

Note: These statements are translated by Deepseek.

T1-Edit

Problem Description

Little M has two binary strings, s_1 and s_2 , each of length n with character set $\{0, 1\}$.

Little M wants to maximize the number of positions where the corresponding characters in the two strings are the same, i.e., the number of indices i ($1 \leq i \leq n$) where $s_{1,i} = s_{2,i}$. To achieve this, Little M has a string editing tool that allows swapping two adjacent characters in a string. However, to maintain the recognizability of the strings, certain characters in each string are marked as non-swappable. Little M can use the tool to perform multiple swaps on either s_1 or s_2 , where swappable characters can be swapped any number of times.

Now, Little M wants to know the maximum number of positions where the characters can be made the same after using the editing tool.

Input Format

This problem contains multiple test cases.

The first line of input contains an integer T , representing the number of test cases.

This is followed by T test cases, each formatted as follows:

- The first line contains an integer n , the length of the strings.
- The second line contains a binary string s_1 of length n .
- The third line contains a binary string s_2 of length n .
- The fourth line contains a binary string t_1 of length n , where $t_{1,i} = 1$ indicates that $s_{1,i}$ is swappable, and $t_{1,i} = 0$ indicates it is not.
- The fifth line contains a binary string t_2 of length n , where $t_{2,i} = 1$ indicates that $s_{2,i}$ is swappable, and $t_{2,i} = 0$ indicates it is not.

Output Format

For each test case, output one line containing an integer, representing the maximum number of positions where the characters can be made the same.

Sample Input #1

```
1
6
011101
111010
111010
101101
```

Sample Output #1

Explanation for Sample 1

Initially, $s_1 = 011101$, where the 4th and 6th characters cannot be swapped; $s_2 = 111010$, where the 2nd and 5th characters cannot be swapped.

One possible sequence of operations:

1. Swap $s_{1,1}$ and $s_{1,2}$ to get $s_1 = 101101$.
2. Swap $s_{1,2}$ and $s_{1,3}$ to get $s_1 = 110101$.
3. Swap $s_{2,3}$ and $s_{2,4}$ to get $s_2 = 110110$.

Now, the first 4 positions of s_1 and s_2 are identical. It can be proven that no better solution exists, so the output is 4.

Data Range

For all test cases, it is guaranteed that: $1 \leq T \leq 10$, $1 \leq n \leq 10^5$.

Test Case ID	$n \leq$	Special Properties
1 ~ 4	10	None
5, 6	10^3	A
7, 8	10^5	A
9, 10	10^3	B
11, 12	10^5	B
13, 14	10^3	C
15, 16	10^5	C
17, 18	10^3	None
19, 20	10^5	None

- **Special Property A:** All characters in s_1 are the same.
- **Special Property B:** $t_1 = t_2$.
- **Special Property C:** Each of t_1 and t_2 has exactly one 0.

T2-assign

Problem Description

Little F has n variables $x_1, x_2, \dots, x_n$. Each variable can take an integer value between 1 and v .

Little F adds $n - 1$ binary constraints between these variables, where the i -th constraint ($1 \leq i \leq n - 1$) is: If $x_i = a_i$, then x_{i+1} must be b_i , where a_i and b_i are integers between 1 and v . If $x_i \neq a_i$, the constraint imposes no restriction on x_{i+1} . Additionally, Little F adds m unary constraints, where the j -th constraint ($1 \leq j \leq m$) is: $x_{c_j} = d_j$.

Little F remembers all c_j and d_j values but forgets all a_i and b_i values. However, Little F knows that there exists at least one assignment of values to the variables that satisfies all constraints.

Now, Little F wants to know the number of possible combinations of a_i and b_i ($1 \leq i \leq n - 1$) such that at least one valid assignment exists. Since the number can be large, Little F asks you to output the result modulo $10^9 + 7$.

Input Format

This problem contains multiple test cases.

The first line of input contains an integer T , the number of test cases.

This is followed by T test cases, each formatted as follows:

- The first line contains three integers n, m, v , representing the number of variables, the number of unary constraints, and the upper bound of variable values.
- The next m lines each contain two integers c_j, d_j , describing a unary constraint.

Output Format

For each test case, output one line containing an integer, the number of valid combinations modulo $10^9 + 7$.

Sample Input #1

```
3
2 1 2
1 1
2 2 2
1 1
2 2
2 2 2
1 1
1 2
```

Sample Output #1

```
4
3
0
```

Explanation for Sample 1

- For the first test case, all possible (a_1, b_1) combinations $(1, 1), (1, 2), (2, 1), (2, 2)$ satisfy the constraints. For example, $(a_1, b_1) = (1, 1)$ allows $(x_1, x_2) = (1, 1)$, while $(a_1, b_1) = (2, 2)$ allows $(x_1, x_2) = (1, 1)$ or $(1, 2)$.
- For the second test case, only $(x_1, x_2) = (1, 2)$ is valid, so $(a_1, b_1) = (1, 1)$ is invalid, and the other three combinations are valid.
- For the third test case, no assignment satisfies both $x_1 = 1$ and $x_1 = 2$, so no valid (a_1, b_1) exists.

Data Range

For all test cases, it is guaranteed that:

- $1 \leq T \leq 10$,
- $1 \leq n \leq 10^9, 1 \leq m \leq 10^5, 2 \leq v \leq 10^9$,
- For each j ($1 \leq j \leq m$), $1 \leq c_j \leq n, 1 \leq d_j \leq v$.

Test Case ID	$n \leq$	$m \leq$	$v \leq$	Special Properties
1, 2	6	6	2	None
3	9	9	2	None
4, 5	12	12	2	None
6	10^3	1	10^3	None
7	10^5	1	10^5	None
8, 9	10^9	1	10^9	None
10	10^3	10^3	10^3	A
11	10^4	10^4	10^4	A
12	10^5	10^5	10^5	A
13	10^4	10^3	10^4	B
14	10^6	10^4	10^6	B
15, 16	10^9	10^5	10^9	B
17	10^4	10^3	10^4	None
18	10^6	10^4	10^6	None
19, 20	10^9	10^5	10^9	None

- **Special Property A:** $m = n$, and for each j ($1 \leq j \leq m$), $c_j = j$.
- **Special Property B:** $d_j = 1$ for all j .

T3-traverse

Problem Description

Little Q, a beginner in competitive programming, is learning about tree traversal in graph theory. A tree with n nodes and $n - 1$ edges is initially unmarked. The traversal process is as follows:

1. Select a node s as the starting node and mark it.
2. Suppose the current node is u . Find any adjacent unmarked node v , set v as the current node, and mark it. Repeat this step.
3. If no adjacent unmarked nodes exist for u , and $u = s$, the traversal ends. Otherwise, backtrack to the previous node and repeat step 2.

For example, in the following tree, one possible traversal is:

- Start at node 1 and mark it.
- Node 2 is adjacent and unmarked; mark it and set it as current.
- Node 3 is adjacent and unmarked; mark it and set it as current.
- No adjacent unmarked nodes for 3; backtrack to 2.
- Node 4 is adjacent and unmarked; mark it and set it as current.
- No adjacent unmarked nodes for 4; backtrack to 2.
- No adjacent unmarked nodes for 2; backtrack to 1.
- No adjacent unmarked nodes for 1; traversal ends.

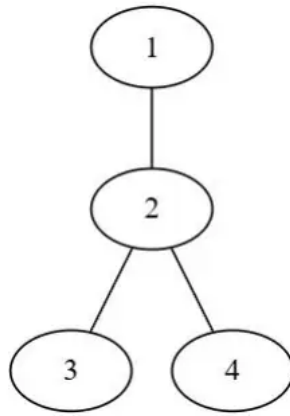


图 1: 样例 1 中的树

洛谷

As a creative student, Little Q extends this traversal to edges. Two edges are adjacent if they share a common node. Initially, all edges are unmarked. The edge-based traversal is as follows:

1. Select an edge b as the starting edge and mark it.
2. Suppose the current edge is e . Find any adjacent unmarked edge f , set f as the current edge, and mark it. Repeat this step.
3. If no adjacent unmarked edges exist for e , and $e = b$, the traversal ends. Otherwise, backtrack to the previous edge and repeat step 2.

For example, in the same tree, one possible edge traversal is:

- Start at edge $\{1, 2\}$ and mark it.
- Edge $\{2, 3\}$ is adjacent and unmarked; mark it and set it as current.
- Edge $\{2, 4\}$ is adjacent and unmarked; mark it and set it as current.
- No adjacent unmarked edges for $\{2, 4\}$; backtrack to $\{2, 3\}$.
- No adjacent unmarked edges for $\{2, 3\}$; backtrack to $\{1, 2\}$.
- No adjacent unmarked edges for $\{1, 2\}$; traversal ends.

Little Q observes that this edge traversal generates a new tree where each edge is a node, and new edges connect nodes e and f if f was traversed from e

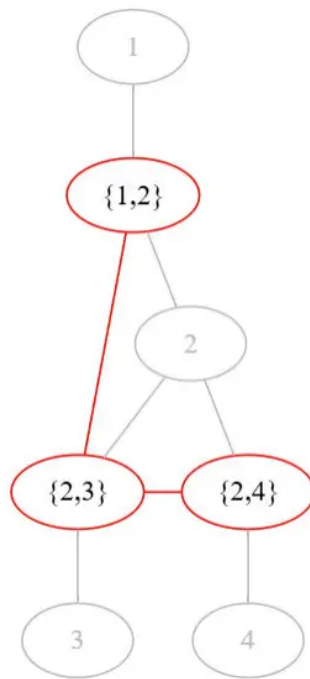


图 2: 一种可能的新树

洛谷

Now, Little Q selects k critical edges and wants to know how many distinct new trees can be generated by starting the traversal from any critical edge. Two trees are distinct if there exists at least one pair of nodes connected in one tree but not the other.

Output the result modulo $10^9 + 7$.

Input Format

This problem contains multiple test cases.

The first line contains two integers c, T , representing the test point ID and the number of test cases. In samples, c indicates the data range matches test point c .

This is followed by T test cases, each formatted as follows:

- The first line contains two integers n, k , the number of nodes and the number of critical edges.
- The next $n - 1$ lines describe the tree edges, each with two integers u_i, v_i .
- The next line contains k integers $e_1, e_2, \dots, e_k$, the indices of the critical edges (all distinct).

Output Format

For each test case, output one line containing an integer, the result modulo $10^9 + 7$.

Sample Input #1

```
1 1
4 1
1 2
2 3
2 4
1
```

Sample Output #1

2

Explanation for Sample 1

Two possible new trees:

1. Connect $\{1, 2\}$ to $\{2, 3\}$, and $\{2, 3\}$ to $\{2, 4\}$.
2. Connect $\{1, 2\}$ to $\{2, 4\}$, and $\{2, 4\}$ to $\{2, 3\}$.

Data Range

For all test cases, it is guaranteed that:

- $1 \leq T \leq 10$,
- $2 \leq n \leq 10^5$,
- $1 \leq k < n$,
- The input describes a valid tree,
- All critical edge indices are distinct.

Test Case ID	$n \leq$	k	Special Properties
1 ~ 3	≤ 5	≤ 1	None
4 ~ 6	10^5	≤ 1	None
7 ~ 10	10^5	≤ 2	None
11, 12	≤ 500	≤ 8	None
13, 14	$\leq 10^2$	$< n$	None
15	≤ 500	$< n$	None
16, 17	10^5	≤ 500	None
18	10^5	$< n$	A
19 ~ 21	10^5	$< n$	B
22, 23	2×10^4	$< n$	None
24, 25	10^5	$< n$	None

- **Special Property A:** For each i ($1 \leq i \leq n - 1$), $u_i = i$, $v_i = i + 1$.
- **Special Property B:** For each i ($1 \leq i \leq n - 1$), $u_i = 1$, $v_i = i + 1$.

T4-Ica

Problem Description

One day, Little S and her friend Little N were studying a tree with n nodes.

This is a rooted tree with the root at node 1. The depth of a node u , dep_u , is defined as the number of nodes on the simple path from u to 1.

Additionally, $\text{LCA}^*(l, r)$ is defined as the lowest common ancestor of all nodes with IDs in $[l, r]$, i.e., the deepest node that is an ancestor of all nodes $l, l + 1, \dots, r$.

Little N poses q queries. In each query, Little N provides three parameters l, r, k , and wants to know the maximum depth of $\text{LCA}^*(l', r')$ for any continuous subinterval $[l', r']$ of $[l, r]$ with length $\geq k$. Formally, the answer is:

$$\max_{l \leq l' \leq r' \leq r \wedge r' - l' + 1 \geq k} \text{dep}_{\text{LCA}^*(l', r')}$$

Your task is to help Little S answer these queries.

Input Format

The first line contains a positive integer n , the number of nodes in the tree.

The next $n - 1$ lines each contain two positive integers u, v , representing an edge between nodes u and v .

The $(n + 1)$ -th line contains a positive integer q , the number of queries.

The next q lines each contain three positive integers l, r, k , describing a query.

Output Format

For each query, output one line containing an integer, the answer.

Sample Input #1

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

Sample Output #1

```
3
4
3
```

Explanation for Sample 1

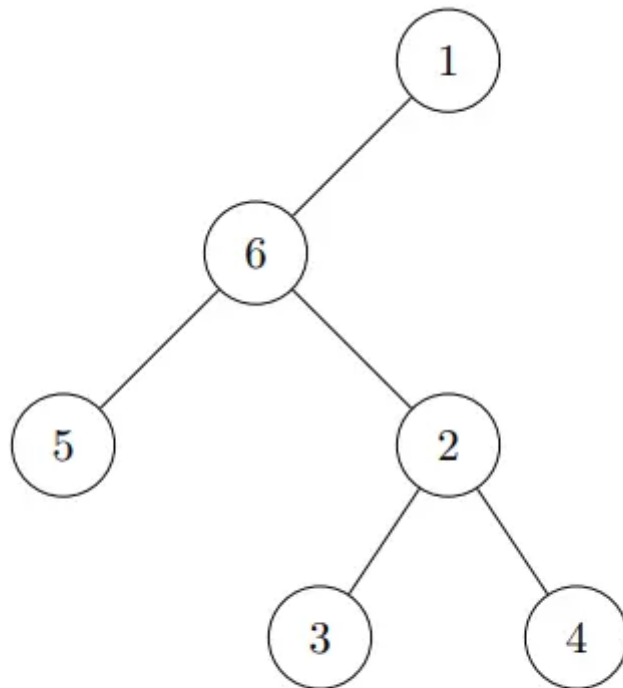


图 3: 样例 1 中的树

洛谷

- For the first query, $LCA^*(2, 3) = 2$, $LCA^*(3, 4) = 2$, $LCA^*(4, 5) = 6$. The depths are 3, 3, and 2, so the answer is $\max\{3, 3, 2\} = 3$.
- For the second query, the answer is the maximum depth of nodes 1, 2, 3, 4, which is 4.
- For the third query, $LCA^*(1, 3) = 1$, $LCA^*(2, 4) = 2$, $LCA^*(3, 5) = 6$, $LCA^*(4, 6) = 6$. The maximum depth is 3 (node 2).

Data Range

For all test cases, it is guaranteed that: $1 \leq n, q \leq 5 \times 10^5$, $1 \leq l \leq r \leq n$, $1 \leq k \leq r - l + 1$.

Test Case ID	$n, q \leq$	Special Properties
1 ~ 2	500	None
3 ~ 5	5000	None
6 ~ 9	10^5	A
10 ~ 13	5×10^5	A
14 ~ 16	5×10^5	B
17 ~ 20	10^5	None
21 ~ 25	5×10^5	None

- **Special Property A:** The input tree is a chain, and the root has degree 1.
- **Special Property B:** For each query, $k = r - l + 1$.