A. Constructive Problems

```
Input file:     standard input
Output file:    standard output
Time limit:     1 second
Memory limit:   256 megabytes
```

Gridlandia has been hit by flooding and now has to reconstruct all of it's cities. Gridlandia can be described by an $n \times m$ matrix.

Initially, all of its cities are in economic collapse. The government can choose to rebuild certain cities. Additionally, any collapsed city which has at least one vertically neighboring rebuilt city and at least one horizontally neighboring rebuilt city can ask for aid from them and become rebuilt **without help from the government**. More formally, collapsed city positioned in (i, j) can become rebuilt if **both** of the following conditions are satisfied:

- At least one of cities with positions $(i+1, j)$ and $(i-1, j)$ is rebuilt;
- At least one of cities with positions $(i, j+1)$ and $(i, j-1)$ is rebuilt.

If the city is located on the border of the matrix and has only one horizontally or vertically neighbouring city, then we consider only that city.

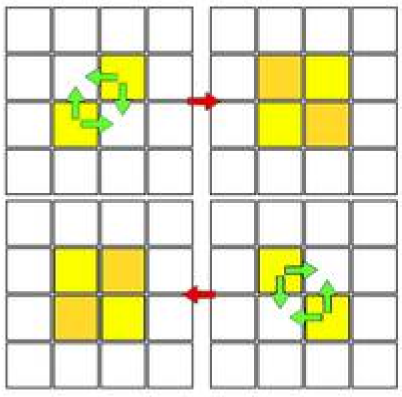


Illustration of two possible ways cities can be rebuilt by adjacent aid. White cells are collapsed cities, yellow cells are initially rebuilt cities (either by the government or adjacent aid), and orange cells are rebuilt cities after adjacent aid.

The government wants to know the minimum number of cities it has to rebuild such that **after some time** all the cities can be rebuild.

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 only line of each test case contains two integers n and m $(2 \leq n, m \leq 100)$ — the sizes of Gridlandia.

Output

For each test case, output a single integer — the minimum number of cities the government needs to rebuild.

Standard Input	Standard Output

3
2 2
5 7
3 2
	2
	7
	3

Note

In the first test case, it's enough for the government to rebuild cities $(1, 2)$ and $(2, 1)$.

In the second test case, it's enough for the government to rebuild cities $(1, 4)$, $(2, 2)$, $(3, 1)$, $(3, 6)$, $(4, 3)$, $(5, 5)$, $(5, 7)$.

,

B. Begginer's Zelda

```
Input file:     standard input
Output file:    standard output
Time limit:     1 second
Memory limit:   256 megabytes
```

You are given a tree[†]. In one *zelda-operation* you can do follows:

- Choose two vertices of the tree u and v;
- Compress all the vertices on the path from u to v into one vertex. In other words, all the vertices on path from u to v will be erased from the tree, a new vertex w will be created. Then every vertex s that had an edge to some vertex on the path from u to v will have an edge to the vertex w.

