ICPC Kanpur Regionals 2025-26

A - Array Balancing

You are given two integer arrays a and b of the same **even** length n, where $0 \le a_i \le n$ and $0 \le b_i \le n$ for all i.
You want to transform array a into array b using the following operation, which you may apply at most $2 \cdot n$ times (possibly zero):
Choose a subset S of indices of size exactly $\frac{n}{2}$, that is, $S \subseteq \{1, 2, \ldots, n\}$ and $|S| = \frac{n}{2}$. Then, for every i from 1 to n:

- If $i \in S$, assign $a_i := a_i + 1$.
- If $i \notin S$, assign $a_i := a_i - 1$.

Determine whether it is possible to transform a into b using at most $2 \cdot n$ operations. If it is possible, you also need to print the operations. You do not need to minimize the number of operations.
It can be proven that if it is possible, you can do it in at most $2 \cdot n$ moves. If it is not possible, output -1.

Input Format

- The first line contains a single integer t, denoting the number of test cases.
- For each test case:
  - The first line contains a single integer n, denoting the length of the arrays.
  - The second line contains n integers $a_1, a_2, \ldots, a_n$, representing the initial array.
  - The third line contains n integers $b_1, b_2, \ldots, b_n$, representing the target array.

Output Format

For each test case:

- If it is not possible to transform a into b using at most $2 \cdot n$ operations, output a single line containing -1.
- Otherwise, output the number of operations k $(0 \le k \le 2 \cdot n)$ on the first line. Then output k lines, where the i-th line contains $\frac{n}{2}$ distinct integers $s_1, s_2, \ldots, s_{\frac{n}{2}}$ $(1 \le s_j \le n)$, the indices in the subset S chosen for the i-th operation, in any order.
- If there are multiple valid sequences of operations, you can output any of them.

Constraints

- $1 \le t \le 10^4$
- $2 \le n \le 1000$
- n is even
- $0 \le a_i \le n$
- $0 \le b_i \le n$
- The sum of n^2 over all test cases does not exceed 10^6

Sample Input 1

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

Sample Output 1

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

Explanation

- **Test case 1:** It can be proven that it is not possible to transform $a = [1, 1]$ into $b = [2, 2]$ using the given operations.
- **Test case 2:** We can select $S = \{1, 2\}$. After performing one operation, we obtain $a = b$.
- **Test case 3:** The arrays are already equal, so no operations are needed.

B - Bicycle Coloring

You are given a **bicycle wheel graph**.
A bicycle wheel graph consists of the following components:

- A simple cycle, i.e., a cycle with no repeated vertices except the first and last.
- Several *spokes*, where each spoke is a simple path consisting of at least one edge that connects a vertex on the cycle to the central node. Further, every vertex on the spoke other than the central node and the cycle vertex must have degree exactly equal to 2.
- Each vertex on the cycle is connected to at most one spoke.
- There is at least one spoke in the graph.

More formally, a bicycle wheel graph is a graph that can be decomposed into two parts:

- A simple cycle $(v_1, v_2, \ldots, v_k)$ of length $k \geq 3$.
- A *rooted* tree, with the root being a special vertex v^* (this is the central node described above), that satisfies the following properties:
  - The tree has at least one edge.
  - For any pair of leaf vertices (x, y) in the tree such that $x \neq y$, the unique simple path between x and y in the tree contains the root v^*.

The tree and the cycle have the following interactions:

- Every edge of the graph must belong to either the cycle or the rooted tree, but not to both.
- Every leaf vertex of the tree must lie on the cycle, i.e. must equal one of the vertices v_i. Note that the vertex v^* is not considered to be a leaf, even if it has degree 1.
- Every non-leaf vertex of the tree, including v^*, does not lie on the cycle.

You can also refer to the images in the explanation section for more clarity.
The graph is provided as input, but you are not given any information about which node is the central node or which edges belong to the spokes.
Your task is to color the graph using the minimum possible number of colors such that no two adjacent nodes share the same color. Additionally, you must output the color assigned to each node.

Input Format

- The first line contains a single integer t, denoting the number of test cases.
- For each test case:
  - The first line contains two integers n and m, denoting the number of nodes and edges in the graph.
  - The next m lines each contain two integers u and v, denoting an edge between nodes u and v.

Output Format

For each test case, output two lines:
The first line contains a single integer x, the minimum number of colors needed to color the graph such that no two adjacent nodes have the same color.
The second line contains n integers $c_1, c_2, \ldots, c_n$ $(1 \leq c_i \leq x)$, the color assigned to node i, where c_i denotes the color of node i.
If there are multiple valid colorings, you can output any of them.

Constraints

- $1 \leq t \leq 10^4$
- $1 \leq n \leq 2 \cdot 10^5$
- $n - 1 \leq m \leq 2 \cdot 10^5$
- $1 \leq u, v \leq n$, $u \neq v$
- It is guaranteed that the graph is a bicycle wheel graph.
- There is at least one spoke.
- The sum of m over all test cases does not exceed $2 \cdot 10^5$.

Sample Input 1

```
4
4 4
1 2
2 3
3 1
4 1
5 6
1 2
2 3
3 4
4 1
5 1
5 3
7 12
1 2
2 3
3 4
4 5
5 6
6 1
7 1
7 2
7 3
7 4
7 5
7 6
7 9
1 2
2 3
1 3
1 4
4 5
2 6
6 5
3 7
7 5
```

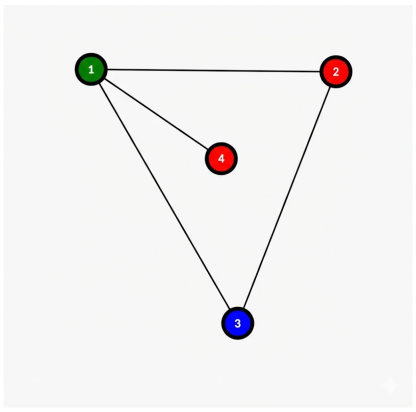
Sample Output 1

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

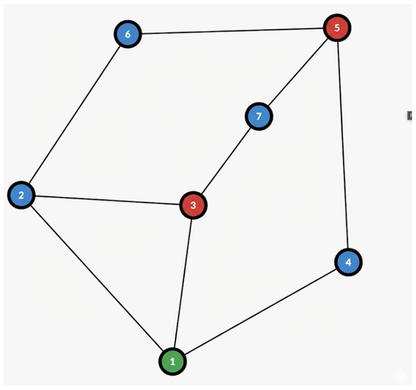
```
3
2 3 2 3 2 3 1
3
1 2 3 2 3 2 2
```

Explanation

- **Test case 1:** The central node is 4.



- **Test case 4:** The central node is 3.



C - Edge Elimination

You're given a tree (a simple connected acyclic graph) on n vertices. The i-th edge of the tree is between vertices u_i and v_i, and has the value w_i written on it.
Initially, your score is 0.
You can repeatedly perform the following operation:

- Choose an edge (u, v, w) that exists in the tree. This is an edge between u and v with a value of w.
- Add its value w to your score.
- Delete the edge from the tree.
- After deletion, you will be left with two disjoint trees. You must choose to keep one of them and discard the other one.

This process will then repeat until there are no more edges that can be deleted in your tree.
Find the minimum possible final score that can be attained.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of multiple lines of input:
  - The first line of each test case contains a single integer n, denoting the number of vertices in the tree.
  - The next $n-1$ lines describe the edges. The i-th of these $n-1$ lines contains three space-separated integers u_i, v_i, and w_i, denoting an edge between u_i and v_i with weight w_i.

Output Format

For each test case, output a single integer, the minimum possible final score that can be attained.

Constraints

- $1 \leq t \leq 3 \cdot 10^4$
- $1 \leq n \leq 2 \cdot 10^5$
- $1 \leq u_i, v_i \leq n$
- The input edges describe a tree on the vertex set $\{1, 2, \ldots, n\}$.
- $1 \leq w_i \leq 10^9$
- The sum of n over all test cases won't exceed $2 \cdot 10^5$.

Sample Input 1

```
2
2
1 2 5
3
1 3 8
3 2 4
```

Sample Output 1

```
5
4
```

Explanation

- **Test case** 1**:** There is only one edge, with a weight of 5.
- **Test case** 2**:** Delete the edge between 2 and 3, which has a weight of 4. This results in two trees: $(1, 3)$ with an edge between them, and (2) with no edges. We keep the second tree and discard the first; and the process ends since there are now no more edges. The final score is 4, and this is optimal.

D - Empty Intersection

You are given two arrays a and b, consisting of n and m integers respectively.
You can perform the following operation:

- Select an element e ($e \geq 2$) from either a or b.
- Remove exactly one occurrence of e from the array from which it was taken.
- Choose two integers x and y such that $\min(x, y) \geq 1$ and $x + y = e$.
- Insert x and y into the same array from which e was chosen.

Your goal is to ensure that there doesn't exist any integer v which is contained in both a and b simultaneously.
Determine if it is possible to achieve this goal by performing the given operation several times (possibly, zero times).

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of three lines of input:
  - The first line of each test case contains two space-separated integers n and m, denoting the lengths of the arrays a and b, respectively.
  - The second line contains n space-separated integers $a_1, a_2, \ldots, a_n$.
  - The third line contains m space-separated integers $b_1, b_2, \ldots, b_m$.

Output Format

For each test case, output `YES` if it is possible to achieve the goal, and `NO` otherwise.
You can output the answer in any case (upper or lower). For example, the strings `yEs`, `yes`, `Yes`, and `YES` will be recognized as positive responses.

Constraints

- $1 \leq t \leq 10^4$
- $1 \leq n, m \leq 2 \cdot 10^5$
- $1 \leq a_i, b_i \leq 10^9$
- The sum of n and the sum of m over all test cases each won't exceed $2 \cdot 10^5$.

Sample Input 1

```
3
5 6
1 2 3 4 5
6 7 8 9 10 11
1 1
1
1
2 1
5 6
5
```

Sample Output 1

```
Yes
No
Yes
```

Explanation

- **Test case** 1**:** The current arrays are already valid.
- **Test case** 2**:** No further operations can be performed. Therefore, 1 will be present in both arrays.
- **Test case** 3**:** We can select $e = 5$ from array b, remove it, and insert 2 and 3. Thus, we obtain $a = [5, 6]$ and $b = [2, 3]$. The intersection is empty.

E - Merging Maximum

You're given a permutation p of $\{1, 2, \ldots, n\}$. That is, p is an array of length n containing every integer from 1 to n exactly once each.
You can perform the following operation on it:

- Choose an index i $(1 < i < |p|)$. Here, $|p|$ denotes the current length of p.
- Set $p_i = \max(p_{i-1}, p_i, p_{i+1})$.
- Delete p_{i-1} and p_{i+1} from p. The remaining elements are then re-indexed to start from 1.

For example, if $p = [3, 1, 4, 2, 5]$, performing the operation with $i = 4$ would do the following:

- First, set $p_4 = \max(p_3, p_4, p_5) = 5$. So, $p = [3, 1, 4, 5, 5]$.
- Then, delete p_3 and p_5. So, $p = [3, 1, 5]$ finally.

Note that each such operation reduces the length of p by 2. In particular, if the length of p is 1 or 2, it cannot be modified by this operation at all.
Find the number of arrays that can be reached by performing this operation zero or more times starting from p. Since this value might be large, you only need to find it modulo $998\,244\,353$.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of two lines of input:
  - The first line of each test case contains a single integer n, denoting the length of the permutation.
  - The second line contains n space-separated integers $p_1, p_2, \ldots, p_n$.

Output Format

For each test case, output a single integer, the number of arrays that can be reached by performing the operation zero or more times on p, modulo $998\,244\,353$.

Constraints

- $1 \le t \le 10^5$
- $1 \le n \le 5000$
- $1 \le p_i \le n$
- $p_i \ne p_j$ for $i \ne j$
- The sum of n^2 over all test cases won't exceed 5000^2.

Sample Input 1

```
5
2
2 1
3
2 1 3
5
3 1 4 5 2
10
1 2 3 4 5 6 7 8 9 10
15
5 7 6 12 2 13 10 1 11 4 3 9 14 15 8
```

Sample Output 1

```
1
2
5
55
365
```

Explanation

- **Test case** 1**:** No operations can be performed on p.
- **Test case** 2**:** There are two possibilities:
  1. Perform no operations. This results in the final array $[2, 1, 3]$.
  2. Perform one operation, choosing $i = 2$. This results in the array $[3]$.

  These are the only reachable arrays.

F - Minimize Minimum

You are given an array a of length n, where n is even. You need to partition a into two disjoint subsequences[†] s_1 and s_2 of length $\frac{n}{2}$ each such that the minimum value of $|s_1[i] - s_2[i]|$ over $1 \le i \le \frac{n}{2}$ is as small as possible.

More formally, you need to choose two disjoint subsequences s_1 and s_2 from a, each containing exactly $\frac{n}{2}$ elements. Among all such partitions, find the one that minimizes $\min_{1 \le i \le \frac{n}{2}} |s_1[i] - s_2[i]|$, and output this minimum value.

Here, $|x|$ refers to the absolute value of x. For example, $|2| = 2$, $|-3| = 3$, $|0| = 0$.

[†] An array b is said to be a subsequence of another array a if b can be obtained from a by deleting some of its elements, without changing the order of the remaining elements. For example, if $a = [2, 1, 3, 2]$, some examples of its subsequences are $[2]$, $[1, 3]$, $[2, 3, 2]$, $[2, 2]$. However, $[4, 2]$ and $[1, 2, 2]$ are *not* subsequences of this array.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of two lines of input:
  - The first line of each test case contains a single integer n, denoting the length of the array a.
  - The second line contains n space-separated integers $a_1, a_2, \ldots, a_n$.

Output Format

For each test case, output a single integer, the minimum possible value of $\min_{1 \le i \le \frac{n}{2}} |s_1[i] - s_2[i]|$ over all valid partitions of a into (s_1, s_2).

Constraints

- $1 \le t \le 10^4$
- $2 \le n \le 2 \cdot 10^5$
- n is even
- $1 \le a_i \le 10^9$
- The sum of n over all test cases won't exceed $2 \cdot 10^5$.

Sample Input 1

```
3
2
1 5
4
40 40 2 30
6
5 10 2 3 30 40
```

Sample Output 1

```
4
0
1
```

Explanation

- **Test case** 1**:** Choose $s_1 = [1]$ and $s_2 = [5]$. This gives us a difference of $|1 - 5| = 4$. No other difference is possible since this is the only possible split into subsequences.
- **Test case** 2**:** Choose $s_1 = [40, 30]$ (indices 1 and 4), and $s_2 = [40, 2]$ (indices 2 and 3). This gives us $|s_1[1] - s_2[1]| = |40 - 40| = 0$, which is clearly optimal.
- **Test case** 3**:** Choose $s_1 = [5, 2, 30]$ (indices 1, 3, and 5), and $s_2 = [10, 3, 40]$ (indices 2, 4, and 6). This gives us $|s_1[2] - s_2[2]| = |2 - 3| = 1$, which is clearly optimal.

G - Penny Pinching

For an array $b = [b_1, b_2, \ldots, b_m]$ of length m and a binary string t of length p such that $p < m$, we define $\text{score}(b, t)$ as follows.
Iterate $i = 1, 2, \ldots, p$ in order:

- If $t_i = 0$, delete b_1 from b.
- Otherwise, delete b_2 from b.
- The remaining elements are then re-indexed to start from 1.

After all elements of t have been processed, b will have exactly $(m - p)$ elements remaining. $\text{score}(b, t)$ is defined to be the minimum among these remaining elements.

You would like to have a higher score. To achieve that, you can perform the following operation:

- Choose an index i and flip t_i (i.e. change 0 to 1 and vice versa).

This operation can be done however many times you like (including 0).

After performing all the flip operations you would like, you will receive coins equal to $(\text{score}(b, t) - k)$, where k denotes the number of flip operations performed. Essentially, each flip operation costs one coin. Note that $\text{score}(b, t)$ here is computed based on the string t after *all* the flips have been applied to it.

We then define $f(b, t)$ to be the maximum number of coins you can receive as a result of performing the above operation optimally, given that you can freely choose the number of flips to perform and the indices to perform them on.

You're given an array a of length n and a binary string s of length $n - 1$. For each $i \in [1, n - 1]$, compute the value of $f(a, s[1, i])$, where $s[1, i] = s_1 s_2 \ldots s_i$ denotes the prefix of s of length i.

Each $f(a, s[1, i])$ value is to be computed independently, that is, for each i, assume that you start with the initial array a and a fresh copy of the string $s[1, i]$.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of three lines of input:
  - The first line of each test case contains a single integer n, denoting the length of the array a.
  - The second line contains n space-separated integers $a_1, \ldots, a_n$.
  - The third line contains a binary string s of length $n - 1$.

Output Format

For each test case, output $n - 1$ space-separated integers, the values of $f(a, s[1, 1])$, $f(a, s[1, 2])$, …, $f(a, s[1, n - 1])$ in order.

Constraints

- $1 \le t \le 2 \cdot 10^4$
- $2 \le n \le 2 \cdot 10^5$
- $1 \le a_i \le 10^9$
- s is a binary string of length $n - 1$
- The sum of n over all test cases won't exceed $2 \cdot 10^5$.

Sample Input 1

```
4
2
5 8
0
3
9 3 1
01
4
1 4 2 4
010
8
1 5 3 10 6 4 14 9
1100101
```

Sample Output 1

```
8
1 8
2 4 4
2 3 4 4 8 9 14
```

Explanation

- **Test case** 1**:** We have only $a = [5, 8]$ and $s =$ `0`. Since $s_1 = 0$, the first element of a must be deleted, leaving us with $a = [8]$. It's optimal to leave it at this and not do anything.
- **Test case** 2**:** We have $a = [9, 3, 1]$ and $s =$ `01`.
  - For $i = 1$, we consider the string `0`.
    - If no flips are performed, the resulting array is $[3, 1]$, and we receive $1 - 0 = 1$ coin.
    - If we flip the only character and the string `1`, the resulting array is $[9, 1]$ instead, and we receive $1 - 1 = 0$ coins.
    - Performing more than one flip is clearly suboptimal. So, the answer for $i = 1$ is $\max(1, 0) = 1$.
  - For $i = 2$, we consider the string `01`.
    - Here, it is optimal to spend one operation and turn the string into `11`. This turns the resulting array into $[9]$, giving us $9 - 1 = 8$ coins which is the largest we can obtain.

H - Prefix Game

Alice and Bob are playing a game. There are n piles of stones, with the piles satisfying the condition that $a_1 \geq a_2 \geq \cdots \geq a_n$ (that is, the piles are arranged in non-increasing order).

On their turn, a player can choose an index i ($1 \leq i \leq n$) and a **positive** integer x, and take x stones from each of the piles $a_1, a_2, \ldots, a_i$. However, after each turn, the condition $a_1 \geq a_2 \geq \cdots \geq a_n$ must still be satisfied.

The players take turns, with Alice going first. The player who cannot make a valid move loses. Both players play optimally.

Determine who wins the game.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of two lines of input:
  - The first line of each test case contains a single integer n, denoting the number of piles of stones.
  - The second line contains n space-separated integers $a_1, a_2, \ldots, a_n$.

Output Format

For each test case, output `Alice` if Alice wins, and `Bob` otherwise.

You can output the answer in any case (upper or lower). For example, the strings `bOb`, `bob`, `Bob`, and `BOB` will all be recognized as valid responses for cases where Bob is the winner.

Constraints

- $1 \leq t \leq 10^4$
- $1 \leq n \leq 2 \cdot 10^5$
- $1 \leq a_i \leq 10^9$
- $a_1 \geq a_2 \geq \ldots \geq a_n$
- The sum of n over all test cases won't exceed $2 \cdot 10^5$.

Sample Input 1

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

Sample Output 1

```
Bob
Alice
```

Explanation

- **Test case 1:** Alice has two possible first moves:
  - Choose $i = 1$ and $x = 1$, which will turn the piles into $[1, 1]$. Bob can win this by choosing $i = 2$ and $x = 1$ on his turn, after which Alice has no move.
  - Choose $i = 2$ and $x = 1$, which will turn the piles into $[1, 0]$. Bob can win this by choosing $i = 1$ and $x = 1$ on his turn, after which Alice has no move again.

So, Bob is able to win no matter what Alice does. Note that for example Alice cannot choose $i = 1$ and $x = 2$ on her turn, since that would turn the piles into $[0, 1]$ and break the non-increasing criterion.

- **Test case 2:** Alice can choose $i = 3$ and $x = 1$, turning the piles into $[0, 0, 0]$ and hence winning.

I - Prefix MEX Equation

Suppose you have two arrays x and y, both of length m. You can perform the following operation however many times you like:

- Choose an index i ($1 \leq i \leq m$), and swap x_i with y_i.

The pair of arrays (x, y) is said to be *good* if it's possible to make their prefix MEX arrays equal by performing several such swaps.
Note that:

- The MEX of a set of integers is the smallest non-negative integer that doesn't appear in the set. For example, $\text{MEX}([0,1,0]) = 2$, $\text{MEX}([3,2,1]) = 0$, $\text{MEX}([0,1,2,1]) = 3$.
- The prefix MEX array of the array x is an array p, of the same length as x, such that $p_i = \text{MEX}([x_1, x_2, \ldots, x_i])$ for each $1 \leq i \leq |x|$.

You're given two arrays a and b, both of length n.
Count the number of pairs (l, r) such that $1 \leq l \leq r \leq n$, and the pair of arrays $(a[l,r], b[l,r])$ is *good*.
Here, $a[l,r]$ denotes the subarray $[a_l, a_{l+1}, \ldots, a_r]$ of a, and similarly $b[l,r]$ denotes the subarray $[b_l, \ldots, b_r]$.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of three lines of input:
  - The first line of each test case contains a single integer n, denoting the length of the arrays a and b.
  - The second line contains n space-separated integers $a_1, a_2, \ldots, a_n$.
  - The third line contains n space-separated integers $b_1, b_2, \ldots, b_n$.

Output Format

For each test case, output a single integer, the number of pairs (l, r) such that $1 \leq l \leq r \leq n$, and the pair of arrays $(a[l,r], b[l,r])$ is good.

Constraints

- $1 \leq t \leq 10^5$
- $1 \leq n \leq 2 \cdot 10^5$
- $0 \leq a_i, b_i \leq n$
- The sum of n over all test cases won't exceed $2 \cdot 10^5$.

Sample Input 1

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

Sample Output 1

```
2
6
4
```

Explanation

- **Test case 1:** The valid pairs are $(1, 1)$ and $(2, 2)$.
- **Test case 2:** All the pairs are valid.
- **Test case 3:** The valid pairs are $(2, 2)$, $(3, 3)$, $(4, 4)$ and $(3, 4)$.

J - Recursion

Let n be a non-negative integer power of 2. We define a recurrence relation as follows:

$$f_n(x) = \begin{cases} 1, & \text{if } 1 \le x \le n, \\ \big(f_n(x-1) + f_n(x-n)\big) \bmod 2, & \text{if } x > n. \end{cases}$$

Given n and m, find $f_n(m)$.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of a single line of input, containing two space-separated integers n and m.

Output Format

For each test case, output a single integer, the value of $f_n(m)$.

Constraints

- $1 \le t \le 10^4$
- $1 \le n, m \le 10^{18}$
- n is a non-negative integral power of 2.

Sample Input 1

```
6
1 1
1 2
2 3
2 4
288230376151711744 685646715905549873
68719476736 803900416566661387
```

Sample Output 1

```
1
0
0
1
1
0
```

K - Tree Hail Conjecture

For a tree T with n vertices, we define the $(3n+1)$ operation as follows:

- Set T_1, T_2, and T_3 be copies of T, rename the nodes of T_2 by adding n to each original label, and then rename the nodes of T_3 by adding $2n$ to each original label. Then, replace T with T_1, T_2, and T_3.
- Add an edge (u_1, v_1) where $1 \leq u_1 \leq n$, and $n+1 \leq v_1 \leq 2n$. You can choose (u_1, v_1) freely.
- Add an edge (u_2, v_2) where $n+1 \leq u_2 \leq 2n$, and $2n+1 \leq v_2 \leq 3n$. You can choose (u_2, v_2) freely.
- Add a node labeled $3n+1$, and add an edge $(3n+1, v_3)$ where $1 \leq v_3 \leq 3n$. You can choose v_3 freely.

It can be shown that the result of the $(3n+1)$ operation is a tree with $3n+1$ vertices.
For a tree T, if there exists an edge that results in an equal number of nodes for the two connected components after removing this edge, the tree is called *good*.
You are given a tree with n vertices. You would like to make the tree *good*.
Find the minimum number of times the $(3n+1)$ operation needs to be performed to make the tree good, assuming your choices are optimal.

Input Format

- The first line of input will contain a single integer t, denoting the number of test cases.
- Each test case consists of multiple lines of input:
  - The first line of each test case contains a single integer n, denoting the number of vertices in the given tree.
  - The next $n-1$ lines describe the edges of the tree. The i-th of these $n-1$ lines contains two space-separated integers u_i and v_i, denoting an edge between u_i and v_i.

Output Format

For each test case, output a single integer representing the minimum number of operations to make the tree good.

Constraints

- $1 \leq t \leq 10^5$
- $1 \leq n \leq 2 \cdot 10^5$
- $1 \leq u_i, v_i \leq n$
- The input edges describe a tree on the vertex set $\{1, 2, \ldots, n\}$.
- The sum of n over all test cases won't exceed $2 \cdot 10^5$.

Sample Input 1

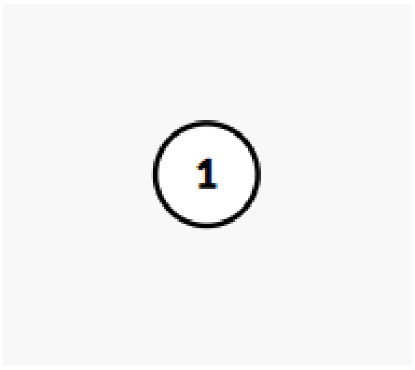
```
3
2
1 2
1
6
1 2
1 3
1 4
2 5
3 6
```

Sample Output 1

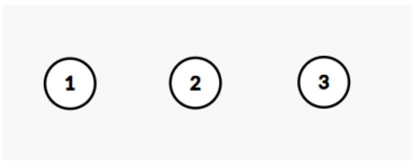
```
0
1
2
```

Explanation

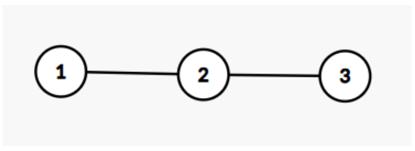
- **Test case 1:** The tree is already good, so the answer is 0.
- **Test case 2:** The minimum number of operations to make the tree good is 1. The following is a possible scheme.



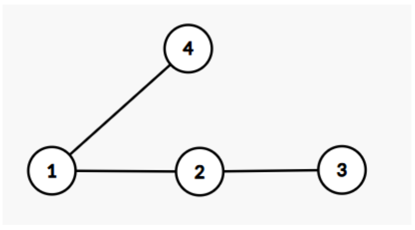
The original tree T.



$T$is replaced by trees $T_1, T_2,$ and T_3



You choose edges $(1, 2)$and $(2, 3)$.



A node labeled 4is added; you choose the edge $(4, 1)$. Now the tree is good.