The 4th Universal Cup

Stage 12: Grand Prix of Shanghai

January 10-11, 2026

This problem set should contain 13 problems on 21 numbered pages.

Based on

International Collegiate Programming Contest (ICPC)

Problem A. Menji, we miss you!

Time limit:	1 second
Memory limit:	1024 megabytes

This is an interactive problem.

Menji is lost and everyone misses him. It's your job to find him!

There is a **binary** tree T, consisting of n vertices, and each edge of tree T is of length 1. Menji is hiding at vertex X. You know the structure of the tree. However, you don't know X.

To find X, you can send signals. You can select a vertex u and select a signal strength k, then send a signal of strength k from vertex u. If the distance between u and X is no more than k, Menji will receive the signal and send a signal back, and you will receive the signal. Otherwise, you won't receive anything.

Sending signals is slow, and you are in a hurry, so please determine the position of Menji in no more than 40 signals., so please determine the position of Menji in no more than 40 signals.

Interaction Protocol

The input contains multiple testcases. The first line of the input contains an integer T ($1 \le T \le 100$), the number of testcases.

For each testcase, the first line contains an integer n ($2 \le n \le 3 \times 10^4$), the number of vertices in the tree.

The second line contains $n-1$ integers $fa_2, fa_3, \cdots, fa_n$ ($1 \le fa_i < i$), where fa_i is the parent of i on the tree. The tree is rooted at vertex 1.

It is guaranteed that the tree is a binary tree, that is, there doesn't exist $1 < i < j < k \le n$**, such that** $fa_i = fa_j = fa_k$.

To send a signal, print a single line in the following format:

- `? u k`: Indicate that you create a signal at vertex u with strength k. You need to ensure $1 \le u \le n, 0 \le k \le n$. Then you have to read an integer o ($o \in \{0, 1\}$). If you received the signal, or equivalently, $dis(u, X) \le k$, then $o = 1$, otherwise $o = 0$.

To report the answer, print a single line in the following format:

- `! u`: Indicate that you have found $X = u$. You need to move on to the next testcase after printing this, or terminate if there's no more.

For each testcase, you can send at most 40 signals. Reporting the answer does not count as sending a signal.

If you send more than 40 signals, or the signal you sent is malformed, or the answer you reported is incorrect, then the interaction will end and you will receive `Wrong answer` verdict.

Note that the interactor is **adaptive**, meaning that the answer may change depending on your queries as long as it remains consistent with the constraints and the answers to the previous queries.

It's guaranteed that the sum of n over all testcases does not exceed 3×10^4.

After printing each line do not forget to output the end of line and flush the output. You may use `fflush(stdout)` or `cout.flush()` to flush the stream for C++, use `System.out.flush()` for Java and `stdout.flush()` for Python.

Example

standard input	standard output

2
7
1 1 2 2 3 3
 	
1
 	
0
 	
0
 	
7
1 1 2 2 3 3
 	
0
 	
1
	? 2 1
	
	? 3 2
	
	? 4 0
	
	! 5
	
	? 2 1
	
	? 3 0
	
	! 3

Problem B. Hamu

Time limit:	1 second
Memory limit:	1024 megabytes

Dingdong plans to travel to the country of Hamu.

The country of Hamu consists of n cities and m bidirectional roads. Before this trip, Dingdong has already visited city i exactly a_i times. During this trip, Dingdong plans to enter Hamu at city s. On each subsequent day, Dingdong will travel along a certain road to a city and visit that city once. At the end of the final day's visit, Dingdong should be at city s and leave Hamu from city s. **Note that on the day of entering the country of Hamu, Dingdong does not visit city s.**

Dingdong hopes that after this trip, combined with his previous visits, every city in Hamu will have been visited by him an even number of times. Dingdong has limited time and can visit at most $5n$ cities during this trip. Please help him construct a valid trip plan, or tell him if it is impossible.

Input

The input contains multiple testcases. The first line of the input contains an integer T ($1 \le T \le 2 \times 10^5$), the number of testcases.

For each testcase, the first line contains three integers n, m, s ($1 \le n \le 2\times10^5, 0 \le m \le 2\times10^5, 1 \le s \le n$), where n is the number of cities, m is the number of roads, s is the index of the starting city.

The next line contains n integers $a_1, a_2, \cdots, a_n$ ($0 \le a_i \le 10^9$), the number of times Dingdong has visited for each city.

