Problem A. Kill Two Birds with One Stone

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

Why are lots of things in twos?
Hands on clocks, and gloves, and shoes,
Scissor-blades, and water taps,
Collar studs, and luggage straps,
Walnut shells, and pigeons' eggs,
Arms and eyes and ears and legs
Will you kindly tell me who's
So fond of making things in twos?

— English For Today, Class 5, 2011 Edition

You are given a binary matrix M of size $n \times m$, where rows are numbered from 1 to n from top to bottom and columns are numbered from 1 to m from left to right. The position (r, c) denotes the cell in row r and column c, where r and c are integers such that $1 \leq r \leq n$ and $1 \leq c \leq m$.

Initially, all entries of the matrix are 1, except for **exactly two cells** which contain 0. The two cells containing 0 are located at positions (r_1, c_1) and (r_2, c_2).

You can apply the following two types of operations on the matrix:

- `1 r c` – choose any position (r, c) with $1 \leq r < n$ and $1 \leq c \leq m$, and decrease both $M_{r,c}$ and $M_{r+1,c}$ by 1.
- `2 r c` – choose any position (r, c) with $1 \leq r \leq n$ and $1 \leq c < m$, and decrease both $M_{r,c}$ and $M_{r,c+1}$ by 1.

In summary, in one move, you can select any two adjacent cells (sharing a side) in the matrix, and decrease both of the entries by 1. You can apply the operations any number of times in any order. Your task is to determine whether it is possible to transform the matrix into a **null matrix** (i.e., a matrix where all entries are 0).

Input

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

Each test case consists of a single line containing six space-separated integers n, m, r_1, c_1, r_2, c_2 ($1 \leq n, m \leq 5 \cdot 10^5$; $1 \leq r_1, r_2 \leq n$; $1 \leq c_1, c_2 \leq m$; $(r_1, c_1) \neq (r_2, c_2)$; $n \cdot m > 2$) — the dimensions of the matrix and the positions of the two cells initially containing 0.

It is guaranteed that the sum of $n \cdot m$ over all test cases does not exceed 10^6.

Output

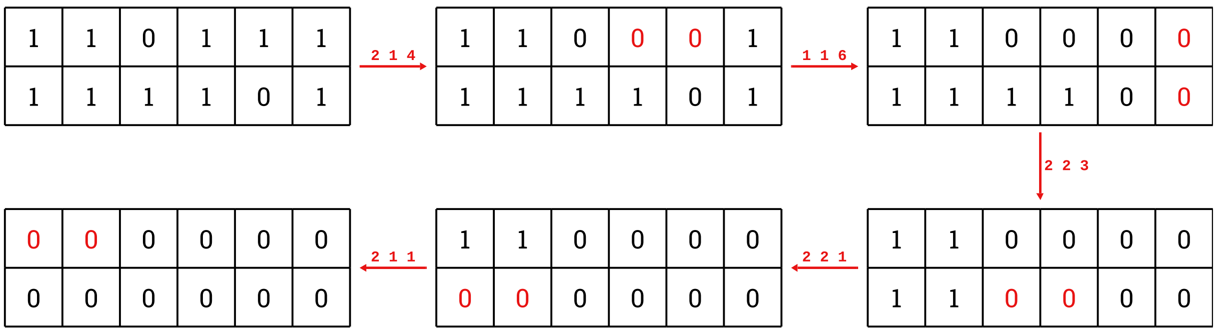
For each test case, output a single word in a line — `YES` if it is possible to transform the matrix into a null matrix, and `NO` otherwise.

Example

standard input	standard output
5 2 6 1 3 2 5 4 4 1 1 2 1 2 2 1 1 2 2 3 3 2 1 3 3 8 7 5 2 2 4	YES YES NO NO YES

Note

The following is an illustration of one of the possible sequences of operations to nullify the given matrix for the first test case:



Problem B. K Floors Down

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	256 megabytes

You are given an integer array a of length n, and an integer k.

