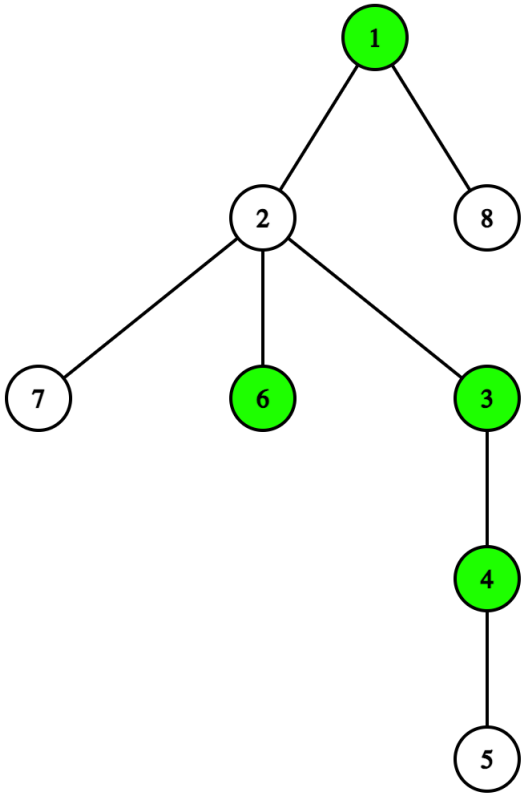
A. Iris and Game on the Tree

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

Iris has a tree rooted at vertex 1. Each vertex has a value of `0` or `1`.

Let's consider a leaf of the tree (the vertex 1 is never considered a leaf) and define its *weight*. Construct a string formed by the values of the vertices on the path starting at the root and ending in this leaf. Then the weight of the leaf is the difference between the number of occurrences of `10` and `01` substrings in it.

Take the following tree as an example. Green vertices have a value of `1` while white vertices have a value of `0`.



- Let's calculate the weight of the leaf 5: the formed string is `10110`. The number of occurrences of substring `10` is 2, the number of occurrences of substring `01` is 1, so the difference is $2 - 1 = 1$.
- Let's calculate the weight of the leaf 6: the formed string is `101`. The number of occurrences of substring `10` is 1, the number of occurrences of substring `01` is 1, so the difference is $1 - 1 = 0$.

The *score* of a tree is defined as the number of leaves with non-zero weight in the tree.

But the values of some vertices haven't been decided and will be given to you as `?`. Filling the blanks would be so boring, so Iris is going to invite Dora to play a game. On each turn, one of the girls chooses any of the remaining vertices with value `?` and changes its value to `0` or `1`, **with Iris going first**. The game continues until there are no vertices with value `?` left in the tree. Iris aims to maximize the score of the tree, while Dora aims to minimize that.

Assuming that both girls play optimally, please determine the final score of the tree.

Input

Each test consists of multiple test cases. The first line contains a single integer t ($1 \leq t \leq 5 \cdot 10^4$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n ($2 \leq n \leq 10^5$) — the number of vertices in the tree.

The following $n - 1$ lines each contain two integers u and v ($1 \leq u, v \leq n$) — denoting an edge between vertices u and v.

It's guaranteed that the given edges form a tree.

The last line contains a string s of length n. The i-th character of s represents the value of vertex i. It's guaranteed that s only contains characters `0`, `1` and `?`.

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

Output

For each test case, output a single integer — the final score of the tree.

Standard Input	Standard Output
6 4 1 2 1 3 4 1 0101 4 1 2 3 2 2 4 ???0 5 1 2 1 3 2 4 2 5 ?1?01 6 1 2 2 3 3 4 5 3 3 6 ?0???? 5 1 2 1 3 1 4 1 5 11?1? 2 2 1 ??	2 1 1 2 1 0

Note

In the first test case, all the values of the vertices have been determined. There are three different paths from the root to a leaf:

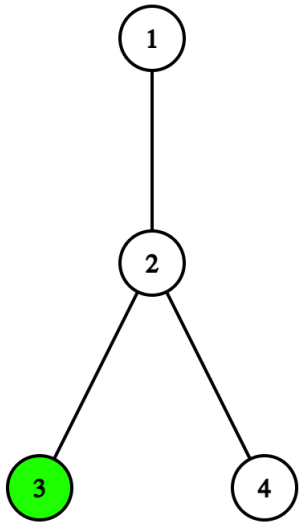
- From vertex 1 to vertex 2. The string formed by the path is $\texttt{01}$, so the weight of the leaf is $0 - 1 = -1$.
- From vertex 1 to vertex 3. The string formed by the path is $\texttt{00}$, so the weight of the leaf is $0 - 0 = 0$.
- From vertex 1 to vertex 4. The string formed by the path is $\texttt{01}$, so the weight of the leaf is $0 - 1 = -1$.

Thus, there are two leaves with non-zero weight, so the score of the tree is 2.

In the second test case, one of the sequences of optimal choices for the two players can be:

- Iris chooses to change the value of the vertex 3 to $\texttt{1}$.
- Dora chooses to change the value of the vertex 1 to $\texttt{0}$.
- Iris chooses to change the value of the vertex 2 to $\texttt{0}$.

The final tree is as follows:



The only leaf with non-zero weight is 3, so the score of the tree is 1. Note that this may not be the only sequence of optimal choices for Iris and Dora.

,

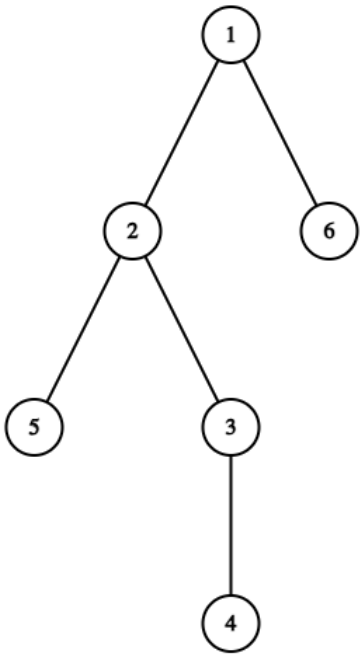
B. Iris and the Tree

```
Input file:     standard input
Output file:    standard output
Time limit:     3 seconds
Memory limit:   256 megabytes
```

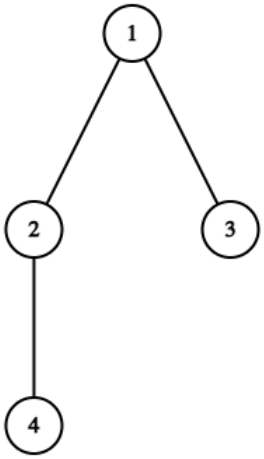
Given a rooted tree with the root at vertex 1. For any vertex i ($1 < i \leq n$) in the tree, there is an edge connecting vertices i and p_i ($1 \leq p_i < i$), with a weight equal to t_i.

Iris does not know the values of t_i, but she knows that $\sum_{i=2}^{n} t_i = w$ and each of the t_i is a **non-negative integer**.

The vertices of the tree are numbered in a special way: the numbers of the vertices in each subtree are consecutive integers. In other words, the vertices of the tree are numbered in the order of a depth-first search.



The tree in this picture satisfies the condition. For example, in the subtree of vertex 2, the vertex numbers are $2, 3, 4, 5$, which are consecutive integers.



The tree in this picture does not satisfy the condition, as in the subtree of vertex 2, the vertex numbers 2 and 4 are not consecutive integers.

We define $\operatorname{dist}(u, v)$ as the length of the simple path between vertices u and v in the tree.

Next, there will be $n-1$ events:

- Iris is given integers x and y, indicating that $t_x = y$.

After each event, Iris wants to know the maximum possible value of $\operatorname{dist}(i, i \bmod n + 1)$ **independently** for each i ($1 \leq i \leq n$). She only needs to know the sum of these n values. Please help Iris quickly get the answers.

Note that when calculating the maximum possible values of $\operatorname{dist}(i, i \bmod n + 1)$ and $\operatorname{dist}(j, j \bmod n + 1)$ for $i \neq j$, the unknown edge weights **may be different**.