The next m lines each contain two integers u_i, v_i ($1 \le u_i, v_i \le n$), indicating an undirected road connecting city u_i and city v_i. **There can be multiple edges and self loops.** In other words, it's **not** guaranteed that $u_i \ne v_i$, and it's **not** guaranteed that $(u_i, v_i) \ne (u_j, v_j)$ for $i \ne j$.

It's guaranteed that the sum of n and the sum of m over all testcases does not exceed 2×10^5, respectively.

Output

For each testcase, if there's no valid trip plan, print `No` in a single line.

Otherwise, print `Yes` in a single line first, then print an integer k ($0 \le k \le 5n$) in the next line, representing the total number of visits during this trip. In the following line, print the indexes of the cities visited in order.

Example

standard input	standard output
5	Yes
4 4 1	4
1 1 1 1	2 3 4 1
1 2	Yes
2 3	6
3 4	2 4 5 2 3 1
4 1	Yes
5 7 1	0
9 4 3 11 7	
1 2	No
2 5	Yes
2 4	2
3 4	2 1
2 3	
1 3	
4 5	
2 2 2	
114 514	
1 2	
1 2	
5 0 1	
114 514 19 19 810	
2 1 1	
3 5	
1 2	

Note

For the first testcase, after visiting along the route $1 \to 2 \to 3 \to 4 \to 1$, cities $1, 2, 3, 4$ have been visited $2, 2, 2, 2$ times respectively, meeting the requirement.

Problem C. Singularity

Time limit: 1 second
Memory limit: 1024 megabytes

In the year of 2077, problemsetting becomes simple. Robots will generate a problem by setting some random operations and then solve it. A problemsetter only has to check whether the problem is correct or not.

Here is a problem from 2077:

Given a permutation $p_1, p_2, \cdots p_n$, it is guaranteed that n is **even**. You wish to sort the permutation, using only one type of operation:

- `FakeSort(l,r)`: You have to guarantee that $r - l + 1$ is even. Let $k = r - l + 1$, then the largest $k/2$ elements and the smallest $k/2$ elements in the continuous subsequence $p_l, p_{l+1}, \cdots p_r$ will be sorted independently. That is, let $L_1, L_2, \cdots L_{k/2}$ be the indices of the largest $k/2$ numbers, and $S_1, S_2 \cdots S_{k/2}$ be the indices of the smallest $k/2$ numbers. We first sort the numbers on the indices $L_1 \sim L_{k/2}$, then sort the numbers on the indices $S_1 \sim S_{k/2}$.

Here is a concrete example; suppose the permutation is $p = \{2, 5, 7, 1, 8, 6, 4, 3\}$. If we call `FakeSort(2,7)`:

- $k = r - l + 1 = 6$. The continuous subsequence $p_l, p_{l+1}, \cdots, p_r$ is $\{5, 7, 1, 8, 6, 4\}$; we will sort the largest 3 numbers and smallest 3 numbers of this sequence independently.
- $p = \{2, 5, \mathbf{7}, 1, \mathbf{8}, \mathbf{6}, 4, 3\}$; the largest 3 numbers are bold. After sorting, they become $p = \{2, 5, \mathbf{6}, 1, \mathbf{7}, \mathbf{8}, 4, 3\}$.
- $p = \{2, \mathbf{5}, 6, \mathbf{1}, 7, 8, \mathbf{4}, 3\}$; the smallest 3 numbers are bold. After sorting, they become $p = \{2, \mathbf{1}, 6, \mathbf{4}, 7, 8, \mathbf{5}, 3\}$.

So $p = \{2, 5, 7, 1, 8, 6, 4, 3\}$ after `FakeSort(2,7)` becomes $\{2, 1, 6, 4, 7, 8, 5, 3\}$.

Please use no more than 114 operations to sort the permutation or determine it is impossible. It can be proved that if a permutation can be sorted with this operation, there is a way to use no more than 114 operations.

Input

The input contains multiple testcases. The first line of the input contains an integer T ($1 \le T \le 10^3$), the number of testcases.

For each testcase, the first line contains an **even** integer n ($4 \le n \le 10^5$), the length of the permutation.

The second line contains n integers $p_1, p_2, \cdots, p_n$ ($1 \le p_i \le n$), the permutation you need to sort. It is guaranteed that p is a permutation.

It's guaranteed that the sum of n over all testcases does not exceed 2×10^5.

Output

For each testcase, if it is impossible to sort the permutation, print -1.

Otherwise, print an integer k ($0 \le k \le 114$), denoting the number of operations used.

In the following k lines, print l, r ($1 \le l \le r \le n, r - l + 1$ is even) in each line, denoting performing `Fakesort(l,r)`.