Consider all sequences $(x_1, x_2, \ldots, x_k)$ of length k, where each element is chosen independently as one of n elements from the array a (it is allowed to repeat elements). There are exactly n^k such sequences.

For any chosen sequence, define:

- $f_1 = x_1$,
- for each $i = 2, 3, \ldots, k$: $f_i = \left\lfloor \dfrac{f_{i-1}}{x_i} \right\rfloor$, where $\lfloor \cdot \rfloor$ denotes the floor function.

The contribution of the sequence is the final value f_k. Your task is to compute the sum of contributions over all n^k sequences. As the answer can be large, output it modulo 998244353.

Input

The first line contains a single integer t ($1 \le t \le 10^5$) — the number of test cases.

The first line of each test case contains two integers n and k ($1 \le n \le 10^6$, $1 \le k \le 10^9$) — the length of the array and the number of elements in each sequence.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the elements of the array.

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

Output

For each test case, output a single integer — the sum of contributions modulo 998244353.

Example

standard input	standard output
3 2 2 1 2 3 1 3 1 3 3 4 1 2 3	4 7 15

Note

In the first test case, we have $a = [1, 2]$ and $k = 2$. The 4 sequences and their contributions are:

- $[a_1, a_1] = [1, 1]$, $f_1 = 1$, $f_2 = \lfloor \frac{1}{1} \rfloor = 1$. Contribution: 1.
- $[a_1, a_2] = [1, 2]$, $f_1 = 1$, $f_2 = \lfloor \frac{1}{2} \rfloor = 0$. Contribution: 0.
- $[a_2, a_1] = [2, 1]$, $f_1 = 2$, $f_2 = \lfloor \frac{2}{1} \rfloor = 2$. Contribution: 2.
- $[a_2, a_2] = [2, 2]$, $f_1 = 2$, $f_2 = \lfloor \frac{2}{2} \rfloor = 1$. Contribution: 1.

The sum of contributions is $1 + 0 + 2 + 1 = 4$.

Problem C. Pattern Purifier

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

A research lab stores its data as a string consisting of lowercase English letters.

During preprocessing, the lab employs an automated tool called the *Pattern Purifier*. This tool can detect and remove any **even-length** substring* that is a **palindrome**†. Whenever a substring is removed, the remaining parts of the string are concatenated together, preserving their relative order.

Your task is to determine whether the entire string can be deleted by repeatedly applying such removals.

*A string a is a substring of a string b if a can be obtained from b by deletion of several (possibly, zero or all) characters from the beginning and several (possibly, zero or all) characters from the end. For example, `riz` is a substring of `horizon`, but `rizz` and `ozon` are not.

†A palindrome is a string that reads the same backward and forward. For example, `abba` and `racecar` are palindromes, while `hello` is not.

Input

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

Each test case consists of a single string s ($1 \le |s| \le 3 \cdot 10^5$) consisting of lowercase English letters.

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

Output

For each test case, print `YES` if the entire string can be deleted, and `NO` otherwise.

Example

standard input	standard output
5	YES
hanoonnnah	NO
abcd	YES
dammittredderimpullupmad	NO
uwu	NO
manacher	

Note

In the first test case, one possible sequence of deletions is shown below (the underlined substring is the one being removed and ε denotes the empty string):

$$\texttt{ha}\underline{\texttt{noon}}\texttt{nnah} \to \underline{\texttt{hannah}} \to \varepsilon.$$

In the second test case, no deletion is possible, so the string cannot be removed.

In the third test case, one possible sequence of deletions is:

$$\texttt{dammitt}\underline{\texttt{redder}}\texttt{impullupmad} \to \texttt{dammittim}\underline{\texttt{pullup}}\texttt{mad} \to \underline{\texttt{dammittimmad}} \to \varepsilon.$$

In the fourth test case, `uwu` is a palindrome, but not an even-length one, so no deletion is possible.

Problem D. The AND, The OR, and The XOR

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

In the lawless lands of the Binary West, three legendary figures govern the fate of numbers: The AND, The OR, and The XOR.