Input

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

The first line of each test case contains two integers n and w ($2 \leq n \leq 2 \cdot 10^5$, $0 \leq w \leq 10^{12}$) — the number of vertices in the tree and the sum of the edge weights.

The second line of each test case contains $n-1$ integers $p_2, p_3, \ldots, p_n$ ($1 \leq p_i < i$) — the description of the edges of the tree.

Then follow $n-1$ lines indicating the events. Each line contains two integers x and y ($2 \leq x \leq n$, $0 \leq y \leq w$), indicating that $t_x = y$.

It is guaranteed that all x in the events are distinct. It is also guaranteed that the sum of all y equals w.

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 one line containing $n-1$ integers, each representing the answer after each event.

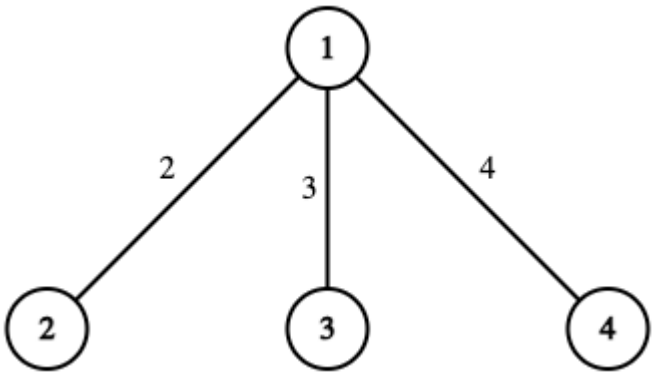
Standard Input	Standard Output
4 2 1000000000000 1 2 1000000000000 4 9 1 1 1 2 2 4 4 3 3 6 100 1 2 3 2 1 6 17 3 32 2 4 4 26 5 21 10 511 1 2 2 4 2 1 1 8 8 3 2 6 16 10 256	2000000000000 25 18 18 449 302 247 200 200 4585 4473 2681 1567 1454 1322 1094 1022 1022

9 128 2 1 5 8 8 64 4 4 7 32	

Note

In the first test case, $\operatorname{dist}(1,2) = \operatorname{dist}(2,1) = t_2 = w = 10^{12}$, so $\operatorname{dist}(1,2) + \operatorname{dist}(2,1) = 2 \cdot 10^{12}$.

In the second test case, the tree after Iris found out all t_x is shown below:



$\operatorname{dist}(1,2) = t_2$, $\operatorname{dist}(2,3) = t_2 + t_3$, $\operatorname{dist}(3,4) = t_3 + t_4$, $\operatorname{dist}(4,1) = t_4$. After the first event, she found out that $t_2 = 2$, so $\operatorname{dist}(1,2) = 2$. At the same time:

- $\operatorname{dist}(2,3)$ is maximized if $t_3 = 7$, $t_4 = 0$. Then $\operatorname{dist}(2,3) = 9$.
- $\operatorname{dist}(3,4)$ and $\operatorname{dist}(4,1)$ are maximized if $t_3 = 0$, $t_4 = 7$. Then $\operatorname{dist}(3,4) = \operatorname{dist}(4,1) = 7$.

Thus, the answer is $2 + 9 + 7 + 7 = 25$.

After the second event, she found out that $t_4 = 4$, then $t_3 = w - t_2 - t_4 = 4$. $\operatorname{dist}(1,2) = 2$, $\operatorname{dist}(2,3) = 2 + 3 = 5$, $\operatorname{dist}(3,4) = 3 + 4 = 7$, $\operatorname{dist}(4,1) = 4$. Thus, the answer is $2 + 5 + 7 + 4 = 18$.

,

C. Eri and Expanded Sets

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

Let there be a set that contains **distinct** positive integers. To expand the set to contain as many integers as possible, Eri can choose two integers $x \neq y$ from the set such that their average $\frac{x+y}{2}$ is still a positive integer and isn't contained in the set, and add it to the set. The integers x and y remain in the set.

Let's call the set of integers *consecutive* if, after the elements are sorted, the difference between any pair of adjacent elements is 1. For example, sets $\{2\}$, $\{2, 5, 4, 3\}$, $\{5, 6, 8, 7\}$ are consecutive, while $\{2, 4, 5, 6\}$, $\{9, 7\}$ are not.