Example

standard input	standard output

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

Problem D. Not a subset sum

Time limit:	1 second
Memory limit:	1024 megabytes

We are given an array $a_0, a_1, ..., a_{2^n-1}$ of length 2^n. For a string q of length n over the alphabet `0,1,?` (1-indexed), define the "generalized subset sum"$S(q)$ as the sum of a_j over all indices j that satisfy:

- For each $1 \le k \le n$, if $q_k = 0$ then the k-th bit of j is 0.
- For each $1 \le k \le n$, if $q_k = 1$ then the k-th bit of j is 1.
- If $q_k =?$ there is no restriction on the k-th bit of j.

For example, when $n = 3$ and $s =$ `0?1`, the generalized subset sum is $S(q) = a_4 + a_6$ (binary 100 and 110). **Note that the first bit is the lowest bit.**

Your task is to compute the generalized subset sum for each string of length n over the alphabet `0,1,?`. Because the total output can be large, you only need to output the **bitwise XOR** of all these sums.

Input

The first line of the input contains an integer n ($1 \le n \le 16$).

The second line of the input contains 2^n integers $a_0, a_1, \cdots, a_{2^n-1}$ ($0 \le a_i \le 9$), contents in array a.

Output

Print an integer denoting the bitwise-XOR of all generalized subset sums.

Examples

standard input	standard output
1 3 5	14
2 2 6 8 8	0

Note

The following are query strings q and their corresponding generalized subset sums $S(q)$ in **sample 2**:

$q =$ `00`$, S(q) = a_0 = 2$

$q =$ `10`$, S(q) = a_1 = 6$

$q =$ `?0`$, S(q) = a_0 + a_1 = 2 + 6 = 8$

$q =$ `01`$, S(q) = a_2 = 8$

$q =$ `11`$, S(q) = a_3 = 8$

$q =$ `?1`$, S(q) = a_2 + a_3 = 16$

$q =$ `0?`$, S(q) = a_0 + a_2 = 10$

$q =$ `1?`$, S(q) = a_1 + a_3 = 14$

$q =$ `??`$, S(q) = a_0 + a_1 + a_2 + a_3 = 24$

Now it is easy to verify that the output is 0.

Problem E. Flower's land 3

Time limit:	1 second
Memory limit:	1024 megabytes

There is a binary string s_1 of length m stored on disk #1.

Then, there are $n-1$ operations. In the i-th operation, a disk $1 \le p_{i+1} \le i$ is chosen, and the string $s_{p_{i+1}}$ on that disk is copied to disk $i+1$, producing s_{i+1}. However, during the copying process, up to k bits may be flipped (i.e., errors occur in at most k positions).

You are given all the final n binary strings $s_1, s_2, \dots, s_n$, each of length m. Your task is to determine how many possible sequences $p_2, p_3, \dots, p_n$ could result in this final configuration.

Since the answer can be large, you only need to find the answer modulo 998244353.

Input

The first line of the input contains three integers n, m, k ($2 \le n \le 5000, 4 \le m \le 15000, 1 \le k \le 3$), described in the statement. It is guaranteed that m is a multiple of 4.

Each of the next n lines contains a hexadecimal string s'_i of length $m/4$. Each character of s'_i is one of `0-9` or `A-F`, where `A` $= 10$, `B` $= 11$, …, `F` $= 15$.

Each bit in the hexadecimal representation s'_i corresponds to four consecutive bits in the binary string s_i. Specifically, for each bit $s'_{i,j}$, it can be proved that there exists a unique tuple (a, b, c, d) satisfying $s'_{i,j} = 8a + 4b + 2c + d$ and $a, b, c, d \in \{0, 1\}$. Bits in the binary string s_i satisfy $(s_{i,4j}, s_{i,4j+1}, s_{i,4j+2}, s_{i,4j+3}) = (a, b, c, d)$.

Output

Print an integer — the number of valid sequences $(p_2, p_3, \dots, p_n)$ modulo 998244353.

Example

standard input	standard output
5 8 2 95 05 BD 9C BD	6

Problem F. Flower's land 4

Time limit:	1.5 seconds
Memory limit:	2048 megabytes

There are n segments on a 2D plane. Each segment starts on the **non-negative part of the x-axis** and ends on the **non-negative part of the y-axis**. In other words, its starting point has coordinates $(x_i, 0)$ with $x_i \geq 0$, and its ending point has coordinates $(0, y_i)$ with $y_i \geq 0$.

