A. All-Star

Limits: 2 sec., 512 MiB

Somebody once told me...

Shrek has n outhouses in his swamp, indexed from 1 to n. These outhouses are connected by $n-1$ bidirectional roads, such that it is possible to get from any outhouse to any other by a series of roads.

In one move, Shrek may choose three distinct outhouses x, y, z such that there is a road from y to both x and z but no road from x to z, and replace the road connecting y to z with a road connecting x to z.

It is well-known that Shrek can only be described as an "all-star", so he wants his swamp to resemble a star. More precisely, he wants to make some number of moves such that there exists an outhouse that is connected to every other one by a road. Help him find a way to do so in the smallest number of moves.

Input

The first line contains a single integer n — the number of outhouses in Shrek's swamp.

The following $n-1$ lines each contain two integers u, v detailing that there exists a road connecting outhouses u and v.

Output

In the first line, print a single integer m — the smallest number of moves needed to turn Shrek's swamp into a star.

In the following m lines print three integers x, y and z; denoting that Shrek should preform the described move to outhouses x, y and z.

If several possible sequences of moves exist, you can print any of them.

Constraints

$3 \leq n \leq 10^3$.

It is guaranteed that there always exists a solution using at most 10^6 moves.

Samples

Input (*stdin*)	Output (*stdout*)
5 1 2 2 3 3 4 4 5	2 3 2 1 3 4 5

Notes

In the sample case, after the first move, outhouse 3 will be connected by road to outhouses 1, 2 and 4, and outhouse 5 will be connected by road to outhouse 4. After the second move, outhouse 3 will have roads connecting it to all other outhouses, and so the swamp will be a star. It is impossible to make the swamp a star after just one move, so the above answer is correct.

B. NonZero PrefSuf Sums

Limits: 2 sec., 512 MiB

Prince Charming wanted to give you a long, tedious legend full of pomp and flair. But Shrek won't allow this! He gives you a completely formal and short statement instead.

Count the number of arrays $[a_1, a_2, \ldots, a_n]$ of integers that satisfy the following conditions:

1. $|a_i| \leq m$ for all $1 \leq i \leq n$.
2. There exists a permutation $[b_1, b_2, \ldots, b_n]$ of elements of a, such that the following holds:
   - $b_1 + b_2 + \cdots + b_k \neq 0$ for all $1 \leq k \leq n$.
   - $b_k + b_{k+1} + \cdots + b_n \neq 0$ for all $1 \leq k \leq n$.

Output the answer modulo p, where p is a big prime number.

Input

The only line of the input contains three integers n, m, p.

Output

Output a single integer — the answer modulo p.

Constraints

$2 \leq n \leq 100$,
$1 \leq m \leq 100$,
$10^8 < p < 10^9$, p is prime.

Samples

Input (*stdin*)	Output (*stdout*)
`2 1 998244353`	`2`
`69 42 696969697`	`378553557`

Notes

In the **first** test case, there are 9 possible arrays: $[-1, -1]$, $[-1, 0]$, $[-1, 1]$, $[0, -1]$, $[0, 0]$, $[0, 1]$, $[1, -1]$, $[1, 0]$, $[1, 1]$. Only arrays $[-1, -1]$ and $[1, 1]$ satisfy the condition from the problem.

C. Duloc Network

Limits: 2 sec., 512 MiB

*This is an **interactive problem**, where your program and the judge interact via standard input and output.*

In the kingdom of Duloc, Lord Farquaad is developing a network of watchtowers to monitor every corner of his land. He has a map of towers and the roads that connect them, forming an undirected simple graph $G = (V, E)$, where each tower is a vertex and each road is an edge between two towers. However, Farquaad is worried that some parts of Duloc might be isolated, making it impossible to reach every tower from any other.

To ensure full connectivity, he tasks you with verifying whether his network is connected. However, there's a catch: you're only allowed limited access to information about the graph.

