Codeforces Round #722 (Div. 2)

A. Eshag Loves Big Arrays

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Eshag has an array a consisting of n integers.

Eshag can perform the following operation any number of times: choose some subsequence of a and delete every element from it which is **strictly** larger than AVG, where AVG is the average of the numbers in the chosen subsequence.

For example, if $a = [1, 4, 3, 2, 4]$ and Eshag applies the operation to the subsequence containing a_1, a_2, a_4 and a_5, then he will delete those of these 4 elements which are larger than $\frac{a_1+a_2+a_4+a_5}{4} = \frac{11}{4}$, so after the operation, the array a will become $a = [1, 3, 2]$.

Your task is to find the **maximum** number of elements Eshag can delete from the array a by applying the operation described above some number (maybe, zero) times.

A sequence b is a subsequence of an array c if b can be obtained from c by deletion of several (possibly, zero or all) elements.

Input

The first line contains an integer t $(1 \le t \le 100)$ — the number of test cases. The description of the test cases follows.

The first line of each test case contains an integer n $(1 \le n \le 100)$ — the length of the array a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 100)$ — the elements of the array a.

Output

For each test case print a single integer — the **maximum** number of elements Eshag can delete from the array a.

Example

input

```
3
6
1 1 1 2 2 3
6
9 9 9 9 9 9
6
6 4 1 1 4 1
```

output

```
3
0
3
```

Note

Consider the first test case.

Initially $a = [1, 1, 1, 2, 2, 3]$.

In the first operation, Eshag can choose the subsequence containing a_1, a_5 and a_6, their average is equal to $\frac{a_1+a_5+a_6}{3} = \frac{6}{3} = 2$. So a_6 will be deleted.

After this $a = [1, 1, 1, 2, 2]$.

In the second operation, Eshag can choose the subsequence containing the whole array a, the average of all its elements is equal to $\frac{7}{5}$. So a_4 and a_5 will be deleted.

After this $a = [1, 1, 1]$.

In the second test case, Eshag can't delete any element.

B. Sifid and Strange Subsequences

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

A sequence $(b_1, b_2, \ldots, b_k)$ is called **strange**, if the absolute difference between any pair of its elements is greater than or equal to the maximum element in the sequence. Formally speaking, it's strange if for every pair (i, j) with $1 \le i < j \le k$, we have $|a_i - a_j| \ge MAX$, where MAX is the largest element of the sequence. In particular, any sequence of length at most 1 is strange.

For example, the sequences $(-2021, -1, -1, -1)$ and $(-1, 0, 1)$ are strange, but $(3, 0, 1)$ is not, because $|0 - 1| < 3$.

Sifid has an array a of n integers. Sifid likes everything big, so among all the strange subsequences of a, he wants to find the length of the **longest** one. Can you help him?

A sequence c is a subsequence of an array d if c can be obtained from d by deletion of several (possibly, zero or all) elements.

Input

The first line contains an 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 an integer n $(1 \le n \le 10^5)$ — the length of the array a.

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

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

Output

For each test case output a single integer — the length of the **longest** strange subsequence of a.

Example

input

```
6
4
-1 -2 0 0
7
-3 4 -2 0 -4 6 1
5
0 5 -3 2 -5
3
2 3 1
4
-3 0 2 0
6
-3 -2 -1 1 1 1
```

output

```
4
5
4
1
3
4
```

Note

In the first test case, one of the longest strange subsequences is (a_1, a_2, a_3, a_4)

In the second test case, one of the longest strange subsequences is $(a_1, a_3, a_4, a_5, a_7)$.

In the third test case, one of the longest strange subsequences is (a_1, a_3, a_4, a_5).

In the fourth test case, one of the longest strange subsequences is (a_2).

In the fifth test case, one of the longest strange subsequences is (a_1, a_2, a_4).

C. Parsa's Humongous Tree

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Parsa has a humongous tree on n vertices.

On each vertex v he has written two integers l_v and r_v.

To make Parsa's tree look even more majestic, Nima wants to assign a number a_v ($l_v \le a_v \le r_v$) to each vertex v such that the beauty of Parsa's tree is maximized.

Nima's sense of the beauty is rather bizarre. He defines the beauty of the tree as the sum of $|a_u - a_v|$ over all edges (u, v) of the tree.

Since Parsa's tree is too large, Nima can't maximize its beauty on his own. Your task is to find the **maximum** possible beauty for Parsa's tree.

Input

The first line contains an integer t $(1 \le t \le 250)$ — the number of test cases. The description of the test cases follows.

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

The i-th of the following n lines contains two integers l_i and r_i $(1 \le l_i \le r_i \le 10^9)$.

Each of the next $n-1$ lines contains two integers u and v $(1 \le u, v \le n, u \ne v)$ meaning that there is an edge between the vertices u and v in Parsa's tree.

