Problem A. Delete, Deduct, and Destroy

Alice is playing with a function $f(x, k)$. Here, x is a positive integer with at least two digits and no leading zeros, and k is an integer satisfying $1 \leq k \leq$ (number of digits of x). To evaluate $f(x, k)$, Alice deletes the k-th digit of x from the right and obtains an integer y (y may contain leading zeros). Then we define $f(x, k) = x - y$. For example $f(1234, 2) = 1234 - 124 = 1110$ because deleting the 2nd digit from the right of 1234 results in 124.

Alice wants to know how many ways she can choose x and k such that $f(x, k) = y$, for a given integer y. For any integer z, define $g(z)$ as the number of pairs (x, k) satisfying $f(x, k) = z$.

For example, $g(10) = 11$ because there are 11 valid pairs (x, k) such that $f(x, k) = 10$:

- For $k = 1$, the only valid pair is $(11, 1)$.
- For $k = 2$, the pairs are $(10, 2), (11, 2), (12, 2), \ldots, (19, 2)$.

You are given a number a_0 with n digits (may include leading zeros). You need to answer q queries.
In the i-th query, you will be given a position p_i and a digit d_i. You can find a_i by updating the p_i-th digit of a_{i-1} (from the right) to d_i. You need to output the value of $g(a_i)$ modulo 998,244,353.

Input

The first line of the input contains a single integer n ($1 \leq n \leq 10^6$) — the number of digits in the initial number a_0.

The second line of the input contains a string of n digits (each digit is between '0' and '9', inclusive) — the initial number a_0.

The third line contains a single integer q ($1 \leq q \leq 10^6$) — the number of queries.

Each of the next q lines contains two space-separated integers p_i and d_i ($1 \leq p_i \leq n$, $0 \leq d_i \leq 9$) — the position (from the right) and the new digit for the i-th query.

Output

For each query, output one integer in a line — the value of $g(a_i) \mod 998{,}244{,}353$ in a new line.

Sample Input	Sample Output
3 010 4 2 1 2 0 1 7 3 2	11 0 0 2

Note

In the sample test case, a_1, a_2, a_3, and a_4 are 010, 000, 007, and 207, respectively.

As explained in the statement, $g(10) = 11$.
$g(0) = 0$ because there is no valid pair (x, k) that produces $f(x, k) = 0$.
$g(7) = 0$ because there is no valid pair (x, k) that produces $f(x, k) = 7$. Note that $(7, 1)$ is not valid since x

must have at least two digits.
It can be shown that $g(207) = 2$.

Problem B. Your Next Line Is, "What A Cool Problem!"

Joseph loves to predict what his opponent is going to say next. However, he is confident that no one else can do the same to him. So he challenges Caesar to a game of **Hangman**.

In this game, there are two roles:

- The **setter** chooses a hidden word of length l.
- The **guesser** tries to discover this word.

The game is played in turns. In each turn, the guesser selects a single letter. After every guess, the setter must give a **response**:

- If the letter appears in the hidden word, the setter must reveal all positions where it appears.
- If the letter does not appear in the word, it is counted as an **incorrect attempt**.

The guesser wins if he discovers **all letters** of the hidden word before making n incorrect attempts. Otherwise, the setter wins.

Turn	Guessed Letter	Correct?	Current State	Incorrect Attempts Remaining
0	–	–	`______`	2
1	`a`	×	`______`	1
2	`e`	✓	`_e__e_`	1
3	`r`	✓	`_e__er`	1
4	`t`	✓	`_etter`	1
5	`l`	×	`_etter`	0

Table 2: Hangman game example for the hidden word `better`.

Caesar believes that Joseph, acting as the setter, may cheat by changing the hidden word during the game. To prevent this, Caesar introduces the following rules:

- Before the game starts, Joseph must reveal an **alphabet** of size a and a **vocabulary** of size v.
  - The alphabet is the set of allowed letters.
  - The vocabulary is a set of words, each using only letters from that alphabet.
- To claim victory at the end of the game, Joseph must show one word from the vocabulary and prove that every response he gave during the game was correct for that word.

Now, Joseph will only play the game if, under these rules, he can guarantee victory. Given the values of a, v, l, and n, determine whether Joseph will play the game.

Input

The first line of the input contains a single integer t ($1 \le t \le 2 \times 10^5$) — the number of test cases.

Each test case consists of a single line containing four space-separated integers a, v, l, and n ($1 \le a, v, l, n \le 26, v \le a^l$) — the size of the alphabet, the size of the vocabulary, the length of the hidden word, and the maximum number of incorrect attempts allowed, respectively.

