Codeforces Round #633 (Div. 1)

A. Powered Addition

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

You have an array a of length n. For every positive integer x you are going to perform the following operation during the x-th second:

- Select some distinct indices $i_1, i_2, \ldots, i_k$ which are between 1 and n inclusive, and add 2^{x-1} to each corresponding position of a. Formally, $a_{i_j} := a_{i_j} + 2^{x-1}$ for $j = 1, 2, \ldots, k$. **Note that you are allowed to not select any indices at all.**

You have to make a nondecreasing as fast as possible. Find the smallest number T such that you can make the array nondecreasing after at most T seconds.

Array a is nondecreasing if and only if $a_1 \le a_2 \le \ldots \le a_n$.

You have to answer t independent test cases.

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 single integer n ($1 \le n \le 10^5$) — the length of array a. It is guaranteed that the sum of values of n over all test cases in the input does not exceed 10^5.

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

Output

For each test case, print the minimum number of seconds in which you can make a nondecreasing.

Example

input

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

output

```
2
0
3
```

Note

In the first test case, if you select indices $3, 4$ at the 1-st second and 4 at the 2-nd second, then a will become $[1, 7, 7, 8]$. There are some other possible ways to make a nondecreasing in 2 seconds, but you can't do it faster.

In the second test case, a is already nondecreasing, so answer is 0.

In the third test case, if you do nothing at first 2 seconds and select index 2 at the 3-rd second, a will become $[0, 0]$.

B. Edge Weight Assignment

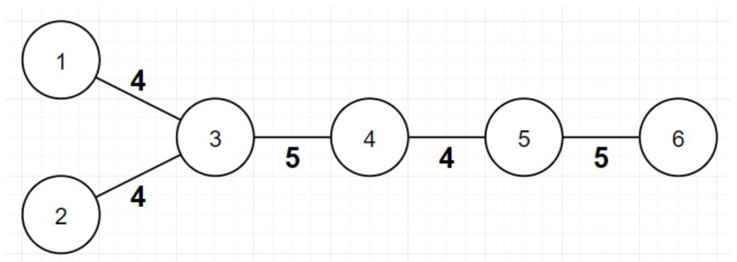
time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have unweighted tree of n vertices. You have to assign a **positive** weight to each edge so that the following condition would hold:

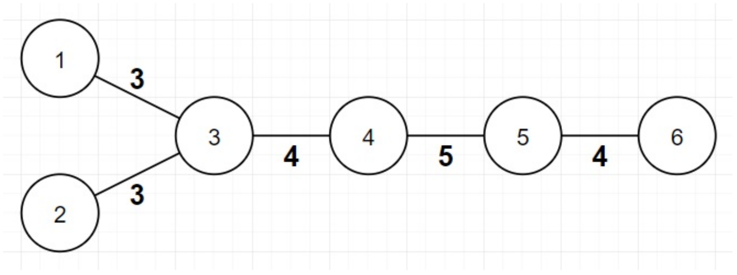
- For every two different leaves v_1 and v_2 of this tree, bitwise XOR of weights of all edges on the simple path between v_1 and v_2 has to be equal to 0.

Note that you can put **very large** positive integers (like $10^{(10^{10})}$).

It's guaranteed that such assignment always exists under given constraints. Now let's define f as **the number of distinct weights** in assignment.



In this example, assignment is valid, because bitwise XOR of all edge weights between every pair of leaves is 0. f value is 2 here, because there are 2 distinct edge weights(4 and 5).



In this example, assignment is invalid, because bitwise XOR of all edge weights between vertex 1 and vertex 6 $(3, 4, 5, 4)$ is not 0.

What are the minimum and the maximum possible values of f for the given tree? Find and print both.

Input

The first line contains integer n ($3 \le n \le 10^5$) — the number of vertices in given tree.

The i-th of the next $n-1$ lines contains two integers a_i and b_i ($1 \le a_i < b_i \le n$) — it means there is an edge between a_i and b_i. It is guaranteed that given graph forms tree of n vertices.

Output

Print two integers — the minimum and maximum possible value of f can be made from valid assignment of given tree. Note that it's always possible to make an assignment under given constraints.