You can query the network to investigate its connectivity. A query allows you to select a subset of towers S and receive a count of the towers not in S that have direct roads connecting them to at least one tower in S. More precisely, $query(S) = |N(S) \setminus S|$, where $S \subseteq V$ and $N(S) = \{x | \exists y \in S$ such that $(x, y) \in E\}$.

Your goal is to use these queries efficiently to determine if the network is connected.

Can you help Lord Farquaad confirm the security of his kingdom by verifying that every tower is reachable from any other in Duloc's network?

Input

Interaction starts by reading an integer — the number of vertices.

Then you can make queries of the type `"? s"` (without quotes) where s is a binary string of length n such that character s_i is 1 if node $i \in S$ and 0 otherwise. After the query, read an integer — the answer to your query.

After printing a query do not forget to output end of line and flush the output. The interactor is non-adaptive. The graph does not change during the interaction.

Output

When you find if G is connected or disconnected, print it in the format `"! x"` (without quotes), where x is 1 if G is connected and 0 otherwise.

Constraints

$1 \leq |V| \leq 200$.

You are allowed to use at most **3500** queries.

Notes

In the following interaction, $|V| = 4$, $G = (V, E)$, $V = \{1, 2, 3, 4\}$, $E = \{(1, 2), (2, 3), (3, 4), (2, 4)\}$.

Input	Output	Description
4		$\|V\|$ is given.
	? 1100	Ask a query for subset $\{1, 2\}$.
2		The judge responds with 2.
	? 0010	Ask a query for subset $\{3\}$.
2		The judge responds with 2.
	? 1001	Ask a query for subset $\{1, 4\}$.
2		The judge responds with 2.
	! 1	The algorithm detected that G is connected.

Here is another example, $|V| = 2$, $G = (V, E)$, $V = \{1, 2\}$, $E = \emptyset$.

Input	Output	Description
2		$\|V\|$ is given.
	? 10	Ask a query for subset $\{1\}$.
0		The judge responds with 0.
	? 11	Ask a query for subset $\{1, 2\}$.
0		The judge responds with 0.
	! 0	The algorithm detected that G is disconnected.

D. Donkey and Puss in Boots

Limits: 2 sec., 512 MiB

In the land of Far Far Away, Shrek has n piles of candies. He likes eating candies so much. There are a_i candies in i-th pile.

Shrek is not the only one who likes eating candies. When he falls asleep, Donkey and Puss in Boots decide to eat his candies. Not only do they want to eat all the candies but also to play the game.

Puss in Boots being an honorable cat, lets Donkey go first. The game will be held by the following rules:

- On Donkey's turn, he must select one pile of candies and eat a positive number of candies from that pile.
- On Puss in Boots' turn, he must make n moves similar to those of Donkey (being much trickier).

The player who cannot make a move loses. You need to identify the winner.

Input

First line contains one integer n — number of piles of candies. This integer describes the number of Puss in Boots' moves as well.

Second line contains n integers a_i — number of candies in i-th pile.

Output

Print `Donkey` or `Puss in Boots` depending on the winner.

Constraints

$1 \le n \le 10^5$,
$0 \le a_i \le 10^9$.

Samples

Input (*stdin*)	Output (*stdout*)
2 2 2	Puss in Boots
4 0 47 0 0	Donkey

Notes

In the first example, there are two options:

- The Donkey takes one candy from a pile. Then, the Puss in Boots takes the second candy from the same pile and takes two candies from another pile.
- The Donkey takes two candies from a pile. Then, the Puss in Boots takes one candy from another pile twice.

In the second example, Donkey takes 47 candies from a second pile and wins.

E. Shrooks

Limits: 2 sec., 512 MiB

Shrek wants to place n rooks on a $n \times n$ board, so that the following conditions hold:

- There is exactly 1 rook in each row and each column;
- Manhattan distance between any two rooks doesn't exceed n.

To make things worse, Donkey already placed some rooks (fortunately, each row and each column still contains at most one rook). Find the number of ways to place the remaining rooks so that conditions hold. Since it can be large, output it modulo 998244353.