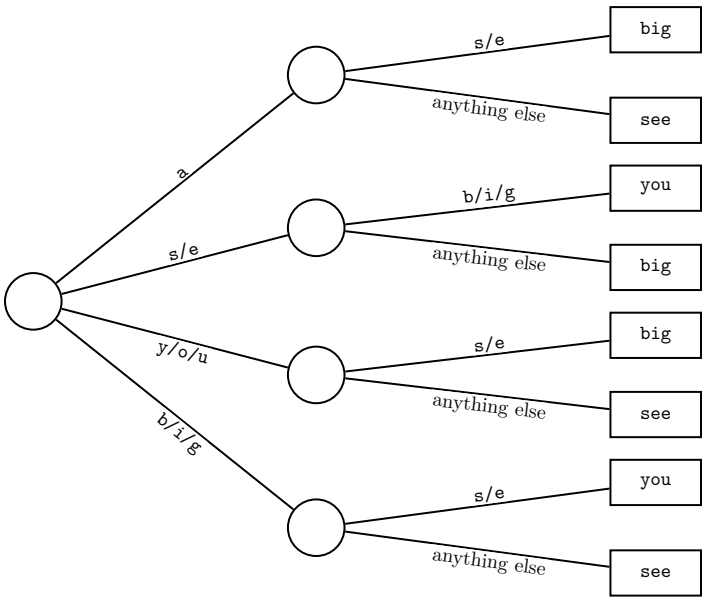
Output

For each test case, output a single string in a line — "YES" if Joseph will play the game, "NO" otherwise.

Sample Input	Sample Output
5 9 3 3 2 11 2 5 1 10 5 4 3 26 1 5 3 26 26 26 26	YES YES YES NO NO

Note

In the first test case, Joseph can choose the alphabet {a, e, i, o, u, b, g, s, y} and choose the vocabulary {you, see, big}. Based on Caesar's guesses, Joseph can always change his hidden word to show Caesar that none of his two guesses was correct, thus win the game. The following diagram illustrates Joseph's strategy:



In the second test case, Joseph can choose the alphabet {a, b, c, d, e, f, g, h, i, j, k} and choose the vocabulary {abcdek, fghijk}. If Caesar guesses k, Joseph will reveal its position as the last letter of the hidden word. No matter which letter Caesar guesses next, Joseph can always find one word in the vocabulary that does not contain that letter, change his hidden word to that word, and win the game. If Caesar does not guess k on his first attempt, Joseph can claim victory immediately.

In the third test case, one vocabulary that can allow Joseph to guarantee victory is {king, ping, ring, sing, wing}.

In the fourth test case, Joseph can keep only one word in the vocabulary. Whatever that word is, Caesar would already know it before the game begins, so he can always guess its letters correctly without making any incorrect attempts.

It can be proven that no alphabet or vocabulary can guarantee victory for Joseph in the fifth test case.

Problem C. Least Compatible Ancestor

You are given a rooted tree with n nodes numbered from 1 to n and the root of the tree is node 1. Find the number of ways, modulo 998244353, to assign a value a_u to each node u such that $1 \leq a_u \leq n$ and $a_{\text{lca}(i,j)} \neq \text{lca}(a_i, a_j)$ is satisfied over all pairs $1 \leq i < j \leq n$, where $\text{lca}(x, y)$ denotes the lowest common ancestor of nodes x and y in the tree.

Input

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

The first line of each test case contains a single integer n ($1 \leq n \leq 20$) — the number of nodes in the tree.

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.

It is guaranteed that the sum of n over all test cases does not exceed 40.

Output

For each test case, output a single integer — the number of ways to assign values to all nodes satisfying the given condition, modulo 998244353.

Sample Input	Sample Output
2 2 1 2 3 1 3 2 1	1 8

Note

In the first test case, there is only one valid way to assign values to the nodes: $a_1 = 2$ and $a_2 = 1$. In this case, the condition is satisfied because for the pair $(1, 2)$, $a_{\text{lca}(1,2)} = a_1 = 2$ and $\text{lca}(a_1, a_2) = \text{lca}(2, 1) = 1$, so they are not equal.

Problem D. Magical Flower Garden

Ananya is the owner of a magical flower garden. The garden contains n **flowers** in a line numbered left to right from 1 to n. Each flower has a **color** that can be represented by a non-negative integer. The garden can be represented as an array A of length n, where A_i denotes the color of the i-th flower from the left.