Examples

input

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

output

```
1 4
```

input

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

output

```
3 3
```

input

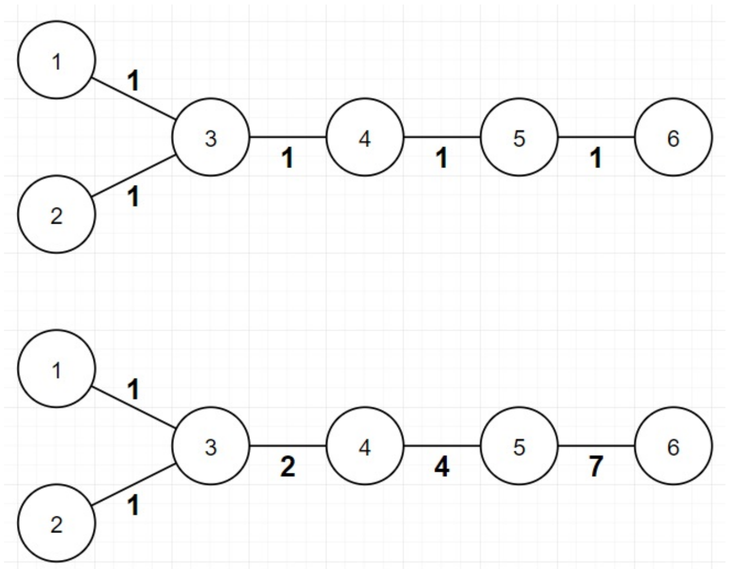
```
7
1 2
2 7
3 4
4 7
5 6
6 7
```

output

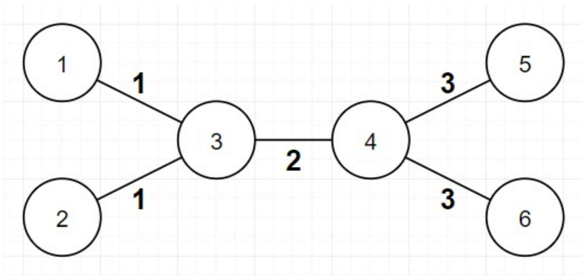
```
1 6
```

Note

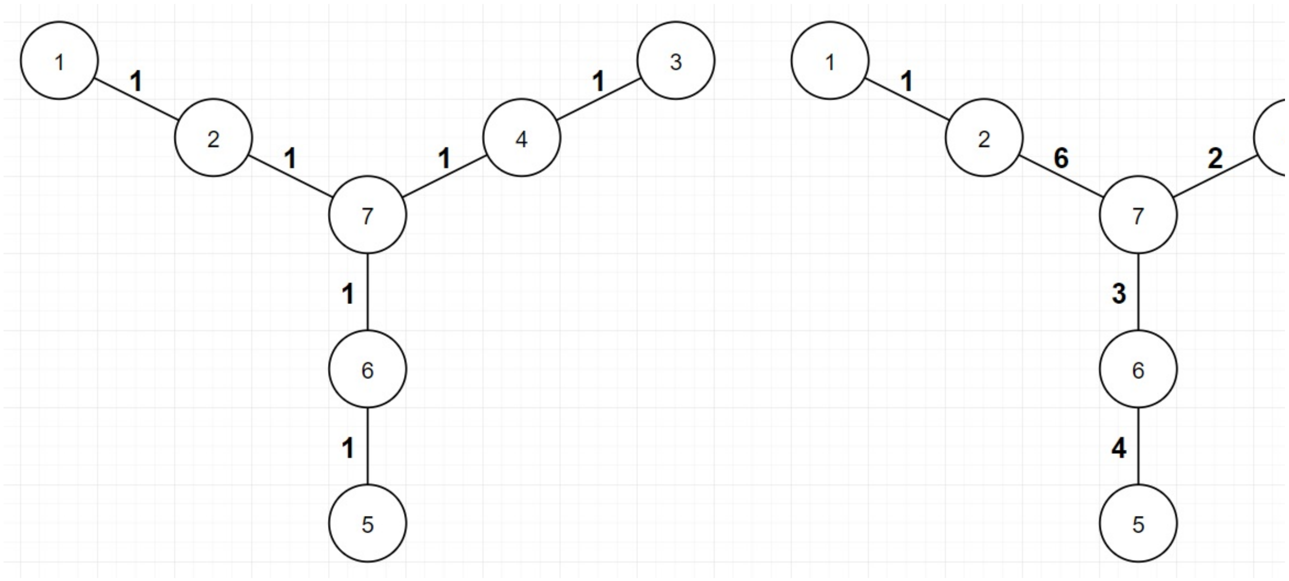
In the first example, possible assignments for each minimum and maximum are described in picture below. Of course, there are multiple possible assignments for each minimum and maximum.



In the second example, possible assignments for each minimum and maximum are described in picture below. The f value of valid assignment of this tree is always 3.



In the third example, possible assignments for each minimum and maximum are described in picture below. Of course, there are multiple possible assignments for each minimum and maximum.



C. Perfect Triples

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Consider the infinite sequence s of positive integers, created by repeating the following steps:

1. Find the lexicographically smallest triple of positive integers (a, b, c) such that
   - $a \oplus b \oplus c = 0$, where $\oplus$ denotes the bitwise XOR operation.
   - a, b, c are not in s.

