Problem A. Another Day of Sun

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

'Cause morning rolls around, and it's another day of sun!

But as a sleep addict, Sean spends lots of hours sleeping, and thus, he can't even remember which day it is when he wakes up.

So starting from someday, he started taking notes: when he wakes up, he writes a number about whether there's sunlight outside. If there is, he will write a 1 on it; otherwise, he will write a 0 . And after taking notes, without enough time for sunlight to fade or for moonlight to dim, he falls asleep again. We assume that every time Sean wakes up, he sees either the sunlight or the moonlight, but not both.

So the notes actually form an array of length n : $[a_1, a_2, \ldots, a_n]$ $(\forall 1 \le i \le n, 0 \le a_i \le 1)$, where a_i represents the i-th number Sean wrote.

However, as time goes on, some numbers written become ambiguous and are just impossible to identify, and you can't really tell whether it is a 1 or a 0 . So if there are k numbers that you can't identify, there can be 2^k different arrays.

For each array, you can calculate the minimum number of days on which Sean sees the sunlight at least once that is consistent with the notes the array represents. If you add the results of different arrays together, what will you get? As the answer can get quite enormous, output it modulo 998 244 353 .

Input

Each test contains multiple test cases. The first line contains the number of test cases T $(1 \le T \le 10^4)$.

Each test case consists of two lines.

The first line contains one integer n $(2 \le n \le 5 \times 10^5)$, the number of notes.

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(-1 \le a_i \le 1)$, each number written on the note. The number is unknown if and only if $a_i = -1$.

It is guaranteed that $\sum n$ over all test cases in one test will not exceed 5×10^5.

Output

For each test case, output 1 integer — the sum of results modulo 998 244 353 .

Example

standard input	standard output

3
3
1 0 1
3
0 0 0
3
1 -1 1
	2
	0
	3

Note

In the first test case, when Sean takes the first note and the third note, the sun he has seen must have been of 2 different days.

In the second test case, Sean never sees the sun, so the answer is 0.

In the third test case, the array can be $[1, 1, 1]$ or $[1, 0, 1]$. If the array is $[1, 1, 1]$, the notes can be taken in the same day, so the result is 1. If the array is $[1, 0, 1]$, the first and third notes must have been taken in different days, so there are at least 2 days of sun that Sean has seen. So the answer is $1 + 2 = 3$.

Problem B. Bitwise Perfect

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

Binary army ants are preparing for a cruel war, but things become more complicated than they thought.

There are n ants in the squad. Originally, each army ant i $(1 \le i \le n)$ has its power a_i .

However, it turns out something strange will happen during the war: a random pair (i, j) $(1 \le i < j \le n)$ will be chosen, and the ant i and the ant j will disappear, and a new ant with power $a_i \oplus a_j$, which is the bitwise exclusive OR (XOR) of a_i and a_j, will magically appear.

This kind of thing is so rare that it happens at most once throughout the war.

The ants think the total strength of the whole squad is $\sum_{i=1}^{n} 2^{a_i}$, and the squad is called **bitwise perfect** if the event above could never decrease its strength.

Too busy practicing, ants don't have enough time to check whether the squad is bitwise perfect. Can you help them with this? There can be T squads.

Input

Each test contains multiple test cases. The first line contains the number of test cases T $(1 \le T \le 10^5)$.

Each test case consists of two lines.

The first line contains one integer n $(2 \le n \le 5 \times 10^5)$, the number of ants in the squad.

The second line contains n integers $a_1, a_2, \dots, a_n$ $(1 \le a_i \le 10^{18})$, the power of each ant.

It is guaranteed that $\sum n$ over all test cases in one test will not exceed 5×10^5.

Output

For each test case, output `YES` if the squad is **bitwise perfect** and `NO` otherwise. You can print the answer in any case.

Example

standard input	standard output
`3`	`YES`
`2`	`YES`
`3 5`	`NO`
`4`	
`1 2 4 8`	
`3`	
`1 2 3`	

Note

In the first case, if the two ants merge, as $3 \oplus 5 = 6$, the total strength of the whole squad changes from $2^3 + 2^5 = 40$ to $2^6 = 64$, so the event will not decrease the total strength. So the squad is **bitwise perfect**.

Problem C. Colorful Tree

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	512 megabytes

It's a colorful world in Color Town!

Color Town is well known for each place of interest: the Colorful Tree. The Colorful Tree is not a real tree. It's actually a gigantic painting on the ground which looks like a tree if you take a hot air balloon and see the painting from up in the sky.