The **saturation** of a color a is defined as popcount(a), which is the number of 1s in the binary representation of a. For example, the saturation of color 14 is 3 since the binary representation of 14 is 1110, which contains three 1s.
To change the color of a flower, Ananya can sprinkle **magical dust** on it. Sprinkling magical dust with value b on a flower changes its color from a to $a|b$, where | denotes the bitwise OR operation.

Ananya loves her garden and wants to keep it beautiful. She defines the beauty of the garden as the sum of scores of all its subarrays. The score of a subarray is defined as the product of its length and the maximum saturation among the colors of the flowers in that subarray.

Formally, the **score** of the subarray $A[l \dots r]$,

$$\text{score}(A[l \dots r]) = (r - l + 1) \cdot \max_{l \le i \le r} \text{popcount}(A_i)$$

The **beauty** of the array A,

$$\text{beauty}(A) = \sum_{l=1}^{n} \sum_{r=l}^{n} \text{score}(A[l \dots r])$$

As your team is participating in the ICPC this year and is known as excellent problem solvers, Ananya has appointed you as the caretaker of her garden. Everyday, she gives you exactly one **task** to perform on her flowers. The tasks can be of one of the following two types:

- **Type 1:** She gives you two numbers p and q ($0 \le p \le 30, 1 \le q \le 10^9$). You have to find the flower whose saturation is exactly p and sprinkle magical dust with value q on it.
  If the garden currently contains multiple flowers with saturation p, pick the **leftmost** one. If there's currently no flower with saturation p, do nothing.
  Formally, find the smallest index i such that popcount(A_i) $= p$ and if such an i exists, perform, $A_i := A_i|q$.
- **Type 2:** She wants to know how beautiful the garden is. You have to output the current beauty of the garden.

Your goal is to perform each daily task carefully.

Input

The first line of the input contains a single integer t ($1 \le t \le 10^3$) — the number of test cases.

The first line of each test case contains two space-separated integers n and d ($1 \le n, d \le 10^5$) — the number of flowers in the garden and the number of days respectively.

The second line of each test case contains n space-separated integers $A_1, A_2, \dots, A_n$ ($0 \le A_i \le 10^9$) — the initial colors of the flowers.

Each of the next d lines represents the task for a particular day, given sequentially. The tasks can be of one of the following two formats:

- `"1 p q"`: Find the leftmost flower whose current saturation is equal to p and if such a flower exists, sprinkle magical dust with value q on it.

- "2": Output the current beauty of the garden.

It is guaranteed that the sum of $(n + d)$ over all test cases does not exceed $2 \cdot 10^5$. It is also guaranteed that each test case contains at least one task of type 2.

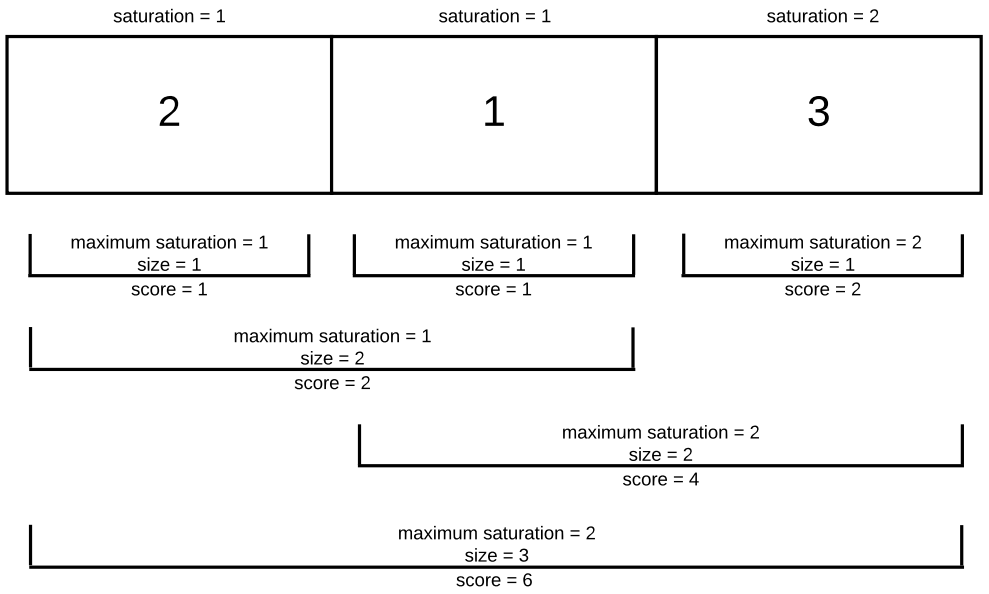
Output

For each task of type 2, print a single integer in a line — the current beauty of the garden.