Eri likes consecutive sets. Suppose there is an array b, then Eri puts all elements in b into the set. If after a finite number of operations described above, the set can become consecutive, the array b will be called *brilliant*.

Note that if the same integer appears in the array multiple times, we only put it into the set **once**, as a set always contains distinct positive integers.

Eri has an array a of n positive integers. Please help him to count the number of pairs of integers (l, r) such that $1 \leq l \leq r \leq n$ and the subarray $a_l, a_{l+1}, \ldots, a_r$ is brilliant.

Input

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

The first line of each test case contains a single integer n ($1 \leq n \leq 4 \cdot 10^5$) — length of the array a.

The second line of each test case contains n integers $a_1, a_2, \ldots a_n$ ($1 \leq a_i \leq 10^9$) — the elements of the array a.

It is guaranteed that the sum of n over all test cases doesn't exceed $4 \cdot 10^5$.

Output

For each test case, output a single integer — the number of brilliant subarrays.

Standard Input	Standard Output
6 2 2 2 6 1 3 6 10 15 21 5 6 30 18 36 9 1 1000000000 6 1 1 4 5 1 4 12	3 18 5 1 18 53

70 130 90 90 90 108 612 500 451 171 193 193	

Note

In the first test case, the array $a = [2, 2]$ has 3 subarrays: $[2]$, $[2]$, and $[2, 2]$. For all of them, the set only contains a single integer 2, therefore it's always consecutive. All these subarrays are brilliant, so the answer is 3.

In the second test case, let's consider the subarray $[3, 6, 10]$. We can do operations as follows:

$$\{3,6,10\} \xrightarrow{x=6,y=10} \{3,6,8,10\} \xrightarrow{x=6,y=8} \{3,6,7,8,10\} \xrightarrow{x=3,y=7} \{3,5,6,7,8,10\}$$
$$\xrightarrow{x=3,y=5} \{3,4,5,6,7,8,10\} \xrightarrow{x=8,y=10} \{3,4,5,6,7,8,9,10\}$$

$\{3, 4, 5, 6, 7, 8, 9, 10\}$ is a consecutive set, so the subarray $[3, 6, 10]$ is brilliant.

,

D. Iris and Adjacent Products

Input file: standard input
Output file: standard output
Time limit: 3 seconds
Memory limit: 256 megabytes

Iris has just learned multiplication in her Maths lessons. However, since her brain is unable to withstand too complex calculations, she could not multiply two integers with the product greater than k together. Otherwise, her brain may explode!

Her teacher sets a difficult task every day as her daily summer holiday homework. Now she is given an array a consisting of n elements, and she needs to calculate the product of each two adjacent elements (that is, $a_1 \cdot a_2$, $a_2 \cdot a_3$, and so on). Iris wants her brain to work safely, and in order to do that, she would like to modify the array a in such a way that $a_i \cdot a_{i+1} \leq k$ holds for every $1 \leq i < n$. There are two types of operations she can perform:

1. She can rearrange the elements of the array a in an arbitrary way.
2. She can select an arbitrary element of the array a and change its value to an arbitrary integer from 1 to k.

Iris wants to minimize the number of operations of **type** 2 that she uses.

However, that's completely not the end of the summer holiday! Summer holiday lasts for q days, and on the i-th day, Iris is asked to solve the Math homework for the subarray $b_{l_i}, b_{l_i+1}, \ldots, b_{r_i}$. Help Iris and tell her the minimum number of type 2 operations she needs to perform for each day. Note that the operations are **independent** for each day, i.e. the array b is not changed.

Input

Each test consists of multiple test cases. The first line contains a single integer t ($1 \leq t \leq 5 \cdot 10^4$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains three integers n, q and k ($2 \leq n \leq 10^5$, $1 \leq q \leq 10^5$, $1 \leq k \leq 10^6$) — the length of array b, the number of days, and the upper bound for the multiplication calculation.

The second line of each test case contains n integers $b_1, b_2, \ldots, b_n$ ($1 \leq b_i \leq k$) — the elements of the array b.