It is guaranteed that the given graph is a tree.

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

Output

For each test case print the **maximum** possible beauty for Parsa's tree.

Example

input

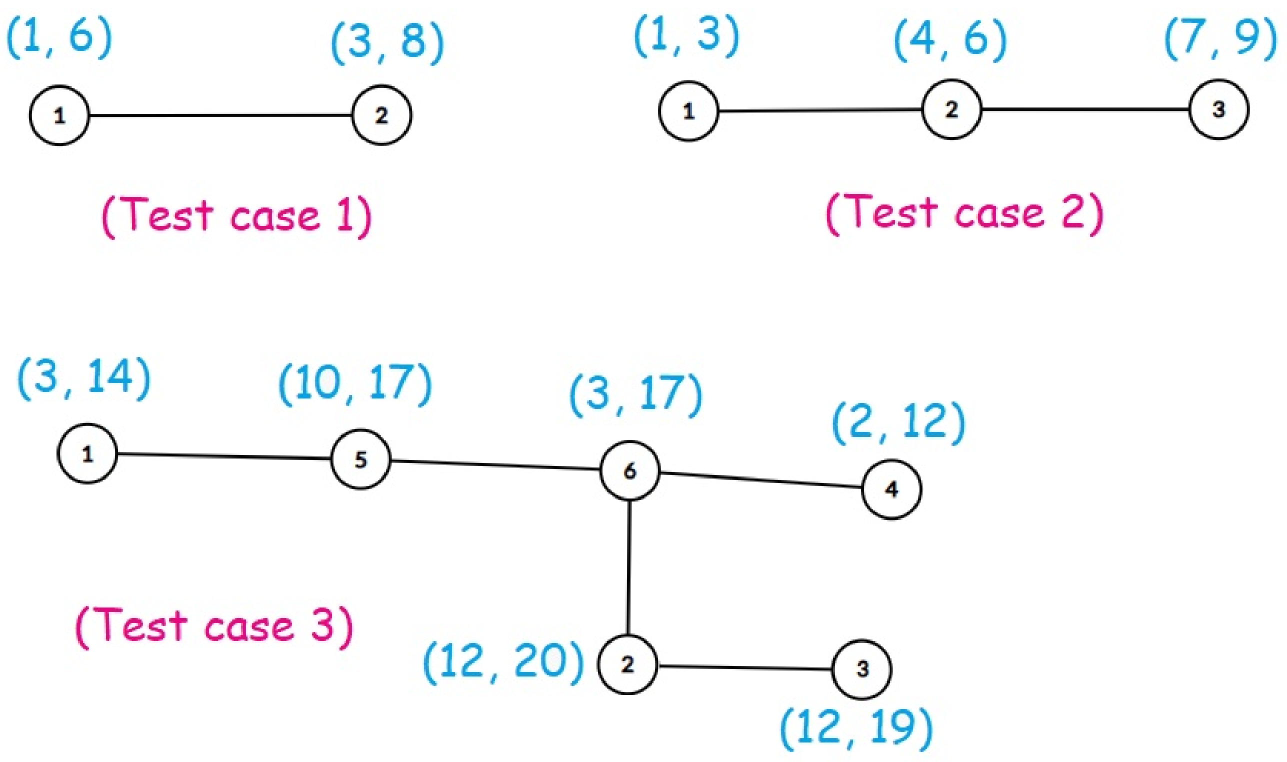
```
3
2
1 6
3 8
1 2
3
1 3
4 6
7 9
1 2
2 3
6
3 14
12 20
12 19
2 12
10 17
3 17
3 2
6 5
1 5
2 6
4 6
```

output

```
7
8
62
```

Note

The trees in the example:



In the first test case, one possible assignment is $a = \{1, 8\}$ which results in $|1 - 8| = 7$.

In the second test case, one of the possible assignments is $a = \{1, 5, 9\}$ which results in a beauty of $|1 - 5| + |5 - 9| = 8$

D. Kavi on Pairing Duty

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

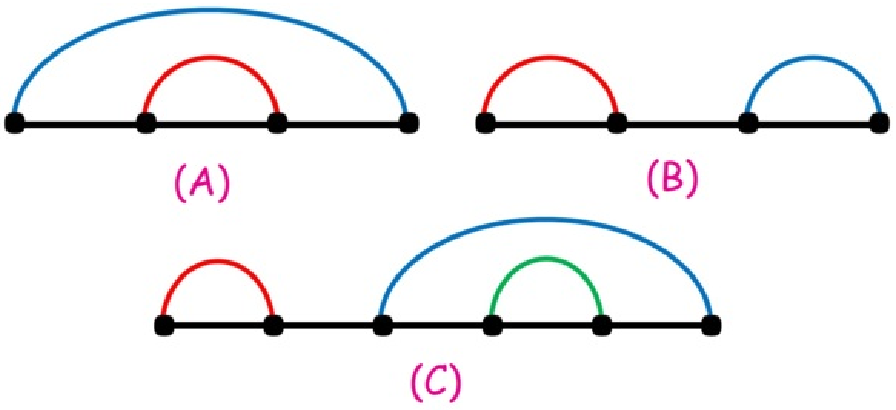
Kavi has $2n$ points lying on the OX axis, i-th of which is located at $x = i$.