Families in the town work together to renew the painting each year. There are n families in the town, and there are n parts that form the whole painting. Family i paints Part i.

Why does the painting look like a tree? Actually, the structure of the painting is almost like a rooted tree in graph theory! Each part other than Part 1 has its "parent", and the parent of Part u is p_u . Part i and Part j are adjacent if and only if i is the parent of j or j is the parent of i .

Each family produces its own paint. But since the paint producing technique of the family can be unstable, the color of the paint they produce can be quite different. We note each color as an integer, and the color of the paint produced by Family i can be a random integer in the range $[l_i, r_i]$ equiprobably.

To make work less tiring, each family paints the part entirely using the paint it produces this year. Together, they form a beautiful painting!

There is another thing about paint that you should know: as time passes, the family may lose or improve its technology, so $[l_i, r_i]$ can change over time. Relevant updates will be provided.

To show the painting to the world, the mayor of the town, Sean, decides to take a picture of the painting each year. In the i-th year, when he takes a photo of the painting, the photo taken shows the parts in the subtree of u_i fully, and no other parts are shown in the photo. Part i is in the subtree of Part j if there is a sequence $v_1, v_2, \ldots, v_k$ such that $v_1 = i, p_{v_t} = v_{t+1}$ $(1 \le t < k), v_k = j$. Specially, Part i is in the subtree of Part i .

Sean wants to tell others how colorful the photo is, so he defines the colorfulness as the number of great components in the photo. A component is an area that is in one color, and any point in the area can go through points in the components to get to another point. A great component is a component which can't be completely covered by any other component.

Let the colorfulness be C_i in the i-th year. As the randomness of the paints produced, C_i can be a random variable each year. To learn more information about C_i, Sean asks you about the expectation and the variance value of C_i. Can you help him?

Recall that the variance of a random variable X is $\mathrm{Var}(X) = \mathrm{E}[(X - \mathrm{E}X)^2]$ where $\mathrm{E}X$ is the expectation of X .

It's guaranteed that the root of the tree is 1, and the tree (except the example) is randomly generated this way:

— Let $rnd(a)$ be the result of selecting a number from $1, 2, \ldots, a$ equiprobably.

— We use $[p_2, p_3, \ldots, p_n]$ to describe the rooted tree, which means the "parent" of Part i is p_i .

— The array $[p_2, p_3, \ldots, p_n]$ is generated this way: $p_2 = rnd(1), p_3 = rnd(2), \ldots, p_{i+1} = rnd(i), \ldots,$ $p_n = rnd(n-1)$.

The queries are not necessarily randomly generated.

Input

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

Each test case consists of many lines.

The first line contains 2 integers n, q ($4 \le n \le 5 \times 10^5, 1 \le q \le 10^5$), the size of the tree and the number of queries.

The second line contains $n-1$ integers $p_2, p_3, \ldots, p_n$ ($1 \le p_i < i$), where p_i is the parent of i.

The third line contains n integers $cl_1, cl_2, \ldots, cl_n$ ($1 \le cl_i < 998\ 244\ 353$) and the fourth line contains n integers $cr_1, cr_2, \ldots, cr_n$ ($cl_i \le cr_i < 998\ 244\ 353$). The original color range of Family i is $[cl_i, cr_i]$.

Each line from the 5-th to the $(q+4)$-th contains a query. The query can be one of the two types:

— `1 u l r`: Type-1 query. Change the color range of vertex u ($1 \le u \le n$) into $[l, r]$ ($1 \le l \le r < 998\ 244\ 353$).

— `2 u`: Type-2 query. Ask about the expected colorfulness and its variance in the subtree of vertex u ($1 \le u \le n$).

It is guaranteed that $\sum n$ over all test cases in one test will not exceed 5×10^5, and $\sum q$ over all test cases in one test will not exceed 10^5.

Output

For each Type-2 query, output 2 integers to represent the expected colorfulness and its variance in the subtree of vertex u.

It can be proved that the answer is always a rational number, which can be expressed as $\frac{p}{q}$ ($\gcd(p,q) = 1$, $p, q \in \mathbb{Z}, 998\ 244\ 353 \nmid q$). You should output a value x such that $q \times x \bmod 998\ 244\ 353 = p$ and $x \in [0, 998\ 244\ 353)$.