Then q lines follow, the i-th of them contains two integers l_i and r_i ($1 \leq l_i < r_i \leq n$) — the boundaries of the subarray on the i-th day.

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

Output

For each test, output a single line containing q integers — the minimum number of operations of type 2 needed for each day.

Standard Input	Standard Output
5 3 1 1	0 0 1

1 1 1 1 3 3 2 10 1 10 9 1 3 2 3 5 4 2 2 2 2 2 2 1 2 2 4 2 5 1 5 6 5 10 3 2 5 10 10 1 1 4 3 6 1 6 2 5 5 6 10 10 10 10 9 8 7 6 5 4 3 2 1 1 10 1 9 1 8 1 7 2 10 3 10 4 10 5 10 3 9 6 8	1 1 2 2 1 1 1 1 0 3 3 4 3 2 2 1 1 2 1

Note

In the first test case, as Iris can always multiply 1 and 1 together, no operations are needed, so the answer is 0.

In the second test case, the first day's homework is $[1, 10, 9]$. Iris can rearrange its elements to get $[9, 1, 10]$, so no operations of type 2 are needed. The second day's homework is $[10, 9]$, and she can change one element to get the array $[1, 9]$, so one operation of type 2 is needed.

,

E. Iris's Full Binary Tree

Input file: standard input
Output file: standard output
Time limit: 4 seconds
Memory limit: 1024 megabytes

Iris likes full binary trees.

Let's define the depth of a rooted tree as the maximum number of **vertices** on the simple paths from some vertex to the root. A full binary tree of depth d is a binary tree of depth d with exactly $2^d - 1$ vertices.

Iris calls a tree a d-*binary* tree if some vertices and edges can be **added** to it to make it a full binary tree of depth d. Note that **any vertex** can be chosen as the root of a full binary tree.

Since performing operations on large trees is difficult, she defines the *binary depth* of a tree as the minimum d satisfying that the tree is d-binary. Specifically, if there is no integer $d \geq 1$ such that the tree is d-binary, the binary depth of the tree is -1.

Iris now has a tree consisting of only vertex 1. She wants to add $n - 1$ more vertices to form a larger tree. She will add the vertices one by one. When she adds vertex i ($2 \leq i \leq n$), she'll give you an integer p_i ($1 \leq p_i < i$), and add a new edge connecting vertices i and p_i.

Iris wants to ask you the binary depth of the tree formed by the first i vertices for each $1 \leq i \leq n$. Can you tell her the answer?

Input

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

The first line of each test case contains a single integer n ($2 \leq n \leq 5 \cdot 10^5$) — the final size of the tree.

The second line of each test case contains $n - 1$ integers $p_2, p_3, \ldots, p_n$ ($1 \leq p_i < i$) — descriptions of all edges of the tree.

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

Output

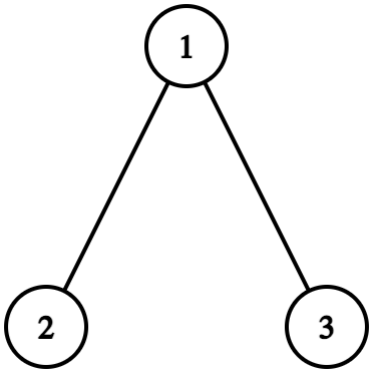
For each test case output n integers, i-th of them representing the binary depth of the tree formed by the first i vertices.

Standard Input	Standard Output
7 3 1 1 6 1 2 3 4 5 7 1 1 3 2 5 1 10 1 1 2 1 4 2 4 5 8 10	1 2 2 1 2 2 3 3 4 1 2 2 3 3 4 4 1 2 2 3 3 3 4 4 5 5 1 2 2 3 3 4 4 4 -1 -1 1 2 2 3 3 4 4 4 4 5 5 5 5 6 6 6 6 6 6 7 1 2 2 3 3 4 4 4 4 5 5 6 6 6 6 6 7 7 7 7 7 8 8 8 8

1 1 3 1 3 2 2 2 6 20 1 1 2 2 4 4 5 5 7 6 8 6 11 14 11 8 13 13 12 25 1 1 3 3 1 5 4 4 6 8 11 12 8 7 11 13 7 13 15 6 19 14 10 23	

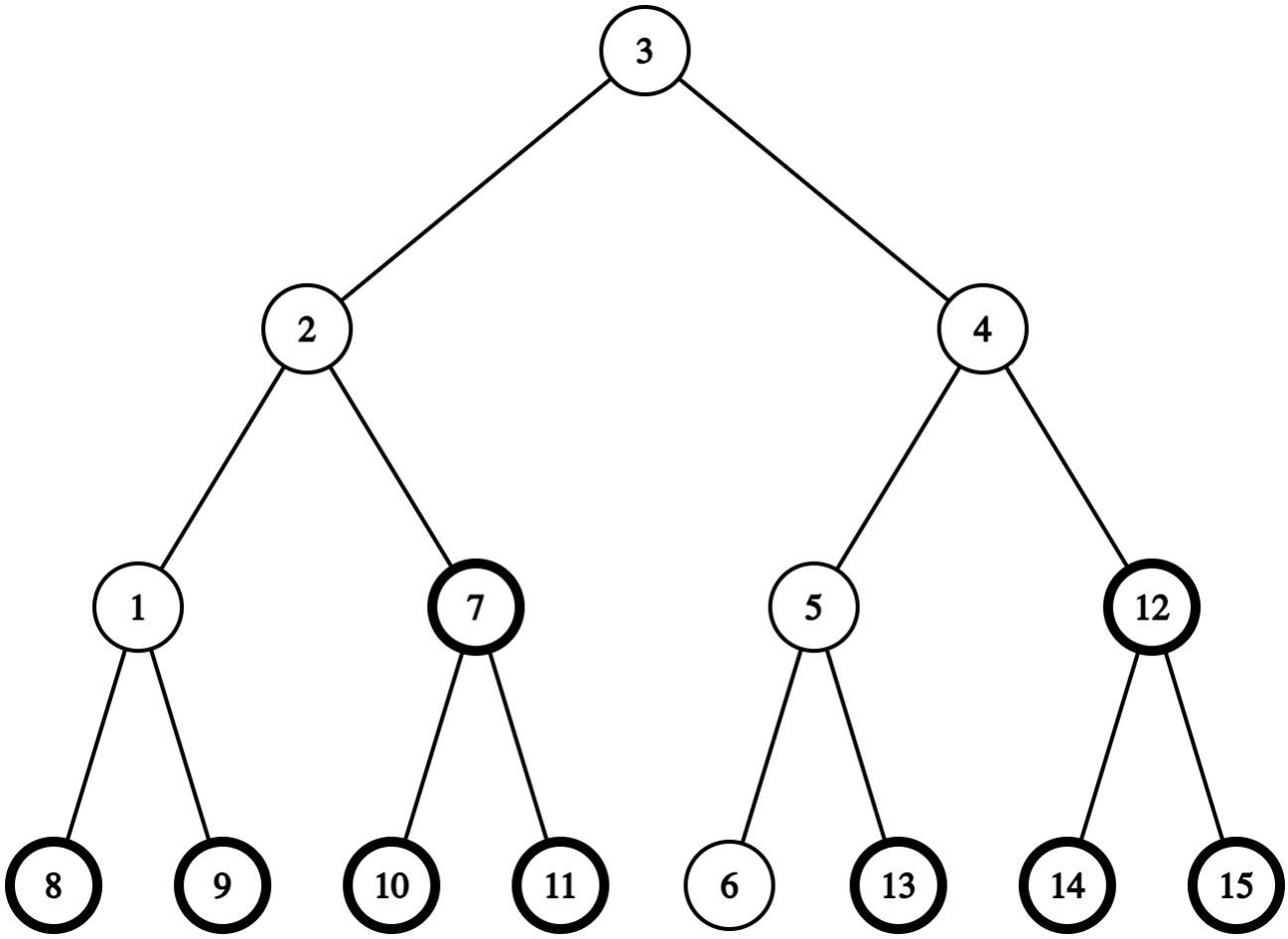
Note

In the first test case, the final tree is shown below:



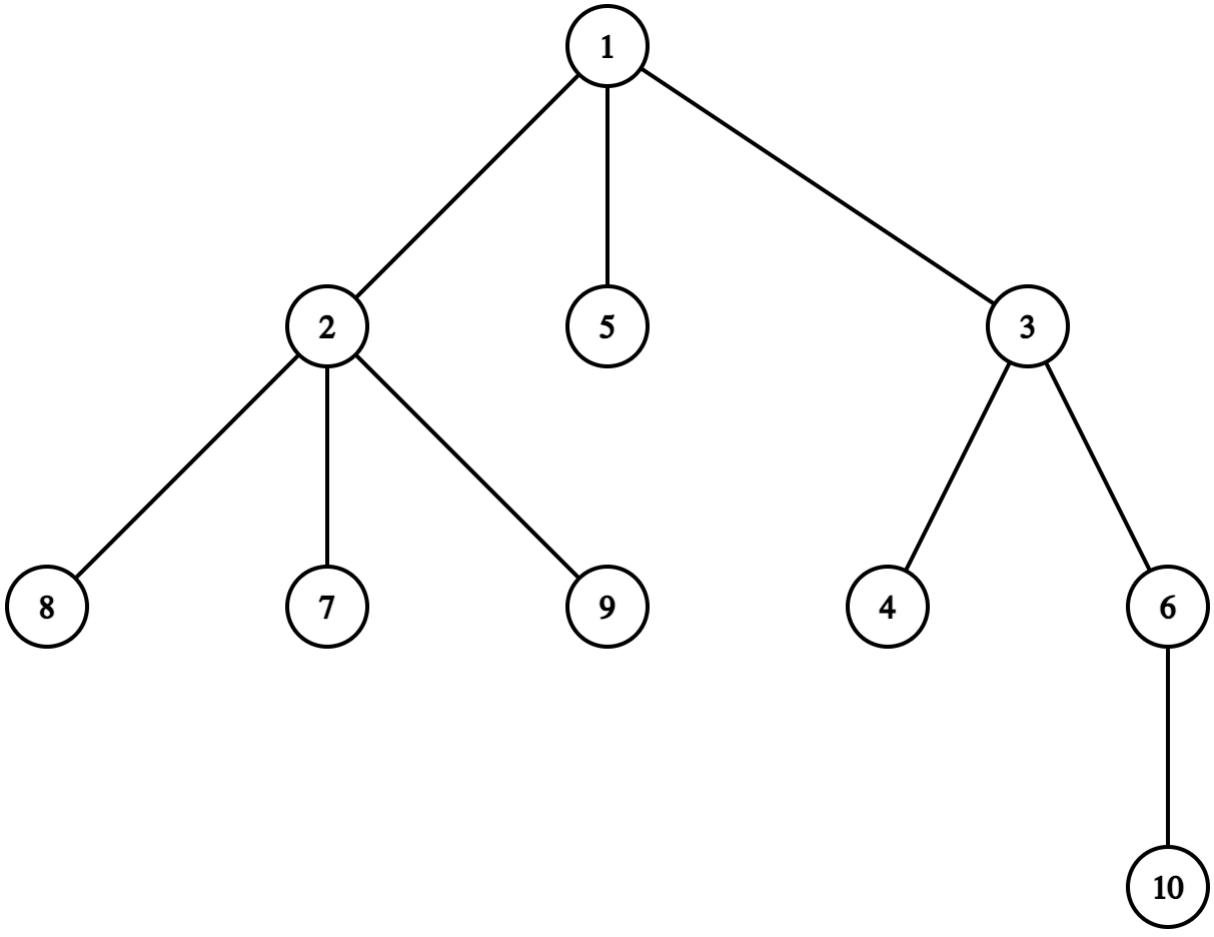
- The tree consisting of the vertex 1 has the binary depth 1 (the tree itself is a full binary tree of depth 1).
- The tree consisting of the vertices 1 and 2 has the binary depth 2 (we can add the vertex 3 to make it a full binary tree of depth 2).
- The tree consisting of the vertices 1, 2 and 3 has the binary depth 2 (the tree itself is a full binary tree of depth 2).

In the second test case, the formed full binary tree after adding some vertices to the tree consisting of n vertices is shown below (bolded vertices are added):



The depth of the formed full binary tree is 4.

