Codeforces Round #769 (Div. 2)

A. ABC

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

Recently, the students of School 179 have developed a unique algorithm, which takes in a binary string s as input. However, they soon found out that if some substring t of s is a palindrome of length greater than 1, the algorithm will work incorrectly. Can the students somehow reorder the characters of s so that the algorithm will work correctly on the string?

A binary string is a string where each character is either `0` or `1`.

A string a is a substring of a string b if a can be obtained from b by deletion of several (possibly, zero or all) characters from the beginning and several (possibly, zero or all) characters from the end.

A palindrome is a string that reads the same backwards as forwards.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 100$). Description of the test cases follows.

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

The second line of each test case contains the string s of length n consisting only of the characters `0` and `1`.

Output

For each test case, print YES (case-insensitive) if it is possible to reorder the characters of s so that there are no substrings that are a palindrome of length greater than 1, or `NO` (case-insensitive) otherwise.

Example

input
4 1 1 2 10 2 01 4 1010
output
YES YES YES NO

Note

In the first three test cases, the given strings do not contain palindromes of length greater than 1, so the answers are YES.

In the last test case, it is impossible to reorder the characters so that the string does not contain palindromes of length greater than 1, so the answer is `NO`.

B. Roof Construction

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

It has finally been decided to build a roof over the football field in School 179. Its construction will require placing n consecutive vertical pillars. Furthermore, the headmaster wants the heights of all the pillars to form a permutation p of integers from 0 to $n - 1$, where p_i is the height of the i-th pillar from the left $(1 \le i \le n)$.

As the chief, you know that the cost of construction of consecutive pillars is equal to **the maximum value of the bitwise XOR** of heights of all pairs of adjacent pillars. In other words, the cost of construction is equal to $\max\limits_{1 \le i \le n-1} p_i \oplus p_{i+1}$, where $\oplus$ denotes the bitwise XOR operation.

Find any sequence of pillar heights p of length n with the smallest construction cost.

In this problem, a permutation is an array consisting of n distinct integers from 0 to $n-1$ in arbitrary order. For example, $[2,3,1,0,4]$ is a permutation, but $[1,0,1]$ is not a permutation (1 appears twice in the array) and $[1,0,3]$ is also not a permutation ($n=3$, but 3 is in the array).

Input
Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). Description of the test cases follows.

The only line for each test case contains a single integer n ($2 \le n \le 2 \cdot 10^5$) — the number of pillars for the construction of the roof.

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

Output
For each test case print n integers $p_1, p_2, \ldots, p_n$ — the sequence of pillar heights with the smallest construction cost.

If there are multiple answers, print any of them.

Example

input
4 2 3 5 10
output
0 1 2 0 1 3 2 1 0 4 4 6 3 2 0 8 9 1 7 5

Note
For $n=2$ there are 2 sequences of pillar heights:

- $[0,1]$ — cost of construction is $0 \oplus 1 = 1$.
- $[1,0]$ — cost of construction is $1 \oplus 0 = 1$.

For $n=3$ there are 6 sequences of pillar heights:

- $[0,1,2]$ — cost of construction is $\max(0 \oplus 1, 1 \oplus 2) = \max(1,3) = 3$.
- $[0,2,1]$ — cost of construction is $\max(0 \oplus 2, 2 \oplus 1) = \max(2,3) = 3$.
- $[1,0,2]$ — cost of construction is $\max(1 \oplus 0, 0 \oplus 2) = \max(1,2) = 2$.
- $[1,2,0]$ — cost of construction is $\max(1 \oplus 2, 2 \oplus 0) = \max(3,2) = 3$.
- $[2,0,1]$ — cost of construction is $\max(2 \oplus 0, 0 \oplus 1) = \max(2,1) = 2$.
- $[2,1,0]$ — cost of construction is $\max(2 \oplus 1, 1 \oplus 0) = \max(3,1) = 3$.

C. Strange Test

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

Igor is in 11th grade. Tomorrow he will have to write an informatics test by the strictest teacher in the school, Pavel Denisovich.

Igor knows how the test will be conducted: first of all, the teacher will give each student two positive integers a and b ($a < b$). After that, the student can apply any of the following operations any number of times:

- $a := a + 1$ (increase a by 1),
- $b := b + 1$ (increase b by 1),
- $a := a \mid b$ (replace a with the bitwise OR of a and b).

To get full marks on the test, the student has to tell the teacher the minimum required number of operations to make a and b equal.

Igor already knows which numbers the teacher will give him. Help him figure out what is the minimum number of operations needed to make a equal to b.

Input
Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). Description of the test cases follows.

The only line for each test case contains two integers a and b ($1 \le a < b \le 10^6$).

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

Output
For each test case print one integer — the minimum required number of operations to make a and b equal.

Example

input
5 1 3 5 8 2 5 3 19 56678 164422

output
1 3 2 1 23329

Note

In the first test case, it is optimal to apply the third operation.

In the second test case, it is optimal to apply the first operation three times.

In the third test case, it is optimal to apply the second operation and then the third operation.

D. New Year Concert

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

New Year is just around the corner, which means that in School 179, preparations for the concert are in full swing.

There are n classes in the school, numbered from 1 to n, the i-th class has prepared a scene of length a_i minutes.