Example

standard input	standard output

2
4 5
1 2 2
1 1 2 3
1 3 2 3
2 1
2 2
1 2 2 3
2 1
2 2
4 7
1 1 1
1 998244352 1 998244352
998244352 998244352 1 998244352
2 1
1 2 1 998244352
2 1
1 3 998244352 998244352
2 1
1 4 1 1
2 1
	3 0
	332748120 887328314
	3 0
	2 0
	7 998244339
	7 998244345
	7 998244343
	7 998244345

Problem D. Donkey Thinks...

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

Donkey thinks potato chips are the best food ever!

So today, when he decides to go on a long journey, he wants his backpack filled with all kinds of potato chips. He searches through the snack zone in his home and finds lots of potato chips.

To better decide which bags of potato chips to bring with him (a subset of the total bags, possibly none of them), he defines the property of a bag of chips as follows:

— h_i. The happiness this bag of chips can give to Donkey.

— s_i. The space this bag of chips occupies.

— d_i. The delicacy of this bag of chips.

For simplicity, we note h_i, s_i, d_i as the "happiness", "space" and "delicacy" of the bag.

The total occupied space of the chosen bags can't exceed the volume of the backpack, which is V.

However, the unoccupied space may cause bumps when Donkey moves during the journey, which further causes value loss. If the chosen chips are $i_1, i_2, \ldots, i_k$ $(k \geq 1)$ and the unoccupied space is U, the total value loss on account of bumps is $(d_{i_1} + d_{i_2} + \cdots + d_{i_k}) \times U$. If you choose no bag of chips, the value loss is 0.

Considering both the advantages and disadvantages of bringing chips, the value of the whole backpack is the sum of happiness brought by these bags of chips minus the value loss. Donkey wants to maximize this value, but just can't make the decision. Help is needed for this!

Input

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

Each test case consists of many lines.

The first line contains 2 integers n, V $(1 \leq n \leq 10^5, 1 \leq V \leq 500)$, the number of chip bags and the total volume of the bag.

Each line from the 2-nd to the $(n+1)$-th contains 3 integers h_i, s_i, d_i $(1 \leq s_i \leq 500, 1 \leq h_i, d_i \leq 10^9)$, the "happiness", "space" and "delicacy" of the i-th bag of chips.

It is guaranteed that $\sum n$ over all test cases in one test will not exceed 10^5, and $\sum V^2$ over all test cases in one test will not exceed 2.5×10^5.

Output

For each test case, output one integer — the maximum value.

Example

standard input	standard output

2
2 5
10 2 1
2 2 100
2 5
10 2 1
2 3 100
	7
	12

Note

In the first case, Donkey only chooses the first bag, resulting in a value of $10 - (5 - 2) \times 1 = 7$.

In the second case, Donkey chooses both the first bag and the second bag, resulting in a value of $10 + 2 - (5 - 2 - 3) \times (1 + 101) = 12$.

It can be proved that there are no better strategies.

Problem E. Effective Numbers

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

Square numbers are always lovable. A square number is a number that can be expressed by the square of an integer.

Here comes the group made of $l, l+1, \ldots, r$, and they want to be square numbers. The witch offers them a special potion. When a number drinks a bottle of potion, it can enlarge itself by one of its prime factors, and it can choose which one to add. As its dream is always to become a square number, if it can enlarge itself into a square number, it will always do so.

The potion is kind of dangerous, so drinking 2 or more bottles can cause the number to disappear, so the numbers won't do so. Also, as the potion can be quite expensive, the group decides to buy as few as they can while making the most amount of numbers into square numbers. This is the most effective way to do it!

How many bottles of potion should they buy?

Input

The input contains one line. The line consists of two integers l, r $(2 \le l < r \le 10^{18})$.

Output

A number representing the number of bottles of potion to buy.

Example

standard input	standard output
`2 9`	`2`

Note

In the interval $[2, 9]$, there are two numbers which need a bottle of potion: 2 as $2 + 2 = 4 = 2^2$ and 6 as $6 + 3 = 3^2$.

Problem F. Field of Fire

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

Alarm! Fire is taking over the forest!

The shape of this forest is like a ring and can be divided into n parts. Part i $(1 \le i \le n)$ is adjacent only to Part $((i-2) \bmod n)+1$ and Part $(i \bmod n)+1$. If no action is taken and there is a fire in Part i, its adjacent parts will also be engulfed by flames in one minute.