   Here triple of integers (a_1, b_1, c_1) is considered to be lexicographically smaller than triple (a_2, b_2, c_2) if sequence $[a_1, b_1, c_1]$ is lexicographically smaller than sequence $[a_2, b_2, c_2]$.
2. Append a, b, c to s in this order.
3. Go back to the first step.

You have integer n. Find the n-th element of s.

You have to answer t independent test cases.

A sequence a is lexicographically smaller than a sequence b if in the first position where a and b differ, the sequence a has a smaller element than the corresponding element in b.

Input

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

Each of the next t lines contains a single integer n ($1 \le n \le 10^{16}$) — the position of the element you want to know.

Output

In each of the t lines, output the answer to the corresponding test case.

Example

input

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

output

```
1
2
3
4
8
12
5
10
15
```

Note

The first elements of s are $1, 2, 3, 4, 8, 12, 5, 10, 15, \dots$

D. Nested Rubber Bands

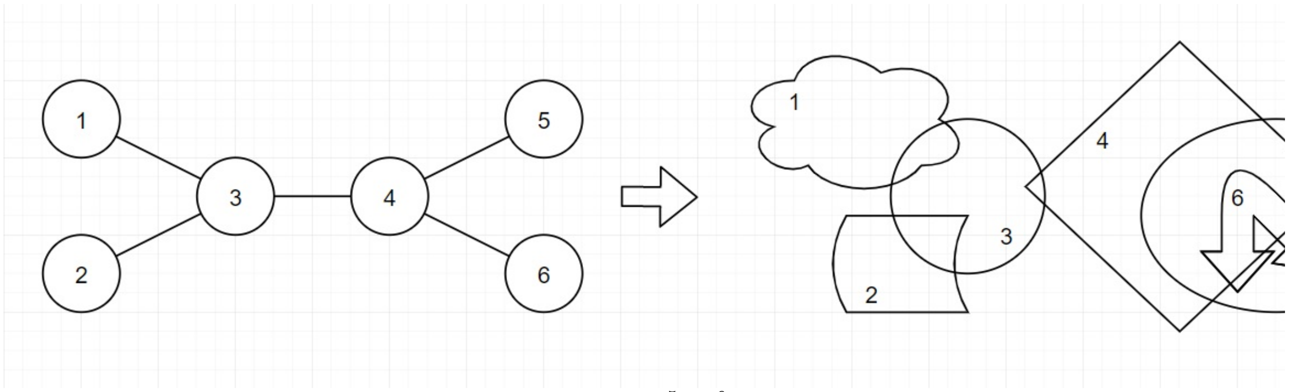
time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have a tree of n vertices. You are going to convert this tree into n rubber bands on infinitely large plane. Conversion rule follows:

- For every pair of vertices a and b, rubber bands a and b should intersect if and only if there is an edge exists between a and b in the tree.
- Shape of rubber bands must be a simple loop. In other words, rubber band is a loop which doesn't self-intersect.

Now let's define following things:

- Rubber band a **includes** rubber band b, if and only if rubber band b is in rubber band a's area, and they don't intersect each other.
- Sequence of rubber bands $a_1, a_2, \dots, a_k$ ($k \ge 2$) are **nested**, if and only if for all i ($2 \le i \le k$), a_{i-1} includes a_i.



This is an example of conversion. Note that rubber bands 5 and 6 are nested.

It can be proved that is it possible to make a conversion and sequence of nested rubber bands under given constraints.

What is the maximum length of sequence of nested rubber bands can be obtained from given tree? Find and print it.

Input

The first line contains integer n ($3 \le n \le 10^5$) — the number of vertices in tree.

The i-th of the next $n-1$ lines contains two integers a_i and b_i ($1 \le a_i < b_i \le n$) — it means there is an edge between a_i and b_i. It is guaranteed that given graph forms tree of n vertices.

Output

Print the answer.