You are given q queries. In each query, a segment is specified whose starting point lies on the x-axis, and whose ending point can be **anywhere in the first quadrant or the non-negative parts of the axes** of the plane. For each query segment, determine whether it intersects with any of the existing segments. **Intersections at endpoints are counted.**

Queries are independent of each other; that is, the segment given in each query will not be kept in the remaining queries.

Input

The input contains multiple testcases. The first line of the input contains an integer T ($1 \leq T \leq 10^6$), the number of testcases.

For each test case, the first line contains two integers n, q ($1 \leq n, q \leq 10^6$), the number of existing segments and the number of queries.

Each of the next n lines contains two integers x_i, y_i ($0 \leq x_i, y_i \leq 10^9$), describing a segment that starts at $(x_i, 0)$ and ends at $(0, y_i)$.

Then, each of the following q lines contains three integers a_j, b_j, c_j ($0 \leq a_j, b_j, c_j \leq 10^9$), describing a query segment that starts at $(a_j, 0)$ and ends at (b_j, c_j).

It's guaranteed that the sum of n and the sum of q over all testcases does not exceed 10^6, respectively.

Output

For each query, print `YES` if the query segment intersects (including at endpoints) with at least one of the existing segments, and `NO` otherwise.

Example

standard input	standard output

1
3 5
6 6
2 6
6 2
10 4 4
10 3 3
0 1 1
0 2 2
5 2 1
	NO
	YES
	NO
	YES
	NO

Problem G. Gemcrate

Time limit:	1 second
Memory limit:	512 megabytes

Noir has n gems. The i-th gem has a positive integer a_i written on it. Noir wants to divide these gems into several non-empty groups. Each gem belongs to exactly one group.

Suppose the i-th group contains gems with label $k_{i,1}, k_{i,2}, \ldots, k_{i,p}$, then Noir treats the *brightness* of the i-th group as $a_{k_{i,1}} \oplus a_{k_{i,2}} \oplus \cdots \oplus a_{k_{i,p}}$, where $\oplus$ is the bitwise-XOR operation. Denote the *brightness* of the i-th group as B_i.

For a grouping method of m groups, Noir treats the *value* of this method as $B_1 \& B_2 \& \ldots \& B_m$, where $\&$ is the bitwise-AND operation.

Noir wants to find the maximum possible *value* over all grouping methods.

Input

The input contains multiple testcases. The first line of the input contains an integer T ($1 \le T \le 10^4$), the number of testcases.

For each testcase, the first line contains an integer n ($1 \le n \le 5 \times 10^5$), the number of gems.

The second line contains n integers $a_1, a_2, \cdots, a_n$ ($1 \le a_i < 2^{60}$), the integers written on gems.

It's guaranteed that the sum of n over all testcases does not exceed 5×10^5.

Output

For each testcase, print an integer representing the maximum possible *value* over all grouping methods.

Example

standard input	standard output

4
4
1 2 3 1
6
4 7 5 2 6 3
4
14 15 9 18
2
251508091405 13011908091815
	2
	6
	26
	13121614001578

Note

For the first testcase, a possible grouping method is $[1, 2, 3, 1] = [1, 3], [2, 1]$ with a value of $B_1 \& B_2 = (1 \oplus 3) \& (2 \oplus 1) = 2 \& 3 = 2$. Another possible grouping method is $[1, 2, 3, 1] = [1, 2, 3, 1]$, with a lower value $B_1 = 1 \oplus 2 \oplus 3 \oplus 1 = 1$. It can be proved that it's not possible to achieve a value greater than 2.

For the second testcase, the best grouping method is $[4, 7, 5, 2, 6, 3] = [7], [5, 3], [6], [4, 2]$ with a value of $7 \& (5 \oplus 3) \& 6 \& (4 \oplus 2) = 6$.

Problem H. AGI

Time limit:	1 second
Memory limit:	1024 megabytes

Dr. Menji is an expert in AI. Now, he is training Bot, an AGI(Artificial Game Intelligence).

To teach Bot how to play games, he plays with Bot every day. Today they are playing the following game:

The game starts with a sequence of $2n$ non-negative integers, $a_1, a_2, \cdots a_{2n}$, and a number S. Initially, $S = 0$.

Menji and Bot take turns; Menji goes first:

- In Menji's turn, he chooses a number x from the sequence, sets $S \leftarrow S \oplus x$, and deletes x from the sequence. Note that $\oplus$ is the bitwise XOR(exclusive or) function.
- In Bot's turn, he chooses a number x from the sequence and deletes x from the sequence.

The game ends when no numbers remain in the sequence. Menji wins if and only if $S = 0$ in the end; otherwise, Bot wins.

Menji wonders if both players play optimally, who is the winner of the game.

Input

The input contains multiple testcases. The first line contains an integer T ($1 \le T \le 10^5$), the number of testcases.

For each testcase, the first line contains an integer n ($1 \le n \le 10^5$), described in the statement.

The second line contains $2n$ integers $a_1, a_2, \cdots, a_{2n}$ ($0 \le a_i < 2^{30}$), representing integers in the game.

It's guaranteed that the sum of n over all testcases does not exceed 2×10^5.

Output

For each testcase, if Menji can win the game, print `Menji` in one line; otherwise, print `Bot` in one line.

Example

standard input	standard output

5
2
1 1 3 3
2
1 1 1 3
3
1 1 4 5 1 4
3
1 9 1 9 8 10
6
1 1 4 5 1 4 1 9 1 9 8 10
	Bot
	Menji
	Menji
	Menji
	Bot

Note

For the first testcase, no matter what number Menji chooses, Bot can always choose a same number, so Menji always chooses an 1 and a 3, $S = 1 \oplus 3 = 2 \neq 0$, so Bot can always win.

For the second testcase, Menji can choose an 1 in the first turn; no matter what Bot chooses, Menji can choose another 1, so Menji always receives two 1s, $S = 1 \oplus 1 = 0$, so Menji can always win.

Problem I. Round screws

Time limit:	1 second
Memory limit:	1024 megabytes

The NIT likes round screws. He also likes the $\oplus$ operator because it reminds him of round screws, where $\oplus$ represents Bitwise-XOR operation.

Define the value V_a of a sequence $a_1, a_2, \cdots, a_n$ as $V_a = a_1 + a_n + \sum_{i=1}^{n-1}(a_i \oplus a_{i+1})$.

Given a sequence $a_1, a_2, \cdots, a_n$, you can perform the following operation for arbitrary times:

- Select an index i $(1 \le i \le n)$, change a_i to any non-negative integer; this operation has a cost C.

Minimize the sum of the value of the sequence and the cost incurred by operations. In other words, let p be the number of operations you performed, and $V_{a'}$ be the value of the a after the operations, then you need to minimize $pC + V_{a'}$.

Input

The input contains multiple testcases. The first line of the input contains an integer T $(1 \le T \le 100)$, the number of testcases.

For each testcase, the first line contains two integers n, C $(2 \le n \le 10^5, 0 \le C < 2^{19})$, the length of the sequence and the cost of performing an operation.

The second line contains n integers $a_1, a_2, \cdots, a_n$ $(0 \le a_i < 2^{18})$, representing the elements in the sequence.

It's guaranteed that the sum of n over all testcases does not exceed 2×10^5.

Output

For each testcase, print an integer in one line, the minimum possible value of $pC + V_{a'}$.

Example

standard input	standard output

3
4 4
1 4 5 6
8 6
6 6 6 1 1 6 6 6
6 7
1 7 2 6 3 5
	14
	24
	29

Note

For the first testcase, one way to achieve minimum is to change the sequence to $1, \mathbf{1}, \mathbf{1}, \mathbf{0}$; the bold numbers are the changed. The final sequence value is 2, and 3 operations are done; the total value + cost is $2 + 3 \times 4 = 14$.

For the second testcase, one way to achieve minimum is to change the sequence to $6, 6, 6, \mathbf{6}, \mathbf{6}, 6, 6, 6$; the final value is $12 + 2 \times 6 = 24$.

Problem J. Yet another mailbox problem

Time limit:	1.5 seconds
Memory limit:	512 megabytes

Yana, Mino, White, and Huzz are best friends.

One day, the coach asked White, Mino, and Huzz to prepare a mock contest. Huzz designed a beautifully search problem whose complexity is exponential with respect to k. He carefully checked the entire problem statement, except for one detail: he accidentally wrote $k \le 5 \times 10^5$ instead of $k \le 10$. Later, this problem appeared in a mock contest. The furious contestant, Yana, came to Huzz, asking how the problem was supposed to be solved. But he seemed to have forgotten something, and the problem statement he provided looked slightly different —

You are given a **directed** graph with n vertices and m edges, where each edge e is assigned an integer weight $w(e)$ between 1 and 8.

A path is a sequence of edges $(e_1, e_2, \ldots, e_\ell)$ such that the endpoint of e_i is the startpoint of e_{i+1} for all $1 \le i < \ell$. The length of the path is ℓ, the number of edges it contains. Note that a path may contain the same edge **multiple times**.