As a responsible resident yourself, you have called the firefighters to come as soon as possible to put out this scary fire. However, it will take t_0 minutes before they can ever get here. Is there anything you could do?

There is! You decide to set a controllable fire by yourself on exactly one part of the forest to get a fire isolation zone immediately before the fire can ever get to it. In that way, the fire can never spread over this place afterwards.

But it's hard to make a wise decision: You need to choose the place of the fire isolation zone to guarantee more parts of the forest remain unburned. What is the maximum number of unburned parts you can get before firefighters come?

Input

Each test contains multiple test cases. The first line contains the number of test cases T $(1 \le T \le 10^4)$.

Each test case consists of 2 lines.

The first line contains two integers n, t_0 $(1 \le n, t_0 \le 10^5)$, the number of parts in the forest and the minutes firefighters need to get to the forest.

The second line contains a string s $(|s| = n)$ consisting of 0 and 1. The part i is currently on fire if and only if $s_i = 1$. It is guaranteed that there exists one i $(1 \le i \le n)$ such that $s_i = 1$.

It is guaranteed that $\sum n$ over all test cases in one test will not exceed 5×10^5.

Output

For each test case, output one integer — the minimum number of unburned parts in the forest when the firefighters come.

Example

standard input	standard output

3
5 2
10000
5 3
10000
10 1
1000000100
	1
	0
	4

Note

For the first test case, once you burn Part 2, the fire only gets time to burn Part 4 and 5, leaving Part 3 unburned.

Problem G. Geometry Friend

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

A convex polygon lies lazily on the Euclidean xOy plane. What is it like to have a friend? It thinks to itself.

Tiny as it is, a point P appears at the position (x, y) and catches the attention of the polygon. They become friends, and the polygon loves rotating anti-clockwise around this little point P, as in that way, it can cover some area it never covered before. An area is covered by the polygon if every point in it is inside or on the edges of the polygon.

But P is worried: There can be one day when the polygon becomes bored with rotating. It thinks that when the area that has once been covered by the polygon doesn't become larger, the polygon will just ditch it and they will no longer be friends.

The angular velocity of the rotation of the polygon is 1 rad per year, and let the time when the polygon starts rotating be Time 0. Help P know the moment when his worries may come true.

Input

Each test contains multiple test cases. The first line contains the number of test cases T $(1 \le T \le 10^4)$.

Each test case consists of many lines.

The first line contains 3 integers n, x, y $(3 \le n \le 5 \times 10^5, |x| \le 10^9, |y| \le 10^9)$, where n is the number of edges of the convex polygon and (x, y) is the position of P.

Each line from the second to the $(n+1)$-th contains two integers x_i, y_i $(|x_i| \le 10^9, |y_i| \le 10^9)$, the position of one vertex in the polygon. It is guaranteed that the vertices given are in anti-clockwise order.

It is guaranteed that $\sum n$ over all test cases in one test will not exceed 5×10^5.

Output

For each test case, output one value — the moment when P's worries may come true. If the real answer is ans, your answer ans' is considered correct if $\frac{|ans-ans'|}{\max(1,ans)} \le 10^{-6}$.

Example

standard input	standard output

3
4 0 0
1 0
0 1
-1 0
0 -1
3 0 0
1 1
2 -1
2 0
3 0 0
0 0
1 -1
1 1
	1.570796326794897
	6.283185307179586
	4.712388980384690

Problem H. Highway Upgrade

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

The highway system in NowLand has been used for quite some decades and requires upgrades.

In the country, there are n cities and m one-way highways that go from one city to another. Using the i-th highway, one car can go from City u_i to City v_i in t_i minutes. An upgrade reduces the time used in the journey by w_i minutes. Each highway can be upgraded as many times as you want.

As a selfish person, the president of NowLand only considers his own interests. It's his last job to upgrade the highways, and after that, he will retire. After he retires, he will leave City 1, the capital of NowLand, for City n, where he will spend the rest of his life. He will only use highways for this journey, so he only cares about the time to go to City n from City 1 and wants it to be as short as possible.

But as the government doesn't have enough money, there's a limited budget for upgrades. With the budget yet to be confirmed, the president needs to prepare for different conditions. That's why he wants to know the minimum time he needs to get to City n from City 1 if he can do k upgrades. What are the results, though?

It's guaranteed that it is always possible to get to City n from City 1 using only highways.

Input

Each test contains multiple test cases. The first line contains the number of test cases T $(1 \le T \le 10^4)$.

Each test case consists of many lines.

The first line contains 2 integers n, m $(4 \le n \le 10^5, 1 \le m \le 3 \times 10^5)$, the number of cities and highways in the country.

Each line from the 2-nd to the $(m+1)$-th contains 4 integers u_i, v_i, t_i, w_i $(1 \le u_i, v_i \le n, u_i \ne v_i,$ $2 \le t_i \le 10^{12}, 1 \le w_i \le \min(t_i - 1, 10^9))$, representing the starting point, ending point, original travel time, and the upgrade parameter of the i-th highway. It is guaranteed that it is possible to travel from City 1 to City n using highways.

The $(m+2)$-th line contains one integer q $(1 \le q \le 3 \times 10^5)$, the number of queries to ask.

Each line from the $(m+3)$-th to the $(m+q+2)$-th contains one integer k_i $(1 \le k_i \le 10^9)$, the number of upgrades. It is guaranteed that $\forall 1 \le j \le m, t_j - w_j \times k_i > 0$.

It is guaranteed that $\sum n$ over all test cases in one test will not exceed 2×10^5, and $\sum m$ and $\sum q$ over all test cases in one test will not exceed 6×10^5 .

Output

For each test case, output q integers — the minimum time to travel from City 1 to City n using k_i upgrades.

Example

standard input	standard output

1
4 4
1 2 15 1
1 3 20 2
2 4 10 1
3 4 10 1
2
9
1
	12
	24

Problem I. Identical Somehow

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

A hash function converts data into an output string with a fixed number of characters; therefore, it creates a "tag"for each data.

In this problem, inputs are always integers. It can be a bad thing if two different integers x, y $(x \neq y)$ are converted into the same string. These two integers are considered identical somehow from the hash function. It is called a hash collision.

We consider the following hash function with parameter k:

$H(x) = (x \bmod k) + (k \bmod x)$

For each input pair (x, y) $(x \neq y)$, is there a hash function that results in a hash collision? If there is, output any such k ; otherwise, output -1 .

Input

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

Each test case consists of one line. The line contains two integers x, y $(1 \leq x, y \leq 10^9, x \neq y)$, the elements of the pair.

Output

For each test case, output one integer — the parameter k $(1 \leq k \leq 10^{18})$ that causes a hash collision. If there is no such positive number no larger than 10^{18} that satisfies the condition, output -1 instead.

Example

standard input	standard output

2
5 9
9 15
	4
	6

Note

For the first case, $5 \bmod 4 + 4 \bmod 5 = 1 + 4 = 5, 9 \bmod 4 + 4 \bmod 9 = 1 + 4 = 5$.

For the second case, $9 \bmod 6 + 6 \bmod 9 = 3 + 6 = 9, 15 \bmod 6 + 6 \bmod 15 = 3 + 6 = 9$.

Problem J. Just Curve It

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	1024 megabytes

Sean is glad that he has already finished grading all his students' tests. But he is no longer glad when he sees the final scores. How low they are!

Luckily for both Sean and his students, the dean's office allows Sean to "curve"the scores to make the scores "look better"as long as higher scores are still higher.

The scores are in the range $[1, 100]$, and there are two classic operators used in curving:

— Type 1: $f_1(x) = kx + b$, where $10^{-2} \le k \le 1, 0 \le b \le 10^4, k + b \ge 1$.

— Type 2: $f_2(x) = \sqrt{x}$.

After curving, Sean will map the curved score into the target score range, which we don't really care for now.

Sean has an array of operators of length $n : [g_1, g_2, \ldots, g_n]$, each of them is either Type 1 or Type 2.

To test the curving result, Sean tries the functions with an initial value x.

Each time, he chooses a subarray of the operator array $[g_l, g_{l+1}, \ldots, g_r]$, and uses the operator like this:

— Initiate a_0 with a value x.

— Calculate the variables $a_1, a_2, \ldots, a_{r-l+1}$ using the following equations: $a_1 = g_l(a_0), a_2 = g_{l+1}(a_1), \ldots,$ $a_{r-l+1} = g_r(a_{r-l})$.

As Sean is way too busy and lazy to do this, he asks you to help him. To know about the curving result and make sure you fully follow his process, he asks you about both a_{r-l+1} and $\sum_{i=1}^{r-l+1} a_i$.

Sean may change some of the operators between the two times he asks questions. Can you handle it?

Input

The first line contains an integer n $(1 \le n \le 10^5)$, the length of the operator array.