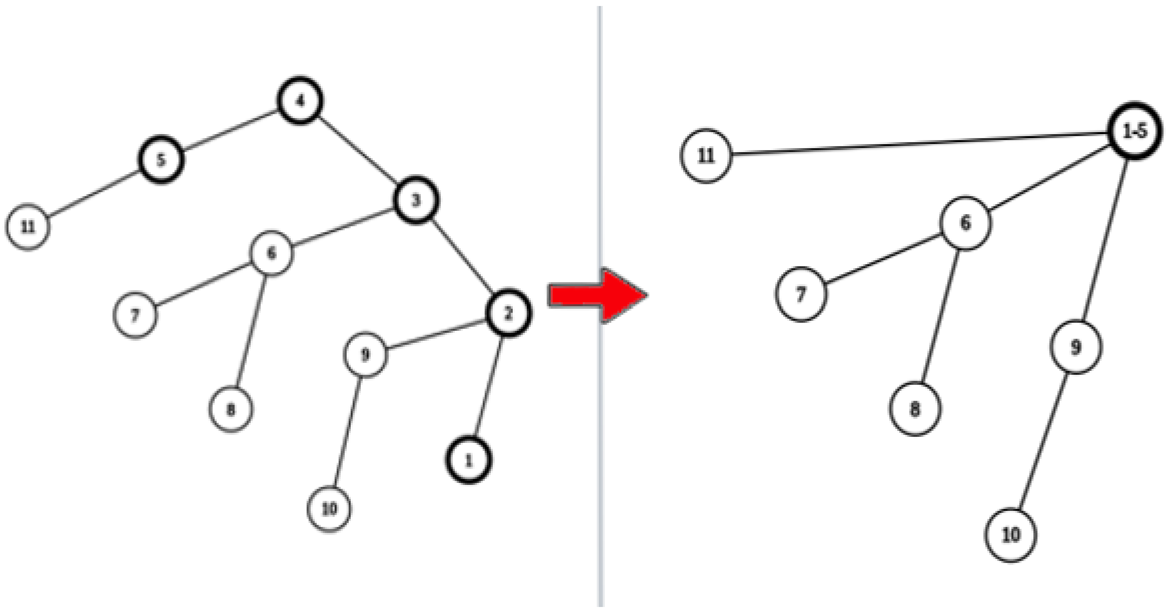


Illustration of a zelda-operation performed for vertices 1 and 5.

Determine the minimum number of zelda-operations required for the tree to have only one vertex.

[†] A tree is a connected acyclic undirected graph.

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 a single integer n ($2 \le n \le 10^5$) — the number of vertices.

i-th of the next $n-1$ lines contains two integers u_i and v_i ($1 \le u_i, v_i \le n, u_i \ne v_i$) — the numbers of vertices connected by the i-th edge.

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 10^5.

Output

For each test case, output a single integer — the minimum number of zelda-operations required for the tree to have only one vertex.

Standard Input	Standard Output

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

```
1
3
2
2
```

Note

In the first test case, it's enough to perform one zelda-operation for vertices 2 and 4.

In the second test case, we can perform the following zelda-operations:

1. $u = 2, v = 1$. Let the resulting added vertex be labeled as $w = 10$;
2. $u = 4, v = 9$. Let the resulting added vertex be labeled as $w = 11$;
3. $u = 8, v = 10$. After this operation, the tree consists of a single vertex.

,

C. Largest Subsequence

```
Input file:     standard input
Output file:    standard output
Time limit:     1 second
Memory limit:   256 megabytes
```

Given is a string s of length n. In one operation you can select the lexicographically largest[†] subsequence of string s and cyclic shift it to the right[‡].

Your task is to calculate the minimum number of operations it would take for s to become sorted, or report that it never reaches a sorted state.

[†] A string a is lexicographically smaller than a string b if and only if one of the following holds:

- a is a prefix of b, but $a \neq b$;
- In the first position where a and b differ, the string a has a letter that appears earlier in the alphabet than the corresponding letter in b.

[‡] By cyclic shifting the string $t_1 t_2 \ldots t_m$ to the right, we get the string $t_m t_1 \ldots t_{m-1}$.

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 a single integer n ($1 \le n \le 2 \cdot 10^5$) — the length of the string s.

The second line of each test case contains a single string s of length n, consisting of lowercase English letters.

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

Output

For each test case, output a single integer — the minimum number of operations required to make s sorted, or -1 if it's impossible.

Standard Input	Standard Output
6 5 aaabc 3 acb 3 bac 4 zbca 15 czddeneeeemigec 13 cdefmopqsvxzz	0 1 -1 2 6 0

Note

In the first test case, the string s is already sorted, so we need no operations.

In the second test case, doing one operation, we will select `cb` and cyclic shift it. The string s is now `abc` which is sorted.

In the third test case, s cannot be sorted.

In the fourth test case we will perform the following operations:

- The lexicographically largest subsequence is `zca`. Then s becomes `abzc`.
- The lexicographically largest subsequence is `zc`. Then s becomes `abcz`. The string becomes sorted.