As the main one responsible for holding the concert, Idnar knows that if a concert has k scenes of lengths b_1, b_2, ..., b_k minutes, then the audience will get bored if there exist two integers l and r such that $1 \leq l \leq r \leq k$ and $\gcd(b_l, b_{l+1}, \ldots, b_{r-1}, b_r) = r - l + 1$, where $\gcd(b_l, b_{l+1}, \ldots, b_{r-1}, b_r)$ is equal to the greatest common divisor (GCD) of the numbers b_l, b_{l+1}, ..., b_{r-1}, b_r.

To avoid boring the audience, Idnar can ask any number of times (possibly zero) for the t-th class ($1 \leq t \leq k$) to make a new scene d minutes in length, where d can be **any positive integer**. Thus, after this operation, b_t is equal to d. Note that t and d can be different for each operation.

For a sequence of scene lengths b_1, b_2, ..., b_k, let $f(b)$ be the minimum number of classes Idnar has to ask to change their scene if he wants to avoid boring the audience.

Idnar hasn't decided which scenes will be allowed for the concert, so he wants to know the value of f for each non-empty prefix of a. In other words, Idnar wants to know the values of $f(a_1)$, $f(a_1,a_2)$, ..., $f(a_1,a_2,\ldots,a_n)$.

Input

The first line contains a single integer n ($1 \leq n \leq 2 \cdot 10^5$) — the number of classes in the school.

The second line contains n positive integers a_1, a_2, ..., a_n ($1 \leq a_i \leq 10^9$) — the lengths of the class scenes.

Output

Print a sequence of n integers in a single line — $f(a_1)$, $f(a_1,a_2)$, ..., $f(a_1,a_2,\ldots,a_n)$.

Examples

input
1 1

output
1

input
3 1 4 2

output
1 1 2

input
7 2 12 4 8 18 3 6

output

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

Note

In the first test we can change 1 to 2, so the answer is 1.

In the second test:

- $[1]$ can be changed into $[2]$,
- $[1, 4]$ can be changed into $[3, 4]$,
- $[1, 4, 2]$ can be changed into $[2, 3, 2]$.

E1. Distance Tree (easy version)

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

This version of the problem differs from the next one only in the constraint on n.

A tree is a connected undirected graph without cycles. A weighted tree has a weight assigned to each edge. The distance between two vertices is the minimum sum of weights on the path connecting them.

You are given a weighted tree with n vertices, each edge has a weight of 1. Denote $d(v)$ as the distance between vertex 1 and vertex v.

Let $f(x)$ be the minimum possible value of $\max\limits_{1 \le v \le n} d(v)$ if you can temporarily add an edge with weight x between any two vertices a and b $(1 \le a, b \le n)$. Note that after this operation, the graph is no longer a tree.

For each integer x from 1 to n, find $f(x)$.

Input

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

The first line of each test case contains a single integer n ($2 \le n \le 3000$).

Each of the next $n - 1$ lines contains two integers u and v ($1 \le u, v \le n$) indicating that there is an edge between vertices u and v. It is guaranteed that the given edges form a tree.

It is guaranteed that the sum of n over all test cases doesn't exceed 3000.

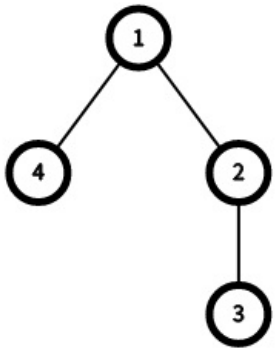
Output

For each test case, print n integers in a single line, x-th of which is equal to $f(x)$ for all x from 1 to n.

Example

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

Note



In the first testcase:

- For $x = 1$, we can an edge between vertices 1 and 3, then $d(1) = 0$ and $d(2) = d(3) = d(4) = 1$, so $f(1) = 1$.
- For $x \geq 2$, no matter which edge we add, $d(1) = 0$, $d(2) = d(4) = 1$ and $d(3) = 2$, so $f(x) = 2$.

E2. Distance Tree (hard version)

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

This version of the problem differs from the previous one only in the constraint on n.

A tree is a connected undirected graph without cycles. A weighted tree has a weight assigned to each edge. The distance between two vertices is the minimum sum of weights on the path connecting them.

You are given a weighted tree with n vertices, each edge has a weight of 1. Denote $d(v)$ as the distance between vertex 1 and vertex v.

Let $f(x)$ be the minimum possible value of $\max\limits_{1 \leq v \leq n} d(v)$ if you can temporarily add an edge with weight x between any two vertices a and b $(1 \leq a, b \leq n)$. Note that after this operation, the graph is no longer a tree.

For each integer x from 1 to n, find $f(x)$.

Input

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

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

Each of the next $n - 1$ lines contains two integers u and v ($1 \leq u, v \leq n$) indicating that there is an edge between vertices u and v. It is guaranteed that the given edges form a tree.

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

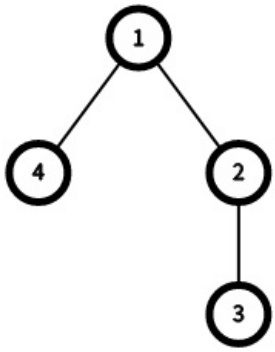
Output

For each test case, print n integers in a single line, x-th of which is equal to $f(x)$ for all x from 1 to n.

Example

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

Note



In the first testcase:

- For $x = 1$, we can an edge between vertices 1 and 3, then $d(1) = 0$ and $d(2) = d(3) = d(4) = 1$, so $f(1) = 1$.
- For $x \geq 2$, no matter which edge we add, $d(1) = 0$, $d(2) = d(4) = 1$ and $d(3) = 2$, so $f(x) = 2$.