In the fifth test case, the final tree is shown below:



It can be proved that Iris can't form any full binary tree by adding vertices and edges, so the binary depth is -1.

,

F. Dora's Paint

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

Sadly, Dora poured the paint when painting the class mural. Dora considers the mural as the matrix b of size $n \times n$. Initially, $b_{i,j} = 0$ for all $1 \le i, j \le n$.

Dora has only two brushes which have two different colors. In one operation, she can paint the matrix with one of two brushes:

- The first brush has color 1 on it and can paint one column of the matrix. That is, Dora chooses $1 \le j \le n$ and makes $b_{i,j} := 1$ for all $1 \le i \le n$;
- The second brush has color 2 on it and can paint one row of the matrix. That is, Dora chooses $1 \le i \le n$ and makes $b_{i,j} := 2$ for all $1 \le j \le n$.

Dora paints the matrix so that the resulting matrix b **contains only** 1 and 2.

For a matrix b, let $f(b)$ denote the minimum number of operations needed to turn the initial matrix (containing only 0) into b. The *beauty* of a matrix b is the number of ways to paint the initial matrix in exactly $f(b)$ operations to turn it into b. If there's no way to turn the initial matrix into b, the beauty of b is 0.

However, Dora made a uniformly random mistake; there's **exactly one** element different in the matrix a given to you from the real matrix b. That is, there is exactly one pair (i, j) such that $a_{i,j} = 3 - b_{i,j}$.

Please help Dora compute the expected beauty of the real matrix b modulo $998\,244\,353$ (all possible n^2 mistakes have equal probability).

Since the size of the matrix is too large, Dora will only tell you the positions of m elements of color 1, and the remaining $n^2 - m$ elements have color 2.

Input

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

The first line of each test case contains two integers n and m ($2 \le n \le 2 \cdot 10^5$, $0 \le m \le \min(10^6, n^2)$) — the size of the matrix and the number of elements of color 1.

Then m lines follow, each containing two positive integers x_i and y_i ($1 \le x_i, y_i \le n$) — denoting that $a_{x_i, y_i} = 1$.

It is guaranteed that if $i \neq j$, then $(x_i, y_i) \neq (x_j, y_j)$.

It is also guaranteed that the sum of n over all test cases does not exceed $4 \cdot 10^5$, and the sum of m over all test cases does not exceed 10^6.

Output

For each test case, output a single integer — the expected beauty of the real matrix b, modulo $998\,244\,353$.

Standard Input	Standard Output

7 2 2 1 1 1 2 2 1 1 1 3 2 1 1 3 3 6 0 5 10 1 1 1 2 1 3 2 1 2 3 5 1 5 2 5 3 5 4 5 5 3 5 1 1 1 3 2 2 3 1 3 3 4 3 1 1 2 3 2 4	1 499122178 665496236 120 79859554 776412275 1

Note

In the first test case, the matrix $a = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix}$. Let's consider changing the element $(1,1)$ to calculate the answer.

It can be proved that the minimum steps to paint the initial matrix into $\begin{bmatrix} 2 & 1 \\ 2 & 2 \end{bmatrix}$ is 3. We can first paint the first row into color 2, then paint the second column into color 1, and finally paint the second row into color 2. The process is listed below:

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 2 & 2 \\ 0 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 2 & 1 \\ 2 & 2 \end{bmatrix}$$

It can be proved that this is the only way to paint the matrix in 3 steps. So the beauty of the matrix $\begin{bmatrix} 2 & 1 \\ 2 & 2 \end{bmatrix}$ is 1. Similarly, if any other element of the matrix is changed, the beauty is always 1, so the expected beauty of the real matrix b is 1.

In the second test case, the matrix $a = \begin{bmatrix} 1 & 2 \\ 2 & 2 \end{bmatrix}$. Let's consider changing the element $(2, 2)$ to calculate the answer.

It can be proven that it's impossible to paint the initial matrix into $\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$, so its beauty is 0. If any other element of the matrix is changed, the beauty is always 2, so the expected beauty is $\frac{0+2+2+2}{4} = \frac{6}{4} \equiv 499\,122\,178 \pmod{998\,244\,353}$.