Kavi considers all ways to split these $2n$ points into n pairs. Among those, he is interested in **good** pairings, which are defined as follows:

Consider n segments with ends at the points in correspondent pairs. The pairing is called good, if for every 2 different segments A and B among those, at least one of the following holds:

- One of the segments A and B lies completely inside the other.
- A and B have the same length.

Consider the following example:



A is a good pairing since the red segment lies completely inside the blue segment.

B is a good pairing since the red and the blue segment have the same length.

C is not a good pairing since none of the red or blue segments lies inside the other, neither do they have the same size.

Kavi is interested in the number of good pairings, so he wants you to find it for him. As the result can be large, find this number modulo 998244353.

Two pairings are called different, if some two points are in one pair in some pairing and in different pairs in another.

Input

The single line of the input contains a single integer n $(1 \le n \le 10^6)$.

Output

Print the number of good pairings modulo 998244353.

Examples

input
```
1
```
output
```
1
```

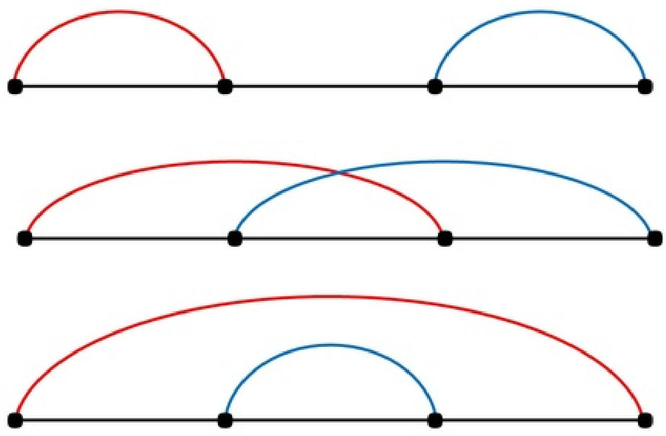
input
```
2
```
output
```
3
```

input
```
3
```
output
```
6
```

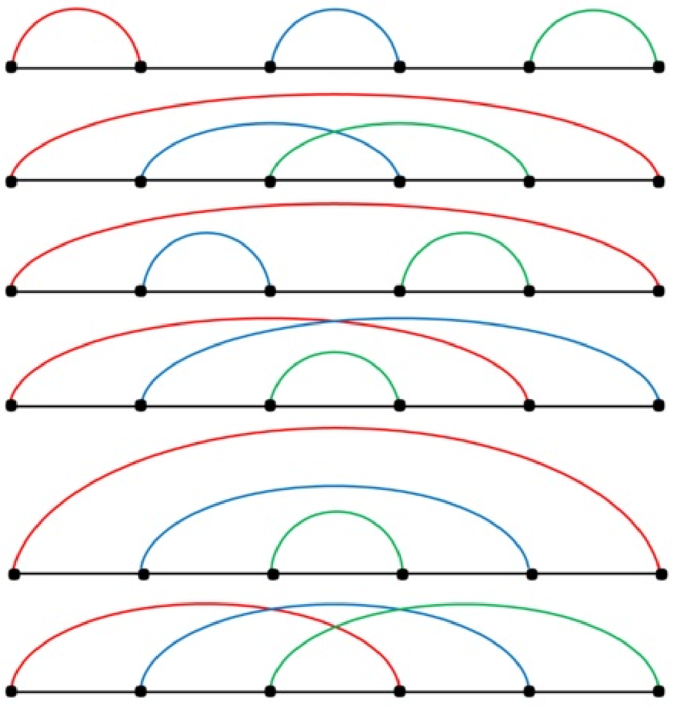
input
```
100
```
output
```
688750769
```

Note

The good pairings for the second example are:



In the third example, the good pairings are:



E. Trees of Tranquillity

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Soroush and Keshi each have a labeled and rooted tree on n vertices. Both of their trees are rooted from vertex 1.

Soroush and Keshi used to be at war. After endless decades of fighting, they finally became allies to prepare a Codeforces round. To celebrate this fortunate event, they decided to make a memorial graph on n vertices.

They add an edge between vertices u and v in the memorial graph if **both** of the following conditions hold:

- One of u or v is the ancestor of the other in Soroush's tree.
- Neither of u or v is the ancestor of the other in Keshi's tree.

Here vertex u is considered ancestor of vertex v, if u lies on the path from 1 (the root) to the v.