The weight sequence of the path is the sequence of edge weights $[w(e_1), w(e_2), \ldots, w(e_\ell)]$. Paths are compared by **lexicographical order** of their weight sequences.

Two paths are considered distinct as long as they use different edges, even if they share the same vertex sequence and weight sequence. For example, if both paths (e_1, e_2) and (e_3, e_4) have weight sequence `[1,2]`, and both traverse vertices $1 \to 2 \to 3$, they are still distinct as long as $e_1 \ne e_3$ or $e_2 \ne e_4$.

White wants to find the lexicographically smallest k paths. Since the total output may be too large, you only need to output the length of each path.

Input

The first line of the input contains three integers n, m, k ($2 \le n \le 5{\times}10^5, 1 \le m \le 5{\times}10^5, 1 \le k \le 5{\times}10^5$), representing the number of vertices, the number of edges, and the required number of paths.

Each of the next m lines contains three integers x, y, z ($1 \le x, y \le n, 1 \le z \le 8, x \ne y$), representing a directed edge $e = (x, y)$ with weight $w(e) = z$. The given edge set may contain **multiple edges**.

Output

Print k lines. The i-th line should contain a single integer, the length of the path whose weight is the i-th smallest in lexicographical order. If there are fewer than i paths, output -1 instead.

Examples

standard input	standard output

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

`3 4 10`
`1 2 1`
`1 2 1`
`2 3 2`
`2 3 3`
	`1`
	`1`
	`2`
	`2`
	`2`
	`2`
	`1`
	`1`
	`-1`
	`-1`

`6 5 15`
`1 2 3`
`2 3 5`
`3 4 2`
`3 5 1`
`5 6 4`
	`1`
	`2`
	`1`
	`1`
	`2`
	`3`
	`4`
	`3`
	`1`
	`1`
	`2`
	`3`
	`2`
	`-1`
	`-1`

Note

For simplicity, let e_j denote the j-th input edge.

For the first testcase, the lexicographically smallest 8 paths are:

- Path (e_1), weight sequence `[1]`.
- Path (e_3), weight sequence `[1]`.
- Path (e_5), weight sequence `[1]`.
- Path (e_5, e_1), weight sequence `[1,1]`.
- Path (e_5, e_1, e_4), weight sequence `[1,1,2]`.
- Path (e_5, e_1, e_4, e_5), weight sequence `[1,1,2,1]`.
- Path $(e_5, e_1, e_4, e_5, e_1)$, weight sequence `[1,1,2,1,1]`.

- Path $(e_5, e_1, e_4, e_5, e_1, e_4)$, weight sequence `[1,1,2,1,1,2]`.

Problem K. No more regrets

Time limit:	4 seconds
Memory limit:	1024 megabytes

After the provincial team selection, White fell into a long period of disappointment. She could not find any hope in her results. Even so, White continued her final training before the NOI. Besides her regular training, she also began to explore topics she had never had the chance to learn during her OI days — hoping that, before saying goodbye, there would be no more regrets.

Shaking off her wandering thoughts, White suddenly focused on a problem in front of her, a plain and boring data structure problem —

White has a sequence of n integers $a_1, a_2, \ldots, a_n$. She will perform q operations on this sequence. Each operation is one of the following three types:

- `1 l r v` — Add v to each element in the interval $[l, r]$.
- `2 l r v` — Assign each element in the interval $[l, r]$ to v.
- `3 l r` — Query the value $\sum_{i=l}^{r}(\min_{j=l}^{i} a_j) \times (\max_{j=l}^{i} a_j)$ modulo 2^{64}.

Your task is to simulate all operations and output the results of all queries.

Input

The first line of the input contains two integers n, q ($1 \le n, q \le 2 \times 10^5$), representing the number of elements and the number of operations.

The second line contains n integers $a_1, a_2, \cdots, a_n$ ($0 \le a_i \le 10^9$), representing the initial elements.

Each of the next q lines describes an operation, in one of the three formats:

- `1 l r v` ($1 \le l \le r \le n, -10^9 \le v \le 10^9$), representing the Add operation.
- `2 l r v` ($1 \le l \le r \le n, 1 \le v \le 10^9$), representing the Assign operation.
- `3 l r` ($1 \le l \le r \le n$), representing the Query operation.

It's guaranteed that $0 \le a_i \le 10^9$ for each $1 \le i \le n$ during the whole process.

Output

For each Query operation, print an integer in a single line—the result of the query modulo 2^{64}.