Each line from the 2-nd to the $(n + 1)$-th contains a line that represents the function g_i. Each line can be of two forms:

— `1 k b`: The operator is Type 1 — $kx + b$ $(10^{-2} \le k \le 1, 0 \le b \le 10^4, k + b \ge 1)$. k, b are two decimals, and there are exactly 2 decimal places after the decimal point for both k and b .

— `2`: The operator is Type 2 — $\sqrt{x}$.

The $(n + 2)$-th line contains an integer q $(1 \le q \le 10^5)$, the number of queries.

Each line from the $(n + 3)$-th to the $(n + q + 2)$-th contains a query. The query is in one of the given two forms:

— `1 l r x` $(1 \le l \le r \le n, 1 \le x \le 100)$: Type 1 query. Asking about a_{r-l+1} and $\sum_{i=1}^{r-l+1} a_i$ for the operator subarray $[g_l, g_{l+1}, \ldots, g_r]$ with initial value x. l, r, x are all integers.

— `2 idx op` $(1 \le idx \le n)$: Type 2 query. Changing the idx-th operator into op, where op is given in the form of operators.

Output

For each Type 1 query, output two values: a_{r-l+1} and $\sum_{i=1}^{r-l+1} a_i$.

Since the answers are not necessarily integers, if the real answer is ans, your answer ans' is considered correct if $\frac{|ans-ans'|}{\max(1,ans)} \le 10^{-4}$.

Example

standard input	standard output

5
1 1.00 10.00
2
1 0.50 50.00
2
1 0.95 20.00
5
1 1 5 80
2 3 2
1 1 5 80
2 4 1 0.35 65.00
1 3 5 100
	27.02893543 188.65806431
	21.66726225 125.98917842
	85.07500000 163.57500000

Problem K. K-th Memorable Sub-string

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	1024 megabytes

A string is a sequence of characters and a 01-string is a string composed of only 0-s and 1-s.

When reading a few 01-strings, Sean found some of them are pretty memorable: There is a prefix and a suffix that are the same with no overlapping. Also, they are separated by no more than k characters.

Then, he came across a long string s. He wants to dive deeper into this string and know about all the memorable sub-strings of s. A sub-string is a contiguous sequence of characters within a string.

He also wants you to do so, which is so inconsiderate. To make sure you really did this, he will ask queries like this:

— What is the q_i-th lexicographically-smallest sub-string of s that is memorable?

A word s_1 is lexicographically smaller than s_2 if and only if one of the following things happens:

— s_1 is a prefix of s_2 .

— There exists a j such that $\forall 1 \le i < j, s_1[i] = s_2[i]$ and $s_1[j] < s_2[j]$.

Also, if $s =' 011101'$, the sub-strings $s[1...2]$ and $s[5...6]$ are considered different, as different indexes are chosen from the original string.

Also, Sean wants to make sure you listen to the words he says, so questions are slightly adjusted based on the answer to the last question. Check the input description for further information.

Input

The first line contains two integers n, k ($2 \le n \le 10^4, 0 \le k \le n-2$), the length of the given string s and the parameter for the memorable sub-string.

The second line contains the string s, which is composed of only 0-s and 1-s.

The third line contains an integer q ($1 \le q \le 5 \times 10^5$), the number of questions Sean asks.

Each of the lines from the 4-th to the $(q+3)$-th contains an integer v_i ($1 \le v_i \le 10^9$), which represents the question that you need to answer.

Let $total$ be the total number of different memorable sub-strings in s, and let ans_i be the answer to the i-th query ($1 \le i \le q$). The answer ans_i is a 01-string, and it is the binary representation (ignore unnecessary leading zeros) of an integer x_i. Let $x_0 = 0$, so in the i-th query, $q_i = ((x_{i-1} \bmod 998\ 244\ 353 + v_i - 1) \bmod total + 1)$, you need to find out the q_i-th lexicographically-smallest memorable substring.

It is guaranteed that there is at least one memorable substring of s.

Output

For each query, output a string as the answer, the q_i-th lexicographically-smallest memorable substring.

It is guaranteed that the total lengths of the output strings will not exceed 10^7 .

Example

standard input	standard output

6 1
110111
2
5
6
	11011
	11

Note

The memorable sub-strings of the original strings are $s[2...4] =' 101'$, $s[1...2] =' 11'$, $s[4...5] =' 11'$, $s[5...6] =' 11'$, $s[1...5] =' 11011'$, $s[4...6] =' 111'$ (in lexicographical order). So, $total = 6$.

The first query asks about the 5-th element, so the result is 11011.

11011 is the binary representation of 27, so the second query asks about the (27 mod 998 244 353 $+ 6 - 1$) mod $6 + 1 = 3$-rd element, so the result is 11.

Problem L. Love Wins All

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

It's a loving community!

There are n residents in the community, and each resident i $(1 \le i \le n)$ in the community has a unique resident a_i $(1 \le i \le n)$ in the community whom he/she loves so much. Each two residents love different residents. A resident can love himself / herself. It is guaranteed that n is even.

One day, a bad thing happens: They need to choose 2 residents to be forbidden to get married forever.

And to prevent such a thing from happening in the future, the rest $n - 2$ residents decide to get married as $\frac{n}{2} - 1$ couples, each couple consisting of 2 persons (of course). It makes no sense that a couple consists of resident x and resident y while neither x loves y nor y loves x, so such a thing never happens.

So, as the planner, you need to figure out how you can arrange this. You want to know the number of different marriage plans. Two marriage plans are considered different if at least one of the following conditions is satisfied:

— In one plan, a person i is married, and in the other, he/she is not.

— In one plan, a person i is married to j, and in the other, he/she is not married to j.

As the number of plans can be quite enormous, output it modulo 998 244 353.

Input

Each test contains multiple test cases. The first line contains the number of test cases T $(1 \le T \le 10^4)$.

Each test case consists of two lines.

The first line contains 1 integer n $(4 \le n \le 5 \times 10^5)$, the number of residents in the community. It's guaranteed that n is even.

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le n)$, where a_i represents the one that the resident i loves. It is guaranteed that if $i \ne j$ $(1 \le i, j \le n)$, $a_i \ne a_j$.

It is guaranteed that $\sum n$ over all test cases in one test will not exceed 5×10^5 .

Output

For each test case, output 1 integer — the number of different marriage plans modulo 998 244 353.

Example

standard input	standard output

2
4
1 3 4 2
6
3 4 5 6 2 1
	3
	9

Problem M. Miscalculated Triangles

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	512 megabytes

In his homework, Sean ran into this problem:

— How many different shapes of triangles are there in a 2D-plane, with lengths of its sides in $\{1, 2, \ldots, s\}$ and perimeter no longer than l ? Two triangles are considered to be in the same shape if they can completely overlap using only translation and rotation. Note that **flips are not allowed** . So for ΔABC and $\Delta A'B'C'$ (A, B, C and A', B', C' are listed anti-clockwise.), if $AB = 2, BC = 3, CA = 4$ and $A'B' = 2, A'C' = 4, C'A' = 3$, they are **not** considered to be in the same shape.

Sean likes to consider problems in the binary system, so he uses the binary representation of numbers. He iterated over all possible (a, b, c) trying to find the answer, so what he wanted is the number of triplets (a, b, c) such that:

— $1 \le a, b, c \le s$ and are all integers.

— $a + b > c, a + c > b, b + c > a$

— $a + b + c \le l$

Then, he chose from these triplets such that each triplet chosen represents a different triangle. However, Sean was so bad at math that when he calculated $a + b + c$, he totally forgot about the carries, and therefore got the result of $a \oplus b \oplus c$, which is the bitwise exclusive OR (XOR) sum of a, b and c .

Despite this mistake, Sean wants to know if he makes other mistakes, so he asks you about the answer if the third condition is $a \oplus b \oplus c \le l$ rather than $a + b + c \le l$. Can you help him out with this?

Since the answer can be enormous, output it modulo 998 244 353 .

Input

Each test contains multiple test cases. The first line contains the number of test cases T $(1 \le T \le 10^4)$.

Each test case consists of a single line. The line contains 2 strings s_l, s_s $(|s_l| \le 10^5, |s_s| \le 10^5)$, the binary representation of the integers l and s . It is guaranteed that $l > 0$ and $s > 0$, and the first character of string s_s and s_l is always 1 .

It is guaranteed that $\sum |s_l|$ over all test cases in one test will not exceed 5×10^5 and $\sum |s_s|$ over all test cases in one test will not exceed 5×10^5.

Output

For each test case, output 1 integer — the number of triangles in different shapes modulo 998 244 353 .

Example

standard input	standard output

2
101 1111
1111 101
	267
	25