Popping out of nowhere, Mashtali tried to find the maximum clique in the memorial graph for no reason. He failed because the graph was too big.

Help Mashtali by finding the size of the maximum clique in the memorial graph.

As a reminder, clique is a subset of vertices of the graph, each two of which are connected by an edge.

Input

The first line contains an integer t $(1 \le t \le 3 \cdot 10^5)$ — the number of test cases. The description of the test cases follows.

The first line of each test case contains an integer n $(2 \le n \le 3 \cdot 10^5)$.

The second line of each test case contains $n - 1$ integers $a_2, \ldots, a_n$ $(1 \le a_i < i)$, a_i being the parent of the vertex i in Soroush's tree.

The third line of each test case contains $n - 1$ integers $b_2, \ldots, b_n$ $(1 \le b_i < i)$, b_i being the parent of the vertex i in Keshi's tree.

It is guaranteed that the given graphs are trees.

It is guaranteed that the sum of n over all test cases doesn't exceed $3 \cdot 10^5$.

Output

For each test case print a single integer — the **size** of the maximum clique in the memorial graph.

Example

input

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

output

```
1
4
1
3
```

Note

In the first and third test cases, you can pick any vertex.

In the second test case, one of the maximum cliques is $\{2, 3, 4, 5\}$.

In the fourth test case, one of the maximum cliques is $\{3, 4, 6\}$.

F. It's a bird! No, it's a plane! No, it's AaParsa!

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n cities in Shaazzzland, numbered from 0 to $n - 1$. Ghaazzzland, the immortal enemy of Shaazzzland, is ruled by AaParsa.

As the head of the Ghaazzzland's intelligence agency, AaParsa is carrying out the most important spying mission in Ghaazzzland's history on Shaazzzland.

AaParsa has planted m transport cannons in the cities of Shaazzzland. The i-th cannon is planted in the city a_i and is initially pointing at city b_i.

It is guaranteed that each of the n cities has **at least one** transport cannon planted inside it, and that no two cannons from the same city are initially pointing at the same city (that is, all pairs (a_i, b_i) are distinct).

AaParsa used very advanced technology to build the cannons, the cannons rotate every second. In other words, if the i-th cannon is pointing towards the city x at some second, it will target the city $(x + 1) \mod n$ at the next second.

As their name suggests, transport cannons are for transportation, specifically for human transport. If you use the i-th cannon to launch yourself towards the city that it's currently pointing at, you'll be airborne for c_i seconds before reaching your target destination.

If you still don't get it, using the i-th cannon at the s-th second (using which is only possible if you are currently in the city a_i) will shoot you to the city $(b_i + s) \mod n$ and you'll land in there after c_i seconds (so you'll be there in the $(s + c_i)$-th second). Also note the cannon that you initially launched from will rotate every second but you obviously won't change direction while you are airborne.

AaParsa wants to use the cannons for travelling between Shaazzzland's cities in his grand plan, and he can start travelling at second 0. For him to fully utilize them, he needs to know the minimum number of seconds required to reach city u from city v using the cannons for every pair of cities (u, v).

Note that AaParsa can stay in a city for as long as he wants.

Input

The first line contains two integers n and m $(2 \leq n \leq 600, n \leq m \leq n^2)$ — the number of cities and cannons correspondingly.

The i-th line of the following m lines contains three integers a_i, b_i and c_i $(0 \leq a_i, b_i \leq n-1, 1 \leq c_i \leq 10^9)$, denoting the cannon in the city a_i, which is initially pointing to b_i and travelling by which takes c_i seconds.

It is guaranteed that each of the n cities has **at least one** transport cannon planted inside it, and that no two cannons from the same city are initially pointing at the same city (that is, all pairs (a_i, b_i) are distinct).

Output

Print n lines, each line should contain n integers.

The j-th integer in the i-th line should be equal to the minimum time required to reach city j from city i.

Examples

input

```
3 4
0 1 1
0 2 3
1 0 1
2 0 1
```

output

```
0 1 2
1 0 2
1 2 0
```

input

```
6 6
0 0 1
1 1 1
2 2 1
3 3 1
4 4 1
5 5 1
```

output

```
0 2 3 3 4 4
4 0 2 3 3 4
4 4 0 2 3 3
3 4 4 0 2 3
3 3 4 4 0 2
2 3 3 4 4 0
```

input

```
4 5
0 1 1
1 3 2
2 2 10
3 0 1
0 0 2
```

output

```
0 1 2 3
3 0 3 2
12 13 0 11
1 2 2 0
```

Note

In the first example one possible path for going from 0 to 2 would be:

1. Stay inside 0 and do nothing for 1 second.
2. Use the first cannon and land at 2 after 1 second.

Note that: we could have used the second cannon in 0-th second but it would have taken us 3 seconds to reach city 2 in that case.