Educational Codeforces Round 67 (Rated for Div. 2)

A. Stickers and Toys

2 seconds, 256 megabytes

Your favorite shop sells n Kinder Surprise chocolate eggs. You know that exactly s stickers and exactly t toys are placed in n eggs in total.

Each Kinder Surprise can be one of three types:

- it can contain a single sticker and **no toy**;
- it can contain a single toy and **no sticker**;
- it can contain both a single sticker **and** a single toy.

But you **don't know** which type a particular Kinder Surprise has. All eggs look identical and indistinguishable from each other.

What is the minimum number of Kinder Surprise Eggs you have to buy to be sure that, whichever types they are, you'll obtain at least one sticker and at least one toy?

Note that you do not open the eggs in the purchasing process, that is, you just buy some number of eggs. It's guaranteed that the answer always exists.

Input

The first line contains the single integer T ($1 \le T \le 100$) — the number of queries.

Next T lines contain three integers n, s and t each ($1 \le n \le 10^9$, $1 \le s, t \le n$, $s + t \ge n$) — the number of eggs, stickers and toys.

All queries are independent.

Output

Print T integers (one number per query) — the minimum number of Kinder Surprise Eggs you have to buy to be sure that, whichever types they are, you'll obtain at least one sticker and one toy

input
3 10 5 7 10 10 10 2 1 1
output
6 1 2

In the first query, we have to take at least 6 eggs because there are 5 eggs with *only toy* inside and, in the worst case, we'll buy all of them.

In the second query, all eggs have both a sticker and a toy inside, that's why it's enough to buy only one egg.

In the third query, we have to buy both eggs: one with a sticker and one with a toy.

B. Letters Shop

2 seconds, 256 megabytes

The letters shop showcase is a string s, consisting of n lowercase Latin letters. As the name tells, letters are sold in the shop.

Letters are sold one by one from the leftmost to the rightmost. Any customer can only buy some prefix of letters from the string s.

There are m friends, the i-th of them is named t_i. Each of them is planning to estimate the following value: how many letters (the length of the shortest prefix) would s/he need to buy if s/he wanted to construct her/his name of bought letters. The name can be constructed if each letter is presented in the equal or greater amount.

- For example, for s="arrayhead" and t_i="arya" 5 letters have to be bought ("**array**head").
- For example, for s="arrayhead" and t_i="harry" 6 letters have to be bought ("**arrayh**ead").
- For example, for s="arrayhead" and t_i="ray" 5 letters have to be bought ("**array**head").
- For example, for s="arrayhead" and t_i="r" 2 letters have to be bought ("**ar**rayhead").
- For example, for s="arrayhead" and t_i="areahydra" all 9 letters have to be bought ("**arrayhead**").

It is guaranteed that every friend can construct her/his name using the letters from the string s.

Note that the values for friends are independent, friends are only estimating them but not actually buying the letters.

Input

The first line contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the length of showcase string s.

The second line contains string s, consisting of exactly n lowercase Latin letters.

The third line contains one integer m ($1 \le m \le 5 \cdot 10^4$) — the number of friends.

The i-th of the next m lines contains t_i ($1 \le |t_i| \le 2 \cdot 10^5$) — the name of the i-th friend.

It is guaranteed that $\sum_{i=1}^{m} |t_i| \le 2 \cdot 10^5$.

Output

For each friend print the length of the shortest prefix of letters from s s/he would need to buy to be able to construct her/his name of them. The name can be constructed if each letter is presented in the equal or greater amount.

It is guaranteed that every friend can construct her/his name using the letters from the string s.

input
9 arrayhead 5 arya harry ray r areahydra
output
5 6 5 2 9

C. Vasya And Array

1 second, 256 megabytes

Vasya has an array $a_1, a_2, \ldots, a_n$.

You don't know this array, but he told you m facts about this array. The i-th fact is a triple of numbers t_i, l_i and r_i ($0 \le t_i \le 1, 1 \le l_i < r_i \le n$) and it means:

- if $t_i = 1$ then subbarray $a_{l_i}, a_{l_i+1}, \ldots, a_{r_i}$ is sorted in non-decreasing order;
- if $t_i = 0$ then subbarray $a_{l_i}, a_{l_i+1}, \ldots, a_{r_i}$ is not sorted in non-decreasing order. A subarray is not sorted if there is at least one pair of consecutive elements in this subarray such that the former is greater than the latter.

For example if $a = [2, 1, 1, 3, 2]$ then he could give you three facts: $t_1 = 1, l_1 = 2, r_1 = 4$ (the subarray $[a_2, a_3, a_4] = [1, 1, 3]$ is sorted), $t_2 = 0, l_2 = 4, r_2 = 5$ (the subarray $[a_4, a_5] = [3, 2]$ is not sorted), and $t_3 = 0, l_3 = 3, r_3 = 5$ (the subarray $[a_3, a_5] = [1, 3, 2]$ is not sorted).

You don't know the array a. Find **any** array which satisfies all the given facts.

Input

The first line contains two integers n and m ($2 \le n \le 1000, 1 \le m \le 1000$).

Each of the next m lines contains three integers t_i, l_i and r_i ($0 \le t_i \le 1, 1 \le l_i < r_i \le n$).

If $t_i = 1$ then subbarray $a_{l_i}, a_{l_i+1}, \ldots, a_{r_i}$ is sorted. Otherwise (if $t_i = 0$) subbarray $a_{l_i}, a_{l_i+1}, \ldots, a_{r_i}$ is not sorted.

Output

If there is no array that satisfies these facts in only line print `NO` (in any letter case).

If there is a solution, print `YES` (in any letter case). In second line print n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the array a, satisfying all the given facts. If there are multiple satisfying arrays you can print any of them.

input
7 4 1 1 3 1 2 5 0 5 6 1 6 7

output
YES 1 2 2 3 5 4 4

input
4 2 1 1 4 0 2 3

output
NO

D. Subarray Sorting

2 seconds, 256 megabytes

You are given an array $a_1, a_2, \ldots, a_n$ and an array $b_1, b_2, \ldots, b_n$.

For one operation you can sort in non-decreasing order any subarray $a[l \ldots r]$ of the array a.

For example, if $a = [4, 2, 2, 1, 3, 1]$ and you choose subbarray $a[2 \ldots 5]$, then the array turns into $[4, 1, 2, 2, 3, 1]$.

You are asked to determine whether it is possible to obtain the array b by applying this operation any number of times (possibly zero) to the array a.

Input

The first line contains one integer t ($1 \le t \le 3 \cdot 10^5$) — the number of queries.

The first line of each query contains one integer n ($1 \le n \le 3 \cdot 10^5$).

The second line of each query contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$).

The third line of each query contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le n$).

It is guaranteed that $\sum n \le 3 \cdot 10^5$ over all queries in a test.

Output

For each query print `YES` (in any letter case) if it is possible to obtain an array b and `NO` (in any letter case) otherwise.

input
4 7 1 7 1 4 4 5 6 1 1 4 4 5 7 6 5 1 1 3 3 5 1 1 3 3 5 2 1 1 1 2 3 1 2 3 3 2 1

output
YES YES NO NO

In first test case the can sort subarray $a_1 \ldots a_5$, then a will turn into $[1, 1, 4, 4, 7, 5, 6]$, and then sort subarray $a_5 \ldots a_6$.

E. Tree Painting

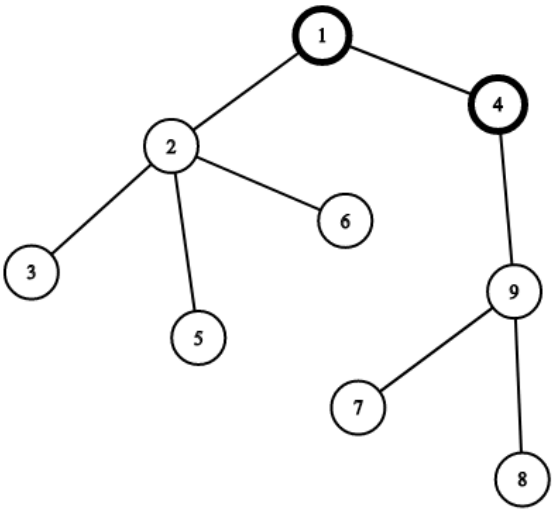
2 seconds, 256 megabytes

You are given a tree (an undirected connected acyclic graph) consisting of n vertices. You are playing a game on this tree.

Initially all vertices are white. On the first turn of the game you choose one vertex and paint it black. Then on each turn you choose a white vertex adjacent (connected by an edge) to **any** black vertex and paint it black.

Each time when you choose a vertex (even during the first turn), you gain the number of points equal to the size of the connected component consisting only of white vertices that contains the chosen vertex. The game ends when all vertices are painted black.

Let's see the following example:



Vertices 1 and 4 are painted black already. If you choose the vertex 2, you will gain 4 points for the connected component consisting of vertices $2, 3, 5$ and 6. If you choose the vertex 9, you will gain 3 points for the connected component consisting of vertices $7, 8$ and 9.

Your task is to maximize the number of points you gain.

Input

The first line contains an integer n — the number of vertices in the tree ($2 \le n \le 2 \cdot 10^5$).

Each of the next $n - 1$ lines describes an edge of the tree. Edge i is denoted by two integers u_i and v_i, the indices of vertices it connects ($1 \le u_i, v_i \le n, u_i \ne v_i$).

It is guaranteed that the given edges form a tree.

Output

Print one integer — the maximum number of points you gain if you will play optimally.

input
9 1 2 2 3 2 5 2 6 1 4 4 9 9 7 9 8
output
36

input
5 1 2 1 3 2 4 2 5
output
14

The first example tree is shown in the problem statement.

F. Expected Square Beauty

2 seconds, 256 megabytes

Let x be an array of integers $x = [x_1, x_2, \ldots, x_n]$. Let's define $B(x)$ as a minimal size of a partition of x into subsegments such that all elements in each subsegment are equal. For example, $B([3, 3, 6, 1, 6, 6, 6]) = 4$ using next partition: $[3, 3 \mid 6 \mid 1 \mid 6, 6, 6]$.

Now you don't have any exact values of x, but you know that x_i can be any integer value from $[l_i, r_i]$ ($l_i \le r_i$) uniformly at random. All x_i are independent.

Calculate expected value of $(B(x))^2$, or $E((B(x))^2)$. It's guaranteed that the expected value can be represented as rational fraction $\frac{P}{Q}$ where $(P, Q) = 1$, so print the value $P \cdot Q^{-1} \mod 10^9 + 7$.

Input

The first line contains the single integer n ($1 \le n \le 2 \cdot 10^5$) — the size of the array x.

The second line contains n integers $l_1, l_2, \ldots, l_n$ ($1 \le l_i \le 10^9$).

The third line contains n integers $r_1, r_2, \ldots, r_n$ ($l_i \le r_i \le 10^9$).

Output

Print the single integer — $E((B(x))^2)$ as $P \cdot Q^{-1} \mod 10^9 + 7$.

input
3 1 1 1 1 2 3
output
166666673

input
3 3 4 5 4 5 6
output
500000010

Let's describe all possible values of x for the first sample:

- $[1, 1, 1]$: $B(x) = 1, B^2(x) = 1$;
- $[1, 1, 2]$: $B(x) = 2, B^2(x) = 4$;
- $[1, 1, 3]$: $B(x) = 2, B^2(x) = 4$;
- $[1, 2, 1]$: $B(x) = 3, B^2(x) = 9$;
- $[1, 2, 2]$: $B(x) = 2, B^2(x) = 4$;
- $[1, 2, 3]$: $B(x) = 3, B^2(x) = 9$;

So $E = \frac{1}{6}(1 + 4 + 4 + 9 + 4 + 9) = \frac{31}{6}$ or $31 \cdot 6^{-1} = 166666673$.
All possible values of x for the second sample:

- $[3, 4, 5]$: $B(x) = 3, B^2(x) = 9$;
- $[3, 4, 6]$: $B(x) = 3, B^2(x) = 9$;
- $[3, 5, 5]$: $B(x) = 2, B^2(x) = 4$;
- $[3, 5, 6]$: $B(x) = 3, B^2(x) = 9$;
- $[4, 4, 5]$: $B(x) = 2, B^2(x) = 4$;
- $[4, 4, 6]$: $B(x) = 2, B^2(x) = 4$;
- $[4, 5, 5]$: $B(x) = 2, B^2(x) = 4$;
- $[4, 5, 6]$: $B(x) = 3, B^2(x) = 9$;

So $E = \frac{1}{8}(9 + 9 + 4 + 9 + 4 + 4 + 4 + 9) = \frac{52}{8}$ or $13 \cdot 2^{-1} = 500000010$.

G. Gang Up

4 seconds, 512 megabytes

The leader of some very secretive organization has decided to invite all other members to a meeting. All members of the organization live in the same town which can be represented as n crossroads connected by m two-directional streets. The meeting will be held in the leader's house near the crossroad 1. There are k members of the organization invited to the meeting; i-th of them lives near the crossroad a_i.

All members of the organization receive the message about the meeting at the same moment and start moving to the location where the meeting is held. In the beginning of each minute each person is located at some crossroad. He or she can either wait a minute at this crossroad, or spend a minute to walk from the current crossroad along some street to another crossroad (obviously, it is possible to start walking along the street only if it begins or ends at the current crossroad). In the beginning of the first minute each person is at the crossroad where he or she lives. As soon as a person reaches the crossroad number 1, he or she immediately comes to the leader's house and attends the meeting.

Obviously, the leader wants all other members of the organization to come up as early as possible. But, since the organization is very secretive, the leader does not want to attract much attention. Let's denote the *discontent* of the leader as follows

- initially the discontent is 0;
- whenever a person reaches the crossroad number 1, the discontent of the leader increases by $c \cdot x$, where c is some fixed constant, and x is the number of minutes it took the person to reach the crossroad number 1;
- whenever x members of the organization walk **along the same street at the same moment in the same direction**, dx^2 is added to the discontent, where d is some fixed constant. This is not cumulative: for example, if two persons are walking along the same street in the same direction at the same moment, then $4d$ is added to the discontent, not $5d$.

Before sending a message about the meeting, the leader can tell each member of the organization which path they should choose and where they should wait. Help the leader to establish a plan for every member of the organization so they all reach the crossroad 1, and the discontent is minimized.

Input

The first line of the input contains five integer numbers n, m, k, c and d ($2 \le n \le 50, n - 1 \le m \le 50, 1 \le k, c, d \le 50$) — the number of crossroads, the number of streets, the number of persons invited to the meeting and the constants affecting the discontent, respectively.

The second line contains k numbers $a_1, a_2, ..., a_k$ ($2 \le a_i \le n$) — the crossroads where the members of the organization live.

Then m lines follow, each denoting a bidirectional street. Each line contains two integers x_i and y_i ($1 \le x_i, y_i \le n, x_i \ne y_i$) denoting a street connecting crossroads x_i and y_i. **There may be multiple streets connecting the same pair of crossroads**.

It is guaranteed that every crossroad can be reached from every other crossroad using the given streets.

Output

Print one integer: the minimum discontent of the leader after everyone reaches crossroad 1.

input
3 2 4 2 3 3 3 3 3 1 2 2 3
output
52

input
3 3 4 2 3 3 2 2 3 1 2 2 3 2 3
output
38

The best course of action in the first test is the following:

- the first person goes along the street 2 to the crossroad 2, then goes along the street 1 to the crossroad 1 and attends the meeting;
- the second person waits one minute on the crossroad 3, then goes along the street 2 to the crossroad 2, then goes along the street 1 to the crossroad 1 and attends the meeting;
- the third person waits two minutes on the crossroad 3, then goes along the street 2 to the crossroad 2, then goes along the street 1 to the crossroad 1 and attends the meeting;
- the fourth person waits three minutes on the crossroad 3, then goes along the street 2 to the crossroad 2, then goes along the street 1 to the crossroad 1 and attends the meeting.