 2 \times 10^5, 1 \leq Q \leq 2 \times 10^5)$ denoting the number of lights and number of days. The next Q lines contain the description of the tasks to perform in the format $1\ L\ R$ or $2\ L\ R$ or 3 $(1 \leq L \leq R \leq N)$.

Output

For each day of the format 3, print the number of magic steps needed if you want to turn off all the current lights. Print -1 if it's impossible to turn off all the lights.

Samples

Input	Output
3 4 2 1 3 3 1 2 2 3	3 4
Initially, the lights are off, 000. After the first day, the state of the lights become, 111. On the second day, 3 magic steps would be needed to turn off all lights $(111 \rightarrow$	

Input	Output
$110 \rightarrow 100 \rightarrow 000$). After the third day, the state of the lights become, 101. On the fourth day, 4 magic steps would be needed to turn them off $(101 \rightarrow 111 \rightarrow 110 \rightarrow 100 \rightarrow 000)$.	

I. Mexy

Consider a rooted tree where each vertex has an integer value assigned to it. The *score* of such a tree is computed as follows:

- For each vertex in the tree, find the $\texttt{MEX}$ of all the values assigned to the vertices in its subtree. Recall that the $\texttt{MEX}$ of a set of non-negative integers is defined as the smallest non-negative integer that is not present in that set. For example: $\texttt{MEX}([1, 2, 3]) = 0$ and $\texttt{MEX}([2, 0, 5, 1, 6, 100, 3]) = 4$.
- The *score* of the tree is the sum of these $\texttt{MEX}$ values for all vertices.

So what's the problem?

You have a tree of n vertices indexed from 0 to $n - 1$. Each vertex also has an integer value between 0 and $n - 1$ assigned to it. For each vertex u, you have to print the *score* of the tree considering the tree is rooted at the vertex u.

Input

The first line contains the integer n $(1 \leq n \leq 10^5)$ denoting the number of vertices.

The second line contains n space-separated integers $v_0, v_1, \ldots, v_{n-1}$ between 0 and $n - 1$, where v_i denotes the value assigned to vertex i for each $0 \leq i < n$.

Each of the next $n - 1$ lines contains a pair of integers between 0 and $n - 1$, denoting the endpoints of an edge in the tree. It is guaranteed that these edges form a tree.

Output

Print n space-separated integers $s_0, s_1, \ldots, s_{n-1}$ where s_i is the *score* of the tree when vertex i is the root, for each $0 \leq i < n$.

Samples

Input	Output
1 0	1
The tree consists of a single vertex with the value 0. So the $\texttt{MEX}$ is 1.	

Input	Output
2 1 0 0 1	3 2

Input	Output
5 0 1 0 2 3 0 1 0 2 2 3 1 4	5 6 6 8 9

Input	Output
6 1 2 1 2 2 2 0 1 1 2 1 3 1 4 1 5	0 0 0 0 0 0
All the MEX values in this case are zeros.	

Input	Output
8 0 4 2 1 0 1 1 4 0 1 1 2 1 3 2 4 3 5 5 6 1 7	8 6 7 7 9 10 13 9

J. Burden of Expectation

Expectations can hurt you both physically and psychologically.

Morty has an undirected graph of n nodes and m edges, where some of the nodes are broken. He placed k tokens on the graph, where the i -th token is placed on node a_i. There can be multiple tokens on a node. He performs the following operation exactly q times.

- Select a token from 1 to k uniformly at random. Say the token is currently in node u. If u is a broken node, then do nothing. Otherwise, pick an adjacent node of u uniformly at random and move the token to that node. If there's no adjacent node, then do nothing.

The final score after all the operations is $x \times y$, where x is the number of tokens in non-broken node and y is the number of tokens in broken node. As Morty is not so intelligent, he asked your help to calculate the expected value of the final score.

It can be proven that the expected value can be represented as an irreducible fraction $\frac{P}{Q}$, where P and Q are integers and Q is coprime to 998244353. Output the value of $P \times Q^{-1}$ modulo 998244353.

Input

The first line of the input contains four integers n, m, k, q $(1 \leq n, k, q \leq 300; 0 \leq m \leq \frac{n \times (n-1)}{2})$ — the number of nodes, the number of edges, the number of tokens and the number of operations respectively.

The second line contains n integers $s_1, s_2, \ldots, s_n$ $(0 \leq s_i \leq 1)$ — if the i -th node is a broken node then $s_i = 1$ and 0 otherwise.

The third line contains k integers $a_1, a_2, \ldots, a_k$ $(1 \leq a_i \leq n)$ — the i-th token is placed in node a_i.

The following m lines describe the edges. The i -th line contains two integers u_i and v_i $(1 \leq u_i, v_i \leq n, u_i \neq v_i)$ — denoting that the i-th edge connects node u_i and v_i.

The input is guaranteed to have no multiple edges or self loops.

Output

Print a single integer — the expected value of the final score modulo 998244353.

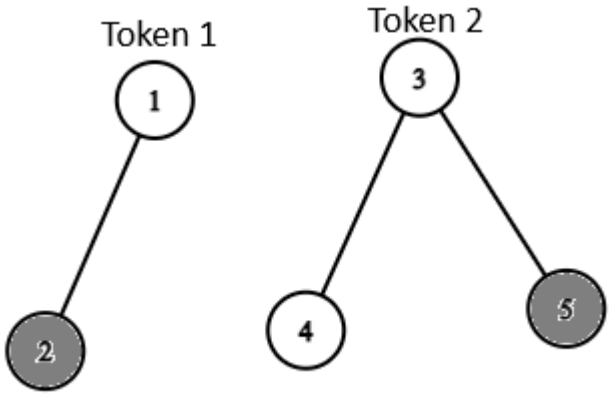
Samples

Input	Output
5 3 2 1 0 1 0 0 1 1 3 1 2 3 4 3 5	249561089

There are 5 nodes in the graph. Node 2 and 5 are broken nodes. There are two tokens on node 1 and 3 respectively. We have to do 1 operation in the graph which can be one of the followings.

- Select token 1 with probability $\frac{1}{2}$ and move that token from node 1 to node 2 with probability 1. The final score is 1 with probability $\frac{1}{2}$.
- Select token 2 with probability $\frac{1}{2}$ and move that token from node 3 to node 4 with probability $\frac{1}{2}$. The final score is 0 with probability $\frac{1}{2} \times \frac{1}{2}$.
- Select token 2 with probability $\frac{1}{2}$ and move that token from node 3 to node 5 with probability $\frac{1}{2}$. The final score is 1 with probability $\frac{1}{2} \times \frac{1}{2}$.

The expected score is $\frac{1}{2} \times 1 + \frac{1}{2} \times \frac{1}{2} \times 0 + \frac{1}{2} \times \frac{1}{2} \times 1 = \frac{3}{4}$.



Input	Output
7 5 4 3 0 1 0 0 0 0 1 1 3 4 5 1 2 2 3 3 1 5 6 6 7	994344963

Input	Output
1 0 1 1 1 1	0

K. Not a Giveaway

What? I told you this is not a giveaway.

Still skeptical?

Okay, fine.

Tell me which one is bigger: 2^n or n^2?

Tough? Don't worry, I'll give you a value of n.

Input

The only line contains an integer n $(1 \leq n \leq 1000)$.

Output

If 2^n is bigger than n^2, print `POWER`.

If 2^n is smaller than n^2, print `SQUARE`.

If both are equal, print `SAME`.

Samples

Input	Output
1	POWER
$2^1 = 2$ and $1^2 = 1$. So $2^n > n^2$ for $n = 1$.	

Input	Output
2	SAME
Since $2^2 = 4$, for $n = 2$ we have $2^n = n^2$.	

Input	Output
3	SQUARE
$2^3 = 8$ and $3^2 = 9$. So $2^n < n^2$ for $n = 3$.	