Examples

standard input	standard output

5 8
2 3 5 4 1
3 1 5
3 2 4
2 4 5 2
3 1 5
3 2 4
1 1 2 5
3 1 5
3 2 4
	35
	39
	40
	34
	177
	120

10 20
1 2 3 4 5 6 7 8 9 10
1 1 10 1
3 1 5
3 2 9
3 8 10
1 2 5 10
3 1 5
3 2 9
3 8 10
1 5 9 -5
3 1 5
3 2 9
3 8 10
1 2 5 -10
3 1 5
3 2 9
3 8 10
1 5 9 5
3 1 5
3 2 9
3 8 10
	40
	156
	270
	120
	1202
	270
	118
	831
	80
	33
	61
	80
	40
	156
	270

Note

In the first testcase:

The original sequence is `2 3 5 4 1`. After the third operation, it becomes `2 3 2 2 1`, and after the sixth operation, it becomes `7 8 7 7 6`.

For the first query, the answer is $\sum_{i=1}^{5}(\min_{j=1}^{i} a_j) \times (\max_{j=1}^{i} a_j) = 2\times 2+2\times 3+2\times 5+2\times 5+1\times 5 = 35$.

For the second query, the answer is $\sum_{i=2}^{4}(\min_{j=2}^{i} a_j) \times (\max_{j=2}^{i} a_j) = 3\times 3+3\times 5+3\times 5 = 39$.

Problem L. Yet another permutation problem

Time limit:	1.5 seconds
Memory limit:	1024 megabytes

Yana, Mino, White, and Huzz are best friends.

Having finally finished the coach's demanding tasks, Huzz had earned a rest. He found it on one lazy afternoon, where the world seemed to slow to a crawl, wrapped in the warm, golden haze of impending dusk. A gentle breeze did little more than stir the dust motes dancing in the sunbeams slanting through the leaves of a great oak tree.

In that sleepy, comfortable void, Huzz found a permutation in his soft shark toy. He decided to share it with his friends for fun.

Mino loves splitting. He can split this permutation into several contiguous segments.

Yana loves swapping. He can choose one contiguous segment and swap its maximum and minimum elements.

Specifically, they can perform the following two types of operations **any number of times, in any order**:

- **split**: Choose one contiguous segment whose length is greater than 1. Then choose a position inside it and split it into two adjacent contiguous segments. For example, $(a_i, \ldots, a_j)$ can be split into $(a_i, \ldots, a_k)$ and $(a_{k+1}, \ldots, a_j)$, where $i \le k < j$.
- **swap**: Choose one contiguous segment and swap its maximum and minimum elements.

After performing any number of operations, they stop performing, and all resulting segments are merged in their original order to form a new permutation.

White loves counting. She wonders — how many distinct permutations can they obtain?

As the result can be very large, you only need to find the answer modulo 998 244 353.

Input

The first line the input contains an integer n ($1 \le n \le 500$), the length of the permutation.

The second line contains n integers $a_1, a_2, \cdots, a_n$ ($1 \le a_i \le n$), representing the permutation itself. It's guaranteed that $\{a_n\}$ is a permutation.

Output

Print an integer, the number of distinct permutations they can obtain modulo 998 244 353.

Examples

standard input	standard output
4 1 4 2 3	10
7 5 1 4 2 6 3 7	340

Note

In the first sample, the following are all possible permutations they can obtain:

(1234), (1243), (1423), (1432), (4123), (4132), (4213), (4231), (4312), (4321)

Problem M. Yet another 01 problem

Time limit:	5 seconds
Memory limit:	1024 megabytes

Yana, Mino, White, and Huzz are best friends.

Mino has been feeling confused lately. Obsessed with his past failures in OI, he has been struggling to plan his future while managing his busy school schedule. His three friends suggested he evaluate the importance of his tasks and prioritize them.

Assume there are n different tasks numbered from 1 to n. Each time, Huzz selects two adjacent tasks, White compares them, and Yana merges them into a single task. Mino is surprised to find that this process accidentally forms a segment tree, which does not necessarily split in the middle. More precisely, it forms a tree with $2n - 1$ nodes, where every subtree corresponds to a contiguous segment. Note that all n tasks are leaves, while the other $n - 1$ nodes each have two children. These nodes, which result from comparisons, define the values of the edges to their children: 0 for the smaller one and 1 for the larger one.

Mino is very fond of calculations. He defines the weight of each task as the XOR sum of the edges along the path from the task to the root. He wonders: if the weights are given, how many ways are there to construct such trees and compare the children?

Once again, Mino is not skilled in OI, which is why he has turned to you for help. To simplify the problem, you only need to find the answer modulo $998,244,353$.

Input

The first line of the input contains a positive integer n ($1 \le n \le 250000$), the number of tasks.

The second line contains a binary string S of length n, where S_i represents the Bitwise-XOR of all values on the edges from task i to the root.

Output

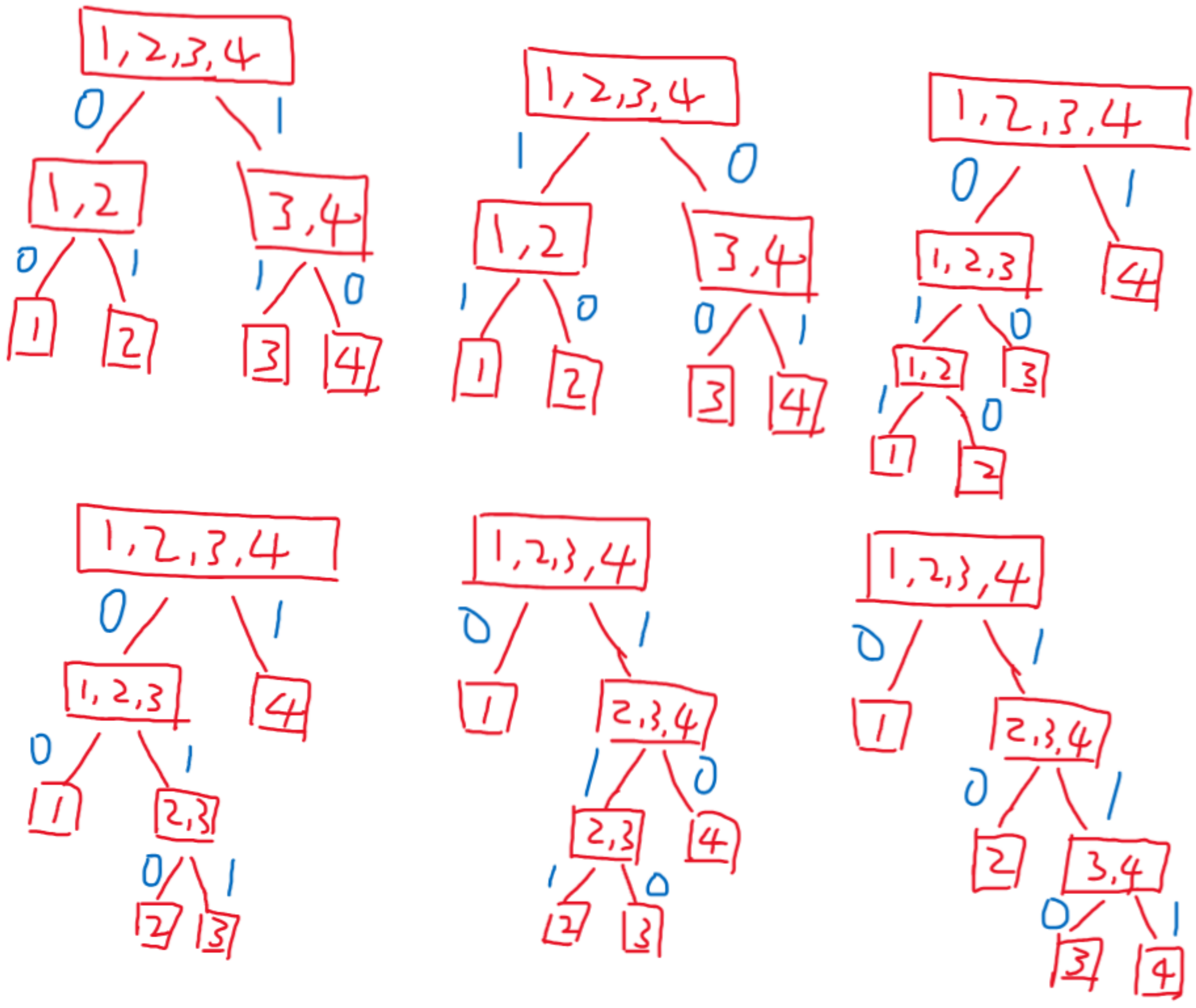
Print an integer, the answer modulo $998,244,353$.

Example

standard input	standard output
4 0101	6

Note

There are 6 ways to construct the tree and compare the children:


1,2,3,4
1,2
3,4
1
2
3
4
1,2,3
2,3
2,3,4
3,4
0
1