The AND is strict, seeking only the common ground among a group. The OR is greedy, claiming every bit of influence it can find. Finally, there is The XOR — the chaotic force that measures the tension between the strictness of The AND and the greed of The OR.

You are given a lineup of n outlaws, where a_i represents the bounty on the head of the i-th outlaw. You must choose a posse (subsequence) of at least two outlaws to minimize the tension.

Formally, for a chosen subsequence $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$ (where $1 \le i_1 < i_2 < \cdots < i_k \le n$), the tension score is defined as:

$$(a_{i_1} \mathbin{\&} a_{i_2} \mathbin{\&} \cdots \mathbin{\&} a_{i_k}) \oplus (a_{i_1} \mid a_{i_2} \mid \cdots \mid a_{i_k})$$

Here, &, |, and $\oplus$ denote the bitwise AND, OR, and XOR operators, respectively.

Find the minimum possible tension score among all subsequences of size **at least** 2.

Input

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

The first line of each test case contains an integer n ($2 \le n \le 3 \cdot 10^5$) — the number of outlaws.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$) — the bounties on the heads of the outlaws.

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

Output

For each test case, output a single integer — the minimum possible tension score among all subsequences of size at least 2.

Example

standard input	standard output
2 3 1 2 3 2 1 1	1 0

Note

In the first test case, the possible subsequences with size at least 2 and their scores are:

- (a_1, a_2): $(1 \mathbin{\&} 2) \oplus (1 \mid 2) = 0 \oplus 3 = 3$
- (a_1, a_3): $(1 \mathbin{\&} 3) \oplus (1 \mid 3) = 1 \oplus 3 = 2$
- (a_2, a_3): $(2 \mathbin{\&} 3) \oplus (2 \mid 3) = 2 \oplus 3 = 1$
- (a_1, a_2, a_3): $(1 \mathbin{\&} 2 \mathbin{\&} 3) \oplus (1 \mid 2 \mid 3) = 0 \oplus 3 = 3$

So the minimum score is 1.

Problem E. A Slice of Pi

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes

Chef Luigi, the legendary culinary artist who defeated Chef Gusteau for the "Golden Ladle" award, is now crafting his masterpiece: the "Paradox Pizza".

The pizza is a perfect circle centered at the origin $(0, 0)$ with radius r, topped with n toppings. The i-th topping is at the coordinate (x_i, y_i), and has a tastiness value v_i — positive for crowd-pleasers like mozzarella, negative for controversial choices like pineapple. It is guaranteed that no coordinate (x_i, y_i) is outside the pizza.

As a food critic, you want to eat a slice from this pizza that maximizes the flavor intensity. A slice is the region bounded by two radii and the arc connecting their endpoints on the circle. Points on the boundary (the radii and the arc) are considered inside the slice. The intensity of a slice is defined as the absolute value of the sum of the tastiness values of all toppings within it, i.e., $\left|\sum_{1 \le j \le k} v_{i_j}\right|$ where $i_1, i_2, \ldots, i_k$ are the indices of the toppings that are inside the slice.

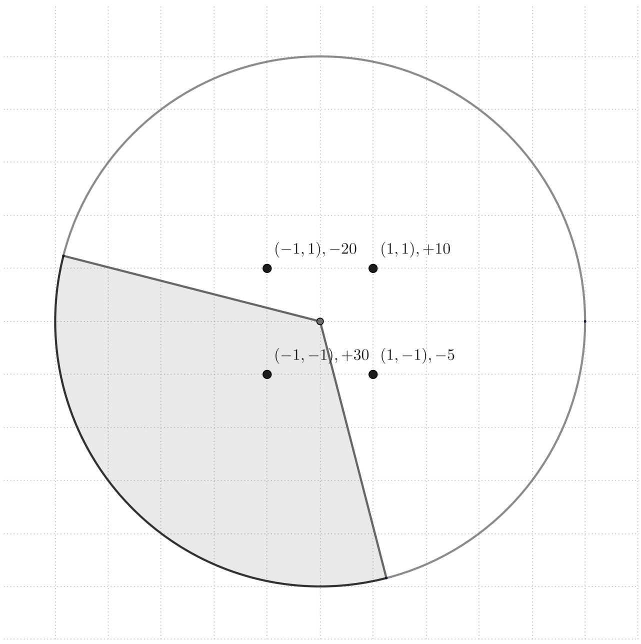


Figure: Explanation for the first sample.