Here, the Manhattan distance between the rook in the cell on the intersection of row x_1 and column y_1 and the rook in the cell on the intersection of row x_2 and column y_2 is defined as $|x_1 - x_2| + |y_1 - y_2|$.

Input

Each test consists of multiple test cases. The first line contains a single integer t — number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n — size of the board.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$. If $a_i = -1$, it means row i doesn't yet have a rook. Otherwise, it indicates that there is a rook at the intersection of row i and column a_i.

Output

For each test case, output a single integer, the answer modulo 998244353.

Constraints

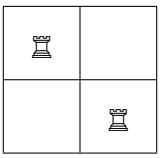
$1 \le t \le 10^5$,
$2 \le n \le 2 \cdot 10^5$,
the sum of n over all test cases doesn't exceed $2 \cdot 10^5$,
$a_i = -1$ or $1 \le a_i \le n$ for all $1 \le i \le n$,
if $a_i, a_j \ne -1$ for $i \ne j$, then $a_i \ne a_j$.

Samples

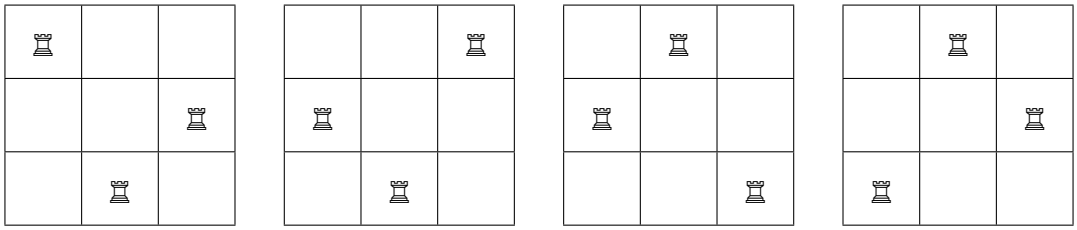
Input (*stdin*)	Output (*stdout*)
6 2 1 2 3 -1 -1 -1 4 1 -1 -1 -1 5 1 -1 -1 -1 5 6 3 -1 -1 -1 -1 4 10 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1	1 4 1 0 6 92

Notes

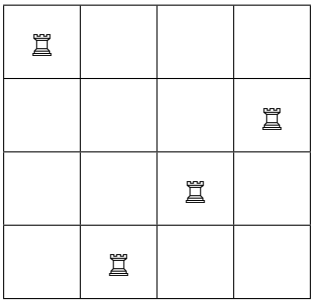
In the **first** test case, the rooks are already placed, and they satisfy the conditions:



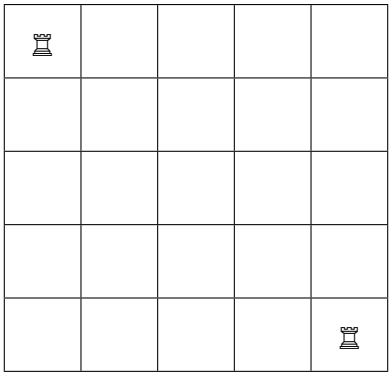
In the **second** test case, there are 4 ways to place rooks:



In the **third** test case, there is exactly one such way:



In the **fourth** test case, there are two rooks placed so far:



The manhattan distance between them is already $8 > 5$, so there is no satisfying placement of remaining rooks.

F. Magical Bags

Limits: 4 sec., 512 MiB

In the magical land of Far Far Away, Shrek and his friends stumbled upon a peculiar puzzle hidden deep within the Enchanted Forest. Donkey, who was always on the lookout for new adventures, discovered n mystical bags, each containing a unique collection of magical objects. Each object held a distinct magic power, represented by an integer.

As usual, Donkey couldn't resist turning it into a game. "Hey, Shrek, I dare you to make every pair of bags good while keeping the least number of magical objects in total," he challenged. Shrek, slightly annoyed but up for the task, asked, "And what do you mean by good, Donkey?".

"A pair of bags is *good* if you can choose one magical object from one bag that's stronger than one in the other bag, and another that's weaker. That way, the magic stays balanced!" Donkey explained, grinning mischievously.