Examples

input

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

output

```
4
```

input

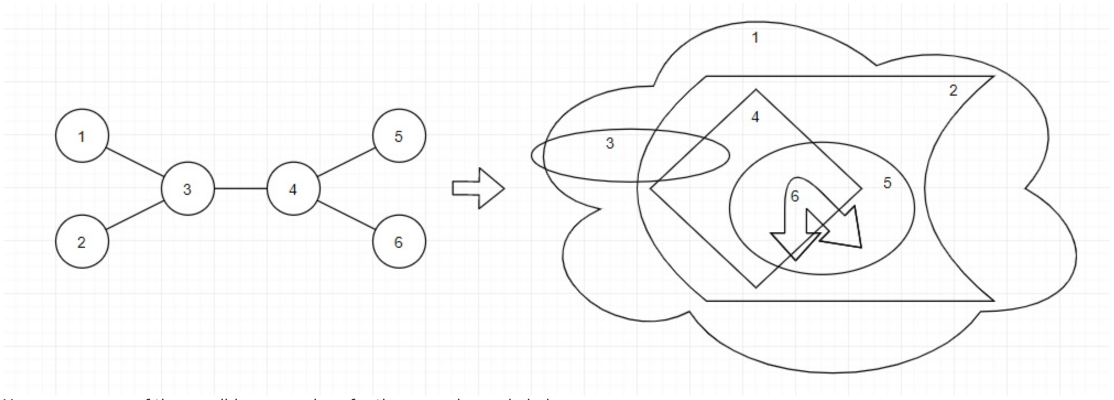
```
4
1 2
2 3
3 4
```

output

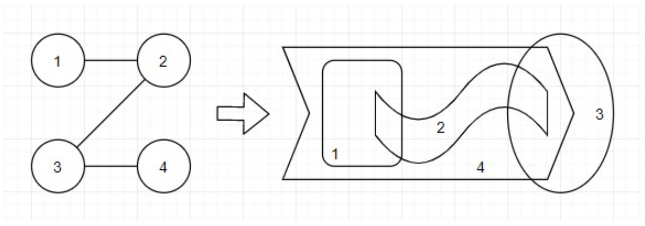
```
2
```

Note

In the first sample, you can obtain a nested sequence of 4 rubber bands(1, 2, 5, and 6) by the conversion shown below. Of course, there are other conversions exist to make a nested sequence of 4 rubber bands. However, you cannot make sequence of 5 or more nested rubber bands with given tree.



You can see one of the possible conversions for the second sample below.



E. JYPnation

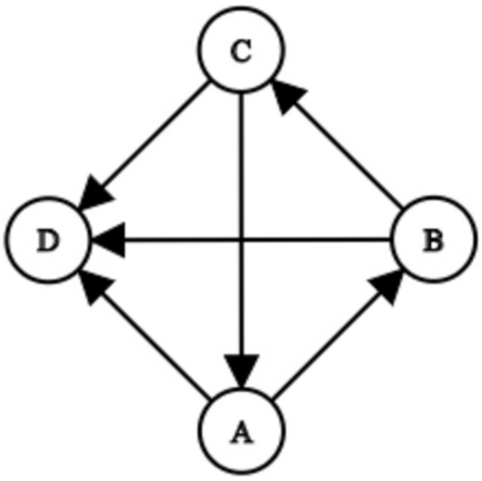
time limit per test: 2 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

Due to the success of TWICE, JYP Entertainment has earned countless money and emerged as the biggest entertainment firm by market capitalization. Therefore, the boss, JYP, has decided to create a new nation and has appointed you to provide a design diagram.

The new nation consists of n cities and some roads between them. JYP has given some restrictions:

- To guarantee efficiency while avoiding chaos, **for any 2 different cities A and B, there is exactly one road between them, and it is one-directional. There are no roads connecting a city to itself**.

- The logo of rivaling companies should not appear in the plan, that is, **there does not exist 4 distinct cities A,B,C,D , such that the following configuration occurs.**



JYP has given criteria for your diagram. For two cities A,B, let $dis(A,B)$ be the smallest number of roads you have to go through to get from A to B. If it is not possible to walk from A to B, $dis(A,B) = 614n$. Then, the efficiency value is defined to be the sum of $dis(A,B)$ for all ordered pairs of distinct cities (A,B).

Note that $dis(A,B)$ doesn't have to be equal to $dis(B,A)$.

You have drawn a design diagram that satisfies JYP's restrictions. Find the sum of $dis(A,B)$ over all ordered pairs of cities (A,B) with $A \neq B$.

Note that the input is given in compressed form. But even though it is compressed, you'd better use fast input.

Input

The first line contains a single integer n ($4 \le n \le 8000$, $n \equiv 0 \pmod 4$) — the number of cities.

A binary matrix is encrypted in the following format. Each of n next lines contains $\frac{n}{4}$ one-digit hexadecimal numbers (that is, these numbers can be represented either as digits from 0 to 9 or as uppercase Latin letters from A to F). Binary representation of each of these numbers denotes next 4 elements of the matrix in the corresponding row. For example, if the number B is given, then the corresponding elements are 1011, and if the number is 5, then the corresponding elements are 0101.

After you obtain the decrypted binary matrix, the j-th character of the i-th row is 1 if the one-directional road between cities i and j is directed from i to j, and 0 otherwise. It is guaranteed that the graph satisfies the restrictions mentioned above.

Output

Output one integer, