CodeForces Problem

May 28, 2023

A. Twin Permutations

Constriants

Time Limit 1 seconds **Memory Limit** 256 MB

Problem Statement

You are given a permutation[†] a of length n.

Find any permutation b of length n such that $a_1 + b_1 \le a_2 + b_2 \le a_3 + b_3 \le \ldots \le a_n + b_n$.

It can be proven that a permutation b that satisfies the condition above always exists.

[†] A permutation of length n is an array consisting of n distinct integers from 1 to n in arbitrary order. For example, $[2, 3, 1, 5, 4]$ is a permutation, but $[1, 2, 2]$ is not a permutation (2 appears twice in the array), and $[1, 3, 4]$ is also not a permutation ($n = 3$ but there is 4 in the array).

Input Description

Each test contains multiple test cases. The first line of input contains a single integer t ($1 \le t \le 2000$) — the number of test cases. The description of test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 100$) — the length of permutations a and b.

The second line of each test case contains n distinct integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the elements of permutation a. All elements of a are distinct.

Note that there is no bound on the sum of n over all test cases.

Output Description :

For each test case, output any permutation b which satisfies the constraints mentioned in the statement. It can be proven that a permutation b that satisfies the condition above always exists.

Examples

```
Input
5
5
1 2 4 5 3
2
1 2
1
1
3
3 2 1
4
1 4 3 2
Output
1 2 4 3 5
2 1
1
1 2 3
1 2 3 4
```

Note

In the first test case $a = [1, 2, 4, 5, 3]$. Then the permutation $b = [1, 2, 4, 3, 5]$ satisfies the condition because $1 + 1 \le 2 + 2 \le 4 + 4 \le 5 + 3 \le 3 + 5$.

B. Array merging

Constriants

Time Limit 1 seconds **Memory Limit** 256 MB

Problem Statement

You are given two arrays a and b both of length n.

You will merge[†] these arrays forming another array c of length $2 \cdot n$. You have to find the maximum length of a subarray consisting of equal values across all arrays c that could be obtained.

[†] A merge of two arrays results in an array c composed by successively taking the first element of either array (as long as that array is nonempty) and removing it. After this step, the element is appended to the back of c. We repeat this operation as long as we can (i.e. at least one array is nonempty).

Input Description

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

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

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 2 \cdot n$) — the elements of array a.

The third line of each test case contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 2 \cdot n$) — the elements of array b.

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

Output Description :

For each test case, output the maximum length of a subarray consisting of equal values across all merges.

Examples

Input

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

Output

```
2
1
2
5
```

Note

In the first test case, we can only make $c = [2, 2]$, thus the answer is 2.
In the second test case, since all values are distinct, the answer must be 1.
In the third test case, the arrays c we can make are $[1, 2, 1, 2]$, $[1, 2, 2, 1]$, $[2, 1, 1, 2]$, $[2, 1, 2, 1]$. We can see that the answer is 2 when we choose $c = [1, 2, 2, 1]$.

C. Copil Copac Draws Trees

Constriants

Time Limit 3 seconds **Memory Limit** 256 MB

Problem Statement

Copil Copac is given a list of $n-1$ edges describing a tree of n vertices. He decides to draw it using the following algorithm: Step 0: Draws the first vertex (vertex 1). Go to step 1. Step 1: For every edge in the input, in order: if the edge connects an already drawn vertex u to an undrawn vertex v, he will draw the undrawn vertex v and the edge. After checking every edge, go to step 2. Step 2: If all the vertices are drawn, terminate the algorithm. Else, go to step 1.

A reading is defined as the number of times Copil Copac performs step 1.

Find the number of readings needed by Copil Copac to draw the tree.

Input Description

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

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

The following $n-1$ lines of each test case contain two integers u_i and v_i ($1 \le u_i, v_i \le n$, $u_i \neq v_i$) — indicating that (u_i, v_i) is the i-th edge in the list. 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 $2 \cdot 10^5$.

Output Description :

For each test case, output the number of readings Copil Copac needs to draw the tree.

Examples

Input

```
2
6
4 5
1 3
1 2
3 4
1 6
7
5 6
2 4
2 7
1 3
1 2
4 5
```

Output

```
2
3
```

Note

In the first test case:
After the first reading, the tree will look like this:
After the second reading:
Therefore, Copil Copac needs 2 readings to draw the tree.

D. The BOSS Can Count Pairs

Constriants

Time Limit 4 seconds **Memory Limit** 512 MB

Problem Statement

You are given two arrays a and b, both of length n.

Your task is to count the number of pairs of integers (i, j) such that $1 \leq i <; j \leq n$ and $a_i \cdot a_j = b_i + b_j$.

Input Description

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

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

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

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

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

Output Description :

For each test case, output the number of good pairs.

Examples

Input
```
3
3
2 3 2
3 3 1
8
4 2 8 2 1 2 7 5
3 5 8 8 1 1 6 5
8
4 4 8 8 8 8 8 8
8 8 8 8 8 8 8 8
```
Output
```
2
7
1
```

Note

In the first sample, there are 2 good pairs: $(1, 2)$, $(1, 3)$.
In the second sample, there are 7 good pairs: $(1, 2)$, $(1, 5)$, $(2, 8)$, $(3, 4)$, $(4, 7)$, $(5, 6)$, $(5, 7)$.

E. Hyperregular Bracket Strings

Constriants

Time Limit 3 seconds **Memory Limit** 256 MB

Problem Statement

You are given an integer n and k intervals. The i-th interval is $[l_i, r_i]$ where $1 \le l_i \le r_i \le n$.

Let us call a regular bracket sequence[†,‡] of length n hyperregular if for each i such that $1 \le i \le k$, the substring $\overline{s_{l_i} s_{l_i+1} \dots s_{r_i}}$ is also a regular bracket sequence.

Your task is to count the number of hyperregular bracket sequences. Since this number can be really large, you are only required to find it modulo 998 244 353.

[†] A bracket sequence is a string containing only the characters "(" and ")".

[‡] A bracket sequence is called regular if one can turn it into a valid math expression by adding characters + and 1. For example, sequences (())(), (), (()(())) and the empty string are regular, while)(, ((), and (()))(are not.

Input Description

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

The first line of each test case contains two integers n and k ($1 \le n \le 3 \cdot 10^5$, $0 \le k \le 3 \cdot 10^5$) — the length of the hyperregular bracket sequences and the number of intervals respectively.

The following k lines of each test case contains two integers l_i and r_i ($1 \le l \le r \le n$).

It is guaranteed that the sum of n across all test cases does not exceed $3 \cdot 10^5$ and the sum of k across all test cases does not exceed $3 \cdot 10^5$.

Output Description :

For each test case, output the number of hyperregular bracket sequences modulo 998 244 353.

Examples

Input

```
7
6 0
5 0
8 1
1 3
10 2
3 4
6 9
1000 3
100 701
200 801
300 901
28 5
1 12
3 20
11 14
4 9
18 19
4 3
1 4
1 4
1 4
```

Output

```
5
0
0
4
839415253
140
2
```

F. Mex Tree

Constriants

Time Limit 3 seconds **Memory Limit** 256 MB

Problem Statement

You are given a tree with n nodes. For each node, you either color it in 0 or 1.

The value of a path (u, v) is equal to the MEX[†] of the colors of the nodes from the shortest path between u and v.

The value of a coloring is equal to the sum of values of all paths (u, v) such that $1 \leq u \leq v \leq n$.

What is the maximum possible value of any coloring of the tree?

[†] The MEX (minimum excluded) of an array is the smallest non-negative integer that does not belong to the array. For instance: The MEX of $[2, 2, 1]$ is 0, because 0 does not belong to the array. The MEX of $[3, 1, 0, 1]$ is 2, because 0 and 1 belong to the array, but 2 does not. The MEX of $[0, 3, 1, 2]$ is 4 because 0, 1, 2, and 3 belong to the array, but 4 does not.

Input Description

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

The first line of each test case contains a single integer n ($1 \leq n \leq 2 \cdot 10^5$) — the number of nodes in the tree.

The following $n-1$ lines of each test case contains 2 integers a_i and b_i ($1 \leq a_i, b_i \leq n, a_i \neq b_i$) — indicating an edge between vertices a_i and b_i. It is guaranteed that the given edges form a tree.

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

Output Description :

For each test case, print the maximum possible value of any coloring of the tree.

Examples

Input

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

Output

```
8
15
96
1
```

Note

In the first sample, we will color vertex 2 in 1 and vertices $1, 3$ in 0. After this, we consider all paths: $(1, 1)$ with value 1 $(1, 2)$ with value 2 $(1, 3)$ with value 2 $(2, 2)$ with value 0 $(2, 3)$ with value 2 $(3, 3)$ with value 1
We notice the sum of values is 8 which is the maximum possible.