More formally, a pair of bags A and B is called *good* if there exist elements $a_1 \in A$ and $b_1 \in B$ such that $a_1 < b_1$ and there also exist elements $a_2 \in A$ and $b_2 \in B$ such that $a_2 > b_2$. It's possible that $a_1 = a_2$ or $b_1 = b_2$.

Now Shrek's challenge is to figure out: What is the minimum number of magical objects that must remain in all the bags so that every pair of bags stays good if it was originally good? Of course, each bag must hold onto at least one magical object, just to keep the magic alive.

Input

The first line contains a single integer n — number of bags.

Next n lines contain a single integer k_i — number of magical object in the i-th bag, followed by k_i integers a_{ij} — magic power of the objects in i-th bag.

Output

Print a single number — minimum total number of magical objects remaining in the bags.

Constraints

$2 \le n \le 2 \cdot 10^5$,
$1 \le k_i$,
$\sum k_i \le 5 \cdot 10^5$,
$1 \le a_{ij} \le 10^9$,
all a_{ij} are distinct, even the objects in different bags can't have the same magic power.

Samples

Input (*stdin*)	Output (*stdout*)
4 3 4 7 10 2 1 9 4 11 2 8 14 3 6 12 13	7
4 1 1 1 2 1 3 1 4	4

Notes

In the first example, each pair of bags is good initially. One of the possible solutions:

1. Object with power 7.
2. Objects with powers 1 and 9.
3. Objects with powers 2 and 8.
4. Objects with powers 6 and 12.

In the second example no pairs of bags are good, but still one object should remain in each bag.

G. Shrek's Song of the Swamp

Limits: 4 sec., 512 MiB

In the faraway land of Duloc, Shrek has discovered a magical melody hidden within the ancient Swamp Songs. The melody, however, is written as a sequence of numbers that represent the croaks of frogs and the rustling of leaves. To unlock its secrets, Shrek needs to decode the longest repeating pattern in this mystical sequence, following a specific set of rules.

Shrek's melody sequence, s, is composed of various sounds, each represented by an integer. The song contains n sounds in total.

Shrek needs to find the longest possible subsequence that follows the strict Dulocian Harmony. To make the Dulocian Harmony, the following condition should hold: each sound must have a neighbour in the chosen subsequence that is the same sound as itself.

More precisely, Shrek must determine the length of the longest subsequence (not necessarily consecutive positions) $x_1x_2 \dots x_k$ of s such that for all $1 \le i \le k$ either $x_i = x_{i-1}$ or $x_i = x_{i+1}$, or both.

For example, Shrek can choose subsequences $[1, 1, 1, 2, 2]$, $[4, 4, 4, 4, 4]$ and can't choose $[1, 2, 2, 1, 1]$ since the first sound doesn't have the same adjacent sounds.

Help Shrek unlock this enchanted melody!

Input

The first line contains a single integer n — the number of sounds in Shrek's song.

The second line contains integer values $s_1, s_2, \dots, s_n$ representing the sounds from Shrek's song.

Output

In the first line, print a single integer — the length of the longest subsequence in s with the properties mentioned in the statement. If there are no such subsequences, print 0.

Constraints

$1 \le n \le 10^6$,
$-10^9 \le s_i \le 10^9$.

Samples

Input (*stdin*)	Output (*stdout*)
9 1 2 3 1 3 1 2 3 2	5
6 3 4 10 1 -3 5	0
4 1 2 1 1	3

Notes

In the first sample, the longest subsequence with the properties described in the statement is $s_1s_4s_6s_7s_9 = [1, 1, 1, 2, 2]$.

In the second sample, there is no subsequence with those properties.

H. Shreckless

Limits: 2 sec., 512 MiB

Lord Farquaad has a table a of integers with n rows and m columns. Since he is obsessed with order, he wants each row of the table to be sorted in nondecreasing order.

Shrek, who finds Farquaad's need for order ridiculous, has other plans. He can arbitrarily permute the numbers in each column, and wants to make sure that not a single row in the table is nondecreasing. Can he achieve his goal?

An array $a_1, a_2, \ldots, a_k$ is called nondecreasing, if $a_1 \leq a_2 \leq \ldots \leq a_k$.

Input

Each test consists of multiple test cases. The first line contains a single integer t — the number of test cases. The description of the test cases follows.

The first line of each test case contains two integer n, m — dimensions of the table.

The i-th of the next n lines contains m integers $a_{i,1}, a_{i,2}, \ldots, a_{i,m}$ — elements of the i-th row.

Output

For each test case, if Shrek can make each row not nondecreasing, output `YES`. Else, output `NO`.

Constraints

$1 \leq t \leq 2 \cdot 10^4$,
$2 \leq n, m \leq 10^5$,
$n \cdot m \leq 2 \cdot 10^5$,
the sum of $n \cdot m$ over all test cases doesn't exceed $2 \cdot 10^5$,
$1 \leq a_{i,j} \leq 10^9$.

Samples

Input (*stdin*)	Output (*stdout*)
3 2 2 69 69 2024 42 3 3 1 1 1 1 1 1 2 2 2 3 4 1 1 1 1 1 1 1 1 2 2 2 2	YES NO YES

Notes

In the **first** test case, initially the first row is $[69, 69]$, and therefore is nondecreasing. However, Shrek can swap the numbers in the first column, getting

2024	69
69	42

In the **second** test case, there is no way to rearrange columns to make each row not nondecreasing.
In the **third** test case, Shrek can rearrange columns as follows:

1	1	2	1
1	2	1	1
2	1	1	2

I. Donkey, Keep Watch

Limits: 4 sec., 512 MiB

Donkey, Shrek and Puss in Boots have snuck into the Fairy Godmother's potion factory to find a potion that will solve the ogre's marital problems. However, they are unable to find such a potion, so they will have to brew one themselves.

In the factory, there are n potions, with the i-th potion having potency a_i. They have to choose some non-empty subset of potions to brew together so that they create a good potion. The new potion will have the following characteristics:

- Its sweetness, calculated by taking the bitwise XOR of all potencies of the potions in the brew.
- Its aroma, calculated by taking the bitwise AND of all potencies of the potions in the brew.
- Its duration, calculated by taking the bitwise OR of all potencies of the potions in the brew.

A brew is good if and only if the sum of its sweetness and its aroma is equal to its duration. You need to figure out the number of ways Shrek and his friends can brew a good potion.

In other words, find the number of non-empty subsequences s of a such that $\text{XOR}(s) + \text{AND}(s) = \text{OR}(s)$.

Input

The first line of the input contains a single integer n — the number of potions available.

The second line of the input contains n integers — the potencies of the potions.

Output

Print a single integer — the number of different ways to brew a good potion. Since this number can be large, print it modulo $10^9 + 7$.

Constraints

$1 \le n \le 10^6$,
$0 \le a_i \le 15 \cdot 10^3$.

Samples

Input (*stdin*)	Output (*stdout*)
5 1 2 3 4 5	11

Notes

In the sample case, the ways to create a good potion are by taking the following subsets of potions: $\{1,2\}$, $\{1,3\}$, $\{1,4\}$, $\{1,5\}$, $\{2,3\}$, $\{2,4\}$, $\{2,5\}$, $\{3,4\}$, $\{3,5\}$, $\{4,5\}$ and $\{1,2,4\}$; which is 11 in total. For example, if we take the set $\{1,2,4\}$, the bitwise OR is 7, the bitwise AND is 0 and the bitwise XOR is 7, and since $0 + 7 = 7$, this indeed yields a good potion.

J. Make Swamp Great Again

Limits: 2 sec., 512 MiB

Around the large swamp where Shrek lives, there are n forest creatures, each living in a house arranged in a circle such that creatures i and $i+1$ live in neighboring houses, and the n-th creature is also a neighbor of the 1st creature. Each creature has a preferred swamp temperature t_i in which it feels most comfortable. One day, Shrek decided he wanted all the creatures to experience the same comfort in the swamp by having the same preferred temperature.

Every evening, Shrek invites a few creatures for a meeting. During this meeting, a group of three creatures from consecutive houses is selected, and one of them can change its preferred temperature to either the minimum or the maximum preferred temperature among the three.

For each creature, determine the minimum number of evenings required for Shrek to make all the creatures have the same preferred temperature as the initial temperature of this creature.

Input

The first line of the input contains a single integer n — the number of forest creatures.

The second line contains n integers t_i — the preferred temperature of the creature, given in clockwise order around the swamp.

Output

Print n space separated numbers in a single line. The i-th number represents the minimum number of evenings required to make all preferred temperatures equal to t_i.

Constraints

$3 \leq n \leq 10^5$,
$1 \leq t_i \leq 10^5$.

Samples

Input (*stdin*)	Output (*stdout*)
6 4 7 47 4 77 47	4 6 4 4 5 4

Notes

The images below show each evening's meeting, with groups of three creatures gathering. By the end, all creatures preferred temperature is 4 degrees.

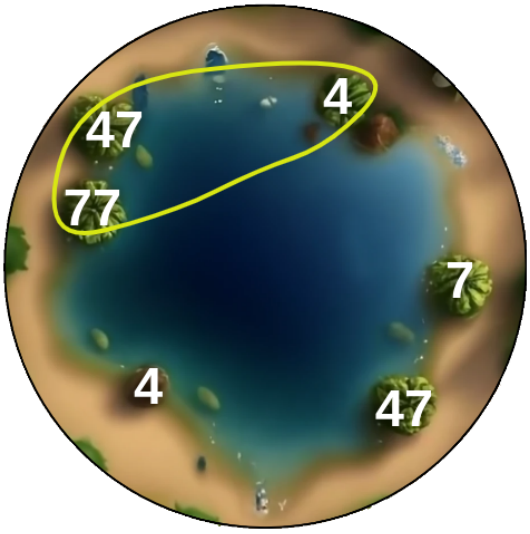


Figure 1: First evening

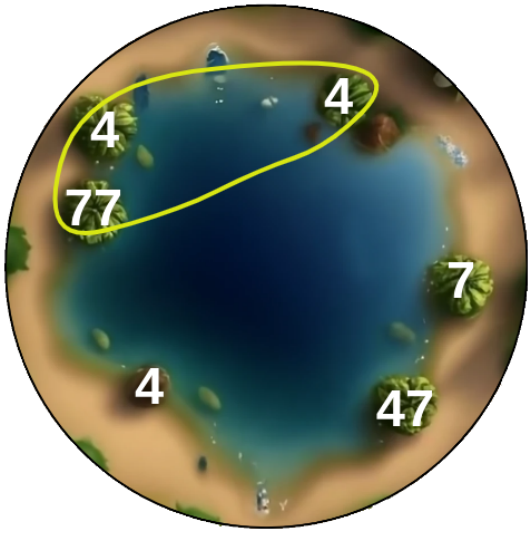


Figure 2: Second evening

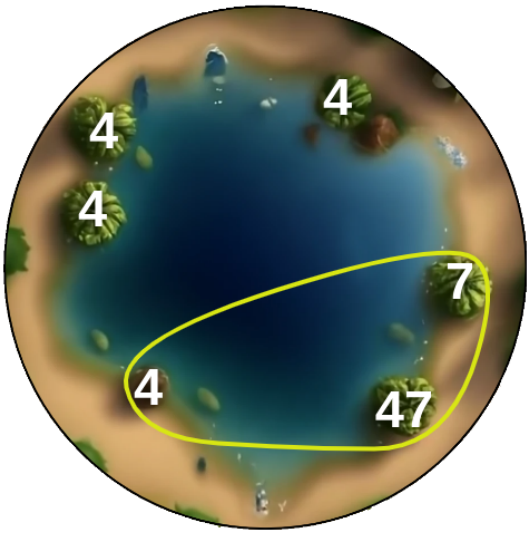


Figure 3: Third evening

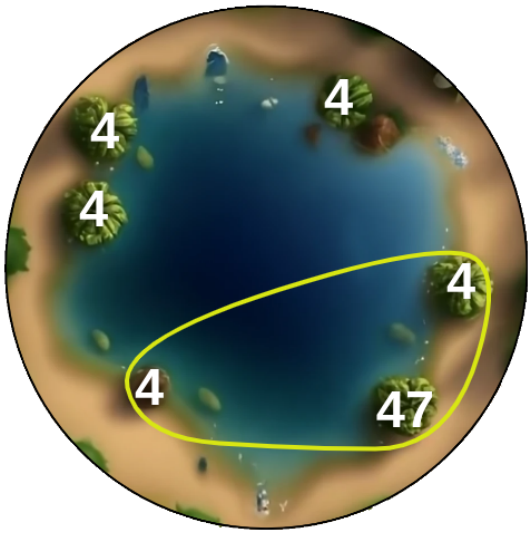


Figure 4: Fourth evening

K. Intrusive Donkey

Limits: 4 sec., 512 MiB

In the land of Far Far Away, Donkey just can't stop talking. Sometimes, he feels that he should definitely repeat one of his previous thoughts. He repeats his phrase in a specific way, where each letter is doubled (for example, `aabcb` turns into `aaaabbccbb`). Given a string s of length n representing Donkey's initial phrase, there are two types of events:

- Donkey changes his phrase, repeating some part of it. More specifically, he chooses some substring from positions l to r and doubles it. (For example, if the string is `aabc` and Donkey repeats the part from 2 to 3, the resulting phrase becomes `aaabbc`).
- Shrek cannot hold in his mind everything that intrusive Donkey said. He is interested what is the i-th letter in Donkey's current phrase.

Note that event of first type changes Donkey's phrase. Shrek needs help managing Donkey's endless questions so he can keep his peace of mind!

Input

First line contains two integers n and q — the length of string s and the number of events.
Second line contains string s of lowercase English letters — the initial Donkey's phrase.
Next q lines contain events as described in the problem statement.

- $1\ l\ r$ — the first type of event.
- $2\ i$ — the second type of event.

Output

For each event of the second type, print the answer in a separate line.

Constraints

$1 \le n, q \le 2 \cdot 10^5$,
$1 \le l \le r \le 10^{18}$,
$1 \le i \le 10^{18}$,
the length of the phrase will not be greater than 10^{18},
All indices will be up to $|s|$ at the moment of the query.

Samples

Input (*stdin*)	Output (*stdout*)
4 7 abac 2 2 2 3 1 2 3 2 3 2 4 2 5 2 6	b a b a a c
5 4 shrek 1 1 2 2 7 1 1 7 2 7	k h

Notes

In the first sample, the phrase of the Donkey changes from `abac` to `abbaac`.

In the second sample, the phrase of the Donkey changes from `shrek` to `sshhrek`, and then to `sssshhhhrreekk`.

L. Ogre Sort

Limits: 3 sec., 512 MiB

What happens if you cast an ogre curse on a stick of RAM? It turns out that it can circularly move arbitrary sequences of data in an efficient manner, but only at night.

Find the minimum cost (and a valid sequence of moves) needed to sort a permutation v of length n. All elements of the permutation are indexed from 1 to n.

The only permitted type of move allows you to take an element from some position x and insert it at another position y, shifting all elements in between by one. The cost of such a move is y.

Formally, a move takes an element valued t from position x, "freeing" the index x. We then shift the remaining elements in v, such that the "free" position becomes y. We then put t in the free position at index y.

For example, if we have a permutation [4, 3, 2, 1], some of the possible moves:

- $x = 2$, $y = 4$, the resulting permutation is [4, 2, 1, 3], the cost of the move is 4.
- $x = 2$, $y = 1$, the resulting permutation is [3, 4, 2, 1], the cost of the move is 1.

Input

The first line contains an integer n — the length of the permutation.