Thus, we need 2 operations.

,

D. Cyclic MEX

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   512 megabytes
```

For an array a, define its *cost* as $\sum_{i=1}^{n} \operatorname{mex}^{\dagger}([a_1, a_2, \ldots, a_i])$.

You are given a permutation[‡] p of the set $\{0, 1, 2, \ldots, n-1\}$. Find the maximum cost across all cyclic shifts of p.

[†] $\operatorname{mex}([b_1, b_2, \ldots, b_m])$ is the smallest non-negative integer x such that x does not occur among $b_1, b_2, \ldots, b_m$.

[‡] A permutation of the set $\{0, 1, 2, \ldots, n-1\}$ is an array consisting of n distinct integers from 0 to $n-1$ in arbitrary order. For example, $[1, 2, 0, 4, 3]$ is a permutation, but $[0, 1, 1]$ is not a permutation (1 appears twice in the array), and $[0, 2, 3]$ is also not a permutation ($n = 3$ but there is 3 in the array).

Input

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

The first line of each test case contains a single integer n ($1 \le n \le 10^6$) — the length of the permutation p.

The second line of each test case contain n distinct integers $p_1, p_2, \ldots, p_n$ ($0 \le p_i < n$) — the elements of the permutation p.

It is guaranteed that sum of n over all test cases does not exceed 10^6.

Output

For each test case, output a single integer — the maximum cost across all cyclic shifts of p.

Standard Input	Standard Output

4
6
5 4 3 2 1 0
3
2 1 0
8
2 3 6 7 0 1 4 5
1
0
	15
	5
	31
	1

Note

In the first test case, the cyclic shift that yields the maximum cost is $[2, 1, 0, 5, 4, 3]$ with cost $0 + 0 + 3 + 3 + 3 + 6 = 15$.

In the second test case, the cyclic shift that yields the maximum cost is $[0, 2, 1]$ with cost $1 + 1 + 3 = 5$.

,

E. One-X

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

In this sad world full of imperfections, ugly segment trees exist.

A segment tree is a tree where each node represents a segment and has its number. A segment tree for an array of n elements can be built in a recursive manner. Let's say function $\text{build}(v, l, r)$ builds the segment tree rooted in the node with number v and it corresponds to the segment $[l, r]$.

Now let's define $\text{build}(v, l, r)$:

- If $l = r$, this node v is a leaf so we stop adding more edges
- Else, we add the edges $(v, 2v)$ and $(v, 2v + 1)$. Let $m = \lfloor \frac{l+r}{2} \rfloor$. Then we call $\text{build}(2v, l, m)$ and $\text{build}(2v + 1, m + 1, r)$.

So, the whole tree is built by calling $\text{build}(1, 1, n)$.

Now Ibti will construct a segment tree for an array with n elements. He wants to find the sum of $\text{lca}^{\dagger}(S)$, where S is a non-empty subset of **leaves**. Notice that there are exactly $2^n - 1$ possible subsets. Since this sum can be very large, output it modulo $998\,244\,353$.

† $\text{lca}(S)$ is the number of the least common ancestor for the nodes that are in S.

Input

Each test consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10^3$) — the number of test cases. The description of the test cases follows.

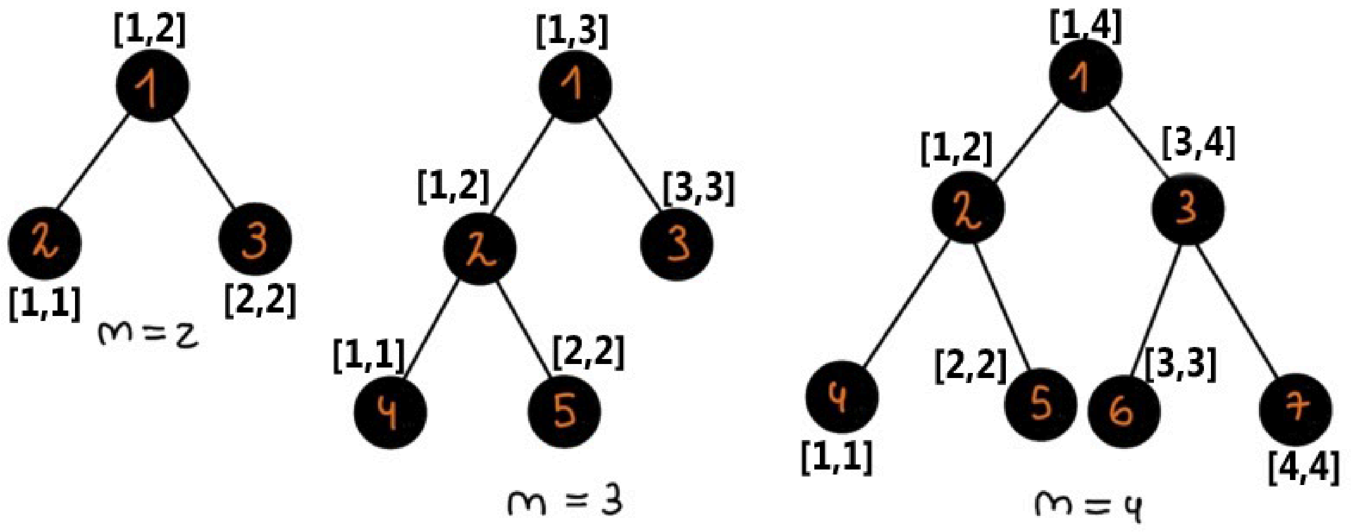
The first line of each test case contains a single integer n ($2 \le n \le 10^{18}$) — the length of the array for which the segment tree is built.

Output

For each test case, output a single integer — the required sum modulo $998\,244\,353$.

Standard Input	Standard Output
5 2 3 4 5 53278	6 17 36 69 593324855

Note



In the first test case:

Let's look at all subsets of leaves.

- $\text{lca}(\{2\}) = 2;$
- $\text{lca}(\{3\}) = 3;$
- $\text{lca}(\{2, 3\}) = 1.$

Thus, the answer is $2 + 3 + 1 = 6$.

In the second test case:

Let's look at all subsets of leaves.

- $\text{lca}(\{4\}) = 4;$
- $\text{lca}(\{5\}) = 5;$
- $\text{lca}(\{3\}) = 3;$
- $\text{lca}(\{4, 5\}) = 2;$
- $\text{lca}(\{4, 3\}) = 1;$
- $\text{lca}(\{5, 3\}) = 1;$
- $\text{lca}(\{4, 5, 3\}) = 1;$

Thus, the answer is $4 + 5 + 3 + 2 + 1 + 1 + 1 = 17$.

,

F. Field Should Not Be Empty

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

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

We call index x *good* if for all $y < x$ it holds that $p_y < p_x$ and for all $y > x$ it holds that $p_y > p_x$. We call $f(p)$ the number of good indices in p.

You can perform the following operation: pick 2 **distinct** indices i and j and swap elements p_i and p_j.

Find the maximum value of $f(p)$ after applying the aforementioned operation **exactly once**.

[†] 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

Each test consists of multiple test cases. The first line of contains a single integer t ($1 \le t \le 2 \cdot 10^4$) — the number of test cases. The description of the test cases follows.

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

The second line of each test case contain n distinct integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$) — the elements of the permutation p.

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

Output

For each test case, output a single integer — the maximum value of $f(p)$ after performing the operation exactly once.

Standard Input	Standard Output

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

Note

In the first test case, $p = [1, 2, 3, 4, 5]$ and $f(p) = 5$ which is already maximum possible. But must perform the operation anyway. We can get $f(p) = 3$ by choosing $i = 1$ and $j = 2$ which makes $p = [2, 1, 3, 4, 5]$.

In the second test case, we can transform p into $[1, 2, 3, 4, 5]$ by choosing $i = 1$ and $j = 2$. Thus $f(p) = 5$.