Sample Input	Sample Output
2 3 5 2 1 3 2 1 2 4 2 1 1 1 2 9 6 8 3 4 6 1 7 3 8 2 2 1 2 5 2 1 2 6 1 3 8 2	16 22 25 443 467 547

Note

The following diagram illustrates the initial state of the garden for the first sample test case, along with the scores of each subarray:



The initial beauty of the garden is $1 + 1 + 2 + 2 + 4 + 6 = 16$.

Problem E. The Perfect View

Alice is an ambitious entrepreneur who wants to open the most popular **cafe** in her city. She has scouted N potential locations for her cafe. The city's main square, where all these locations are, is famous for its M beautiful **landmarks**.
The potential cafe locations are represented as points $C_1, C_2, \ldots, C_N$ in a 2D plane, and the landmarks are represented as points $L_1, L_2, \ldots, L_M$ in the same plane. All points are represented by their Cartesian coordinates.

Alice has a unique theory about what makes a cafe's view "perfect". For any given cafe location, the triangular area formed by the cafe itself and any two distinct landmarks creates a special **scenic zone**.
For any point P, its **scenic value** with respect to a cafe location C_i is defined as the number of scenic zones that originate from C_i and contain P strictly inside them.

The higher the scenic value, the higher the customer satisfaction! Therefore, Alice wants to choose a cafe location and place a table at a position that has the maximum scenic value with respect to that cafe location. However, to achieve this, Alice needs your help. Your task is to find the maximum scenic value that can be achieved among all cafe locations C_i $(1 \leq i \leq N)$ and all possible points P in the plane.

Input

The first line contains an integer T $(1 \leq T \leq 5000)$ — the number of test cases.

The first line of each test case contains two space-separated integers N and M $(1 \leq N \leq 400,\ 2 \leq M \leq 400)$ — the number of cafe locations and the number of landmarks.

The next N lines each contain two space-separated integers x_i and y_i — coordinates of cafe location C_i.

The next M lines each contain two space-separated integers x_j and y_j — coordinates of landmark L_j.

For each test case, it is guaranteed that:

- All coordinates are between -10^9 and 10^9, inclusive.
- The cafe locations and landmarks are all distinct points.
- For any cafe location C_i and any two landmarks L_j, L_k, the three points are not collinear.

It is guaranteed that neither the sum of N over all test cases nor the sum of M over all test cases exceed 20000.

Output

For each test case, output a single integer in a line — the maximum possible scenic value.

Sample Input	Sample Output
2 1 4 -1 4 -7 7 -8 -1 -5 3 2 6 2 3 0 0 1 1 1 0 0 1 -1 -1	3 2

Problem F. Over Counting

Genos is obsessed with counting and spends all his time doing it. To test his abilities, master Saitama gave him a binary string a and asked him to calculate a function $g(a)$.

For any array b of length m, let $f(b)$ denote an array c of length m such that:

$$\text{for each } i\,(1 \le i \le m), \quad c_i \text{ is the number of indices } j \text{ such that } 1 \le j < i \text{ and } b_j > b_i.$$

If b is a binary string, consider its characters as integers (i.e., '0' as 0 and '1' as 1).

Now, $g(b)$ is defined as the sum of all elements of the array $f(f(b))$.

You are another disciple of master Saitama. To prove your superiority over Genos, you decide to calculate the sum of $g(a)$ over **all distinct permutations** of the given binary string a. Since the answer can be very large, output it modulo 998,244,353.

A **permutation** of an array or string is any reordering of its elements.

Two permutations are considered **different** if and only if there exists at least one index i such that the element at position i differs between the two permutations. For example, the permutations $[0, 1, 1]$ and $[0, 1, 1]$ are **not** considered distinct, since swapping identical elements does not create a new array.

Input

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

The first line of each test case contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the length of the string a.

The second line of each test case contains a binary string a of length n (a_i is either '0' or '1').

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single integer in a line — the sum of $g(a)$ over all distinct permutations of a, modulo 998,244,353.

Sample Input	Sample Output
1 4 0101	4

Note

In the first sample test case where $a = 0011$, there are 6 distinct permutations of a. For each permutation a_i, the value of $g(a_i)$ is shown in the following table:

i	a_i	$g(a_i)$
1	$[0, 0, 1, 1]$	0
2	$[0, 1, 0, 1]$	1
3	$[0, 1, 1, 0]$	0
4	$[1, 0, 0, 1]$	2
5	$[1, 0, 1, 0]$	1
6	$[1, 1, 0, 0]$	0

The sum of $g(a_i)$ over all distinct permutations is: $\sum_{i=1}^{6} g(a_i) = 0 + 1 + 0 + 2 + 1 + 0 = 4$.

Problem G. The Matrix

You are Neo, a programmer who has discovered that the world we live in is nothing but a two-dimensional Matrix. With the help of some friends, you have escaped the Matrix. But your friends are still trapped inside. The Agents have assigned a **danger level** to the Matrix, and you must minimize it to save your friends.

The Matrix has n rows and m columns, where each element is a non-negative integer that can be represented by 30 bits. The danger level of a row is defined as the **bitwise XOR** of all numbers in that row, and the danger level of a column as the **bitwise XOR** of all numbers in that column. The total danger level of the Matrix is the **sum** of the danger levels of all rows and all columns in the Matrix.

Formally, let's define the element in the i-th row and j-th column ($1 \le i \le n, 1 \le j \le m$) of the Matrix as $M_{i,j}$.
The danger level of the i-th row, $r_i = M_{i,1} \oplus M_{i,2} \oplus \cdots \oplus M_{i,m}$.
The danger level of the j-th column, $c_j = M_{1,j} \oplus M_{2,j} \oplus \cdots \oplus M_{n,j}$.
The total danger level of the Matrix $= (r_1 + r_2 + \cdots + r_n) + (c_1 + c_2 + \cdots + c_m)$.

With your newfound abilities, you can alter **at most one** element of the Matrix, **replacing** it with any non-negative integer to reduce the danger level.

Find the **minimum total danger level** achievable.

Input

The first line of the input 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 space-separated integers n and m ($1 \le n, m \le 5 \times 10^5$) — the number of rows and columns in the Matrix.

The next n lines each contains m space-separated integers ($0 \le M_{i,j} < 2^{30}$) — the elements of the Matrix. The j-th integer in the i-th of these lines denotes $M_{i,j}$.

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

Output

For each test case, output a single integer in a line — the **minimum total danger level** achievable.

Sample Input	Sample Output
4 2 3 13 9 24 11 20 27 3 3 5 5 5 5 5 5 5 5 5 1 4 1 2 3 4 4 3 0 0 0 0 0 0 0 0 0 0 0 0	14 20 6 0

Note

In the first test case, the danger level of the first row is 28 and the danger level of the second row is 4. The danger levels of the columns are 6, 29, and 3 respectively. This makes the total danger level 70.
By replacing 9 with 21, the danger level can be brought down to 14. It can be proven that this is the minimum total danger level that can be achieved.

Problem H. Chemical Reaction

The Bangladesh Advanced Pharmaceutical Systems (BAPS) is working with the Institute for Chemical Process Control (ICPC) to study how chemicals react inside a special chamber.

The chamber works in a very simple way. At the start, the scientists put some basic chemicals into it.
Each second, the chemicals in the chamber are allowed to react spontaneously. Whenever two chemicals x and y are capable of reacting with one another, they produce a new chemical z. The chamber's conditions prevent x and y from reacting to completion, so both remain present and available for further reactions in subsequent seconds, while the newly formed chemical z is added to the mixture.
Each reaction takes exactly one second to complete, and multiple reactions can occur simultaneously in the chamber during the same second, as long as the required chemicals are present.

At the beginning, BAPS places n different chemicals into the chamber. The scientists have identified m pairs of chemicals that can react with one another, along with the resulting chemical produced from each reaction. Each reaction is described by three integers x, y, and z, indicating that chemicals x and y can react to produce chemical z.

Now, the scientists are interested in determining how many different chemicals can be present in the chamber after 10^{100} seconds. Since they cannot wait that long, they have asked you to figure it out.

Input

The first line of the input contains a single integer t ($1 \le t \le 500$) — the number of test cases.

The first line of each test case contains two space-separated integers n and m ($1 \le n, m \le 1000$) — the number of different chemicals initially placed in the chamber and the number of reaction rules, respectively.

The second line of each test case contains n space-separated integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le 10^9$) — the types of chemicals initially placed in the chamber. It is guaranteed that all n initial chemicals are distinct.

The next m lines each contain three space-separated integers x, y, and z ($1 \le x, y, z \le 10^9$, $x < y, x \ne z, y \ne z$) — indicating that chemicals x and y can react to produce chemical z. It is guaranteed that all m reaction rules are distinct.

Output

For each test case, output a single integer in a line — the total number of different chemicals that can be present in the chamber after 10^{100} seconds.

Sample Input	Sample Output
2 2 3 1 4 1 4 2 2 4 3 1 3 6 2 2 1 1000000000 1 1000000000 500000000 500000000 1000000000 999999999	5 4