The second line contains n integers $v_1, v_2, \ldots, v_n$ — the values of the permutation.

Output

On the first line, print two numbers *min_cost* and *len_moves* — the minimum cost needed to sort the permutation and the length of the proposed sequence of moves respectively.

The next *len_moves* lines should each contain two integers x_k, y_k each, signifying that the k-th operation should move the element from position x_k to position y_k ($1 \le k \le len\_moves$, $1 \le x_k, y_k \le n$).

If several possible sequences of moves exist, you can print any of them. Note that you don't need to minimize the number of moves, only the total cost. It's guaranteed that there always exist a sequence of moves to sort a permutation.

Constraints

$1 \le n \le 3 \cdot 10^5$,
$1 \le v_i \le n$,
$v_i \ne v_j$ for all $1 \le i < j \le n$.

Samples

Input (*stdin*)	Output (*stdout*)
4 1 2 4 3	3 1 4 3
5 2 4 1 3 5	3 2 4 2 4 1
3 1 2 3	0 0

Notes

- For the first example, the only move effectively swaps the elements on positions 3 and 4.
- For the second example, the permutation resulting after the first move is $[2, 3, 4, 1, 5]$.
- For the third example, the permutation is already sorted, so the optimal cost is zero, and no moves are needed.

M. Enchanted Lawns Quest

Limits: 2 sec., 512 MiB

In the land of Far Far Away, Shrek and Donkey found themselves wandering through the Enchanted Forest, connected by a mystical network of enchanted trails winding through magical lawns. Lawns are linked by trails, with each trail carrying a magical resistance value. This resistance determines how difficult it is to travel along a given trail.

There is exactly one route between each pair of lawns, forming a structure known as a tree. In this network, the *hardest path* (known as the diameter of the tree) is defined as the maximum possible sum of resistances along any simple route between two lawns.

As usual, Donkey had an idea. "Shrek, what if we added a bit of magic to these trails?". Shrek, always ready for a new challenge, agreed but knew they had a limited supply of magical energy — exactly w units in total. They had to carefully distribute this extra magic across the trails to minimize the possible diameter, while ensuring they used all w units.

Let a_i represent the initial resistance value of i-th trail, then they needed to find new values b_i such that:

- $b_i \geq a_i$,
- $\sum b_i = w + \sum a_i$,
- each b_i is an integer, since magic can only be applied in whole units,
- the diameter of the enchanted lawns network (the hardest route between any two lawns) is minimized after the adjustments.

Can you help them determine the best way to spread the magic?

Input

The first line contains two integers n and w — number of lawns and amount of magic energy to distribute.

The next $n - 1$ lines contain three integers v_i, u_i and a_i each — there is a trail between lawns v_i and u_i with a magical resistance value a_i.

Output

Print a single integer — the minimum possible diameter after distributing w units of magical energy.

Constraints

$2 \leq n \leq 2 \cdot 10^5$,
$1 \leq w \leq 10^{12}$,
$1 \leq u_i, v_i \leq n$,
$1 \leq a_i \leq 10^7$,
all trails form a tree.

Samples

Input (*stdin*)	Output (*stdout*)
5 7 1 2 2 1 3 4 1 4 5 3 5 2	14
2 7 1 2 4	11

Notes

Possible way to add magic in the first sample:

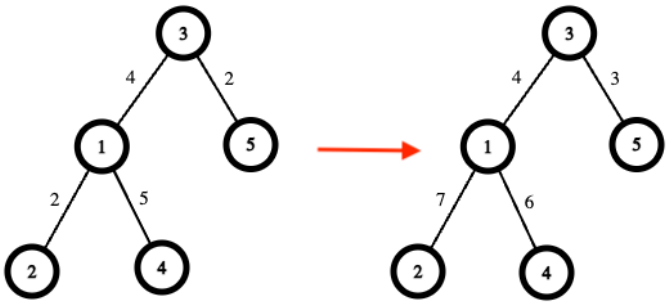


Figure 5: The first sample.

In the second sample, you have to add 7 to a single edge.