However, you're on a diet and can only eat a slice with area at most A. Find the maximum flavor intensity achievable.

Input

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

The first line of each test case contains three integers n, r, and A ($1 \le n \le 3 \cdot 10^5$, $2 \le r \le 10^4$, $1 \le A \le 10^9$) — the number of toppings, the radius of the pizza, and the maximum allowed area of the slice.

Each of the next n lines contains three integers x_i, y_i, v_i ($x_i^2 + y_i^2 \le r^2$, $(x_i,\, y_i) \ne (0,\, 0)$, $-10^9 \le v_i \le 10^9$) — the coordinates and tastiness value of the i-th topping. Multiple toppings can have the same coordinates.

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

Output

For each test case, output a single integer — the maximum flavor intensity achievable.

Example

standard input	standard output
2 4 5 39 1 1 10 -1 1 -20 -1 -1 30 1 -1 -5 3 5 100 1 1 5 2 0 -10 0 2 -10	30 20

Note

In the first test case, the total area of the pizza is $\pi r^2 \approx 78.5$, and $A = 39$ allows a slice slightly less than half the pizza. Taking the slice containing only the topping with value 30 (at position $(-1, -1)$) gives intensity $|30| = 30$, which is the maximum possible.

In the second test case, A covers the entire pizza. Taking a slice that excludes the topping with value 5 and includes both -10s gives intensity $|(-10) + (-10)| = 20$, which is greater than $|5 + (-10) + (-10)| = 15$ (including all toppings).

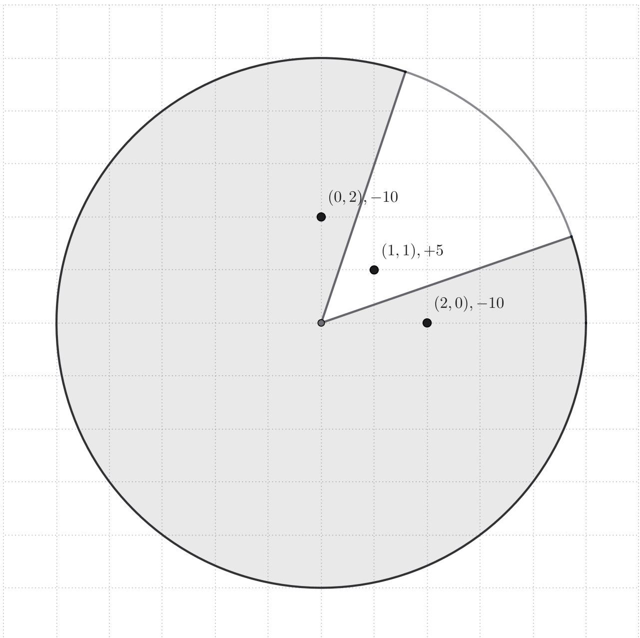


Figure: Explanation for the second sample.

Problem F. Survival of the Fated

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

You are organizing a tournament for n gladiators. The i-th gladiator has a strength level a_i.

The tournament consists of exactly $n - 1$ duels. In each duel:

- Two distinct gladiators are chosen uniformly at random from those remaining in the tournament. Let their strengths be x and y. Note that two distinct gladiators may have equal strengths.
- The duel generates a brutality of $|x - y|$.
- One of the two gladiators is chosen uniformly at random to be eliminated, while the other remains in the tournament.

The process continues until only one gladiator remains. Compute the expected total brutality of the tournament modulo 998244353.

Input

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

The first line of each test case contains an integer n ($2 \le n \le 3 \cdot 10^5$) — the number of gladiators.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$) — the strength levels of the gladiators.

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

Output

For each test case, output a single integer — the expected total brutality modulo 998244353.

Formally, let $M = 998244353$. It can be shown that the answer can be expressed as an irreducible fraction $\frac{p}{q}$, where p and q are integers and $q \not\equiv 0 \pmod{M}$. Output the integer equal to $p \cdot q^{-1} \pmod{M}$.

Example

standard input	standard output
5 3 1 4 2 4 -5 -1 -3 -2 5 10 10 10 10 10 3 1000000000 -1000000000 0 6 3 8 -4 7 -2 5	4 499122183 0 337429843 665496265

Note

In the first test case, let the gladiators be A, B, C with strengths 1, 4, 2 respectively. The tournament tree looks like:

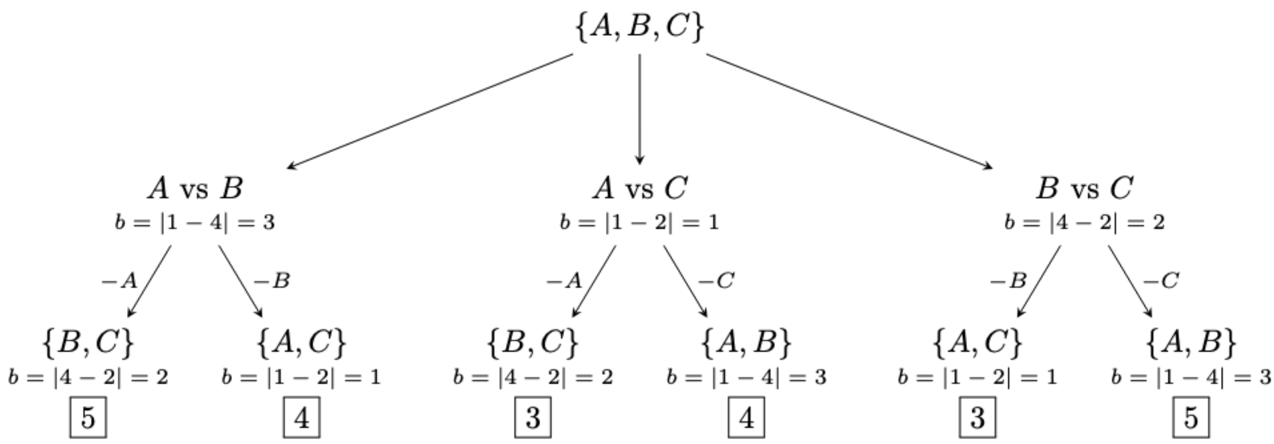


Figure: The tournament tree for the first test case.

Here, b denotes brutality, $-X$ on an edge means gladiator X is eliminated, and the boxed numbers are the total brutality for each scenario. There are 6 equally likely scenarios, so the expected total brutality is:

$$\frac{5+4+3+4+3+5}{6} = \frac{24}{6} = 4.$$

Problem G. Pascal's Tree

Input file:	`standard input`
Output file:	`standard output`
Time limit:	7 seconds
Memory limit:	1024 megabytes

You are given a tree consisting of n nodes rooted at node 1, and a permutation* p of length n.

We define a transformation on a sequence of nodes $a_1, a_2, \ldots, a_k$ ($k \geq 2$) that produces a new sequence $a'_1, a'_2, \ldots, a'_{k-1}$, where $a'_i = \text{LCA}^\dagger(a_i, a_{i+1})$ for $1 \leq i \leq k-1$.

Let $S_0 = p$. For $i \geq 1$, let S_i be the sequence obtained by applying the transformation to S_{i-1}. Note that S_i has length $n - i$.

Your task is to calculate the sum of elements for each sequence $S_0, S_1, \ldots, S_{n-1}$.

---

* A permutation of length n is a sequence of n integers where each integer from 1 to n appears exactly once. For example, $[1, 3, 2]$ is a permutation, but $[2, 3, 2]$ and $[4, 1, 2]$ are not.

† The LCA (Lowest Common Ancestor) of two nodes in a rooted tree is the deepest node that is an ancestor of both nodes. A node is considered an ancestor of itself.

Input

The first line contains a single integer t ($1 \leq t \leq 10^5$) — the number of test cases.

The first line of each test case contains an integer n ($1 \leq n \leq 10^6$) — the number of nodes.

Each of the next $n - 1$ lines contains two integers u and v ($1 \leq u, v \leq n$) indicating there is an edge between nodes u and v. It is guaranteed that the given edges form a tree.

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

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

Output

For each test case, print n space-separated integers — the sum of elements of $S_0, S_1, \ldots, S_{n-1}$ respectively.

Example

standard input	standard output
3 3 2 3 1 3 3 2 1 5 1 2 2 5 2 4 3 1 5 4 2 1 3 6 1 3 5 2 4 1 3 5 2 6 6 2 3 5 4 1	6 4 1 15 6 4 2 1 21 10 8 5 2 1

Note

In the second test case, the tree and the transformation process are shown below:

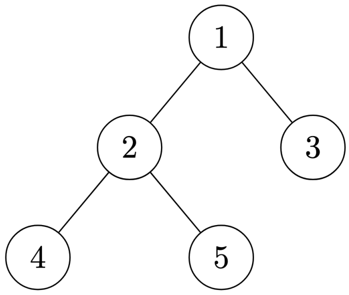


Figure: The tree for the second test case.

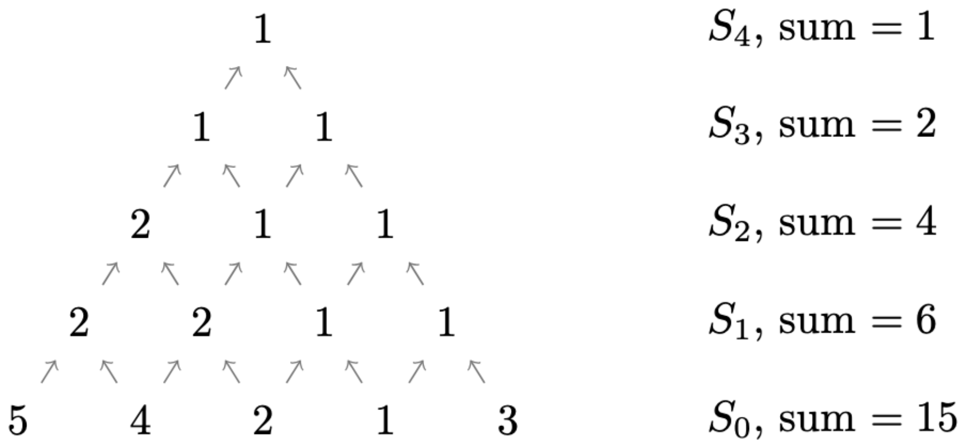


Figure: The transformation process. Each element in S_{i+1} is the LCA of two adjacent elements in S_i.

Problem H. Prime Triangles

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

You are given an integer n. Construct exactly n triangles such that the following conditions are satisfied:

- The vertices of the triangles are chosen from a set of at most $\lceil \frac{n}{4} \rceil + 6$ distinct lattice* points.
- All coordinates of the chosen points are in the range $[-10^7, 10^7]$.
- Each triangle has area equal to a prime† number.
- The areas of the n triangles are pairwise distinct.

* A lattice point is a point with integer coordinates.
† A prime number is a natural number greater than 1 with exactly two positive divisors: 1 and itself. For example, 7 and 11 are prime numbers, but 1, 6, and 12 are not.

Input

The first line contains a single integer t $(1 \le t \le 100)$ — the number of test cases.

Each test case contains a single integer n $(1 \le n \le 10^5)$ — the number of triangles to construct.

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

Output

For each test case, output a valid construction.

First, print a single integer k $(3 \le k \le \lceil \frac{n}{4} \rceil + 6)$ — the number of distinct lattice points.

Then print k lines, each containing two integers x_i, y_i $(-10^7 \le x_i, y_i \le 10^7)$ — the coordinates of the i-th lattice point. The points must be distinct.

Then print n lines. The i-th line must contain three distinct integers a_i, b_i, c_i $(1 \le a_i, b_i, c_i \le k)$ — the indices of the three lattice points forming the i-th triangle.

If there are multiple solutions, output any. It can be shown that a solution always exists under the given constraints.

Example

standard input	standard output
1 3	7 9 7 1 1 6 0 3 3 8 8 5 3 8 2 1 2 3 4 5 6 7 1 5

Note

In the first test case, $n = 3$ and the maximum allowed number of lattice points is $\lceil \frac{3}{4} \rceil + 6 = 7$. The following is a valid construction with 7 points:

- Triangle 1: points $(9, 7)$, $(1, 1)$, $(6, 0)$ with area 19.
- Triangle 2: points $(3, 3)$, $(8, 8)$, $(5, 3)$ with area 5.
- Triangle 3: points $(8, 2)$, $(9, 7)$, $(8, 8)$ with area 3.

All three areas (19, 5, 3) are prime and pairwise distinct.

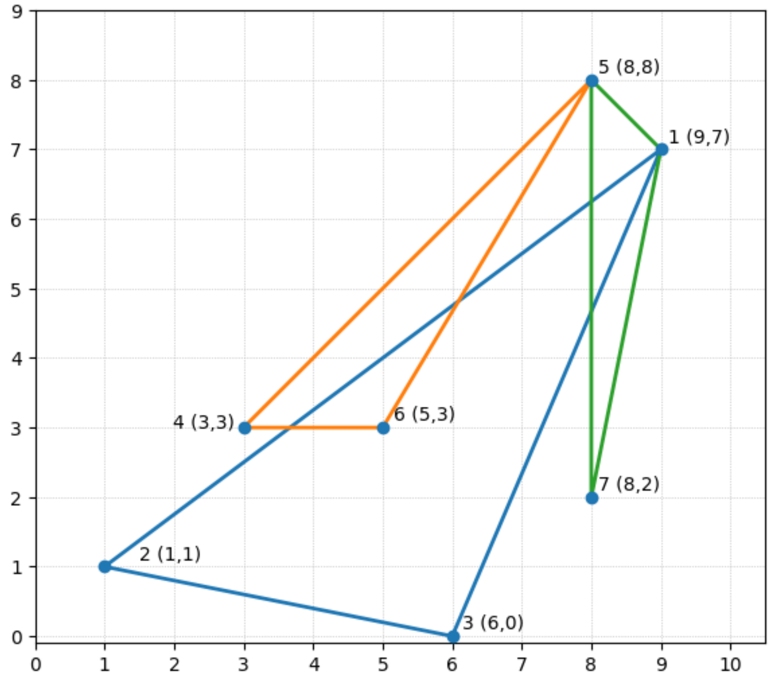


Figure: Prime-area triangle construction for $n = 3$.

Problem I. Peak Reduction

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

You are given a permutation* p of length n. In one operation, you may remove an element p_i if the following conditions are satisfied:

- $1 < i < |p|$ (where $|p|$ denotes the current length of p),
- $p_{i-1} < p_i > p_{i+1}$ (i.e., p_i is a local maximum).

Determine whether it is possible to reduce the permutation length to exactly 2 by applying the operation any number of times (possibly zero).

* A permutation of length n is a sequence of n integers where each integer from 1 to n appears exactly once. For example, $[1, 3, 2]$ is a permutation, but $[2, 3, 2]$ and $[4, 1, 2]$ are not.

Input

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

The first line of each test case contains an integer n $(3 \le n \le 3 \cdot 10^5)$ — the length of the permutation.

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

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 `YES` if it is possible to reduce the permutation to length 2, otherwise print `NO`.

Example

standard input	standard output
2 3 1 3 2 4 1 2 3 4	YES NO

Note

In the first test case, $p = [1, 3, 2]$. Since $p_1 < p_2 > p_3$, we can remove $p_2 = 3$. The resulting array $[1, 2]$ has length 2.

In the second test case, no element satisfies the local maximum condition, so it is impossible to reduce the length.

Problem J. The Power of the Sun

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Dr. Otto Octavius is a brilliant nuclear physicist working on his life's work: a sustainable fusion power source. He claims to have "the power of the sun, in the palm of his hand" with his new *Infinite Factorial Reactor*.

The reactor consists of n fusion cores lined up in a row, indexed from 1 to n. Initially, the i-th core holds an energy level of a_i.

Octavius uses his mechanical arms to conduct q operations. In each operation, he interacts with a specific range of cores $[l, r]$. The operations are of two types:

1. *Fusion Ignition:* Octavius triggers a reaction in the cores from index l to r. The energy level of every core in this range transforms into its own factorial. Formally, for each i such that $l \le i \le r$, the energy level a_i becomes $a_i!$ (where $0! = 1$ and $x! = 1 \times 2 \times \cdots \times x$ for $x \ge 1$).
2. *Stability Check:* Octavius measures the total energy output of the cores from index l to r. However, the inhibitor chip has a limit and can only display the result modulo 998244353. Formally, the reading is $\left(\sum_{i=l}^{r} a_i\right) \pmod{998244353}$.

Help Dr. Octavius simulate the reactor's behavior and determine the energy readings for all type 2 operations.

Input

The first line contains a single integer t ($1 \le t \le 10^5$) — the number of test cases.

The first line of each test case contains two integers n and q ($1 \le n, q \le 5 \cdot 10^5$) — the number of fusion cores and the number of operations.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 5 \cdot 10^5$) — the initial energy levels of the cores.

Each of the next q lines contains three integers *type*, l, r ($type \in \{1, 2\}$, $1 \le l \le r \le n$) — describing an operation of the corresponding type on the range $[l, r]$.

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

Output

For each operation of type 2, print a single integer — the sum of energy levels in the range modulo 998244353.

Example

standard input	standard output
1 5 4 3 2 4 1 0 2 1 5 1 1 3 2 1 5 2 2 4	10 33 27

Problem K. The Great Withering

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	512 megabytes

You are given a tree with n nodes.

Consider assigning an integer weight from 1 to m to each of the $n-1$ edges. Different edges may have the same weight.

You will remove all n nodes one by one. At step i ($1 \le i \le n$):

- Pick any remaining node u. Let s_i be the sum of weights of all edges currently connected to u ($s_i = 0$ if no edges are connected).
- Remove u along with all edges currently connected to it.

A weight assignment is valid if there exists a removal order such that $s_1 > s_2 > \cdots > s_n$. Let $f(m)$ denote the number of valid assignments among all m^{n-1} possible ways of assigning weights from 1 to m.

Given an integer k, compute $\sum_{m=1}^{k} f(m)$ modulo 998244353.

Input

The first line contains an integer t ($1 \le t \le 10^5$) — the number of test cases.

The first line of each test case contains two integers n and k ($2 \le n \le 10^6$, $1 \le k \le 10^9$).

Each of the next $n-1$ lines contains two integers u and v ($1 \le u, v \le n$) indicating there is an edge between nodes u and v. It is guaranteed that the given edges form a tree.

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

Output

For each test case, output a single integer — $\sum_{m=1}^{k} f(m)$ modulo 998244353.

Example

standard input	standard output
2 3 3 1 2 2 3 4 7 1 2 2 3 1 4	8 280

Note

In the first test case, the tree is a path: $1-2-3$. Let $[w_1, w_2]$ denote the weights of edges $(1,2)$ and $(2,3)$ respectively.

- $f(1) = 0$: No valid assignments.
- $f(2) = 2$: Valid assignments are $[1,2]$ and $[2,1]$.
- $f(3) = 6$: Valid assignments are $[1,2]$, $[1,3]$, $[2,1]$, $[2,3]$, $[3,1]$, and $[3,2]$.

So the answer is $0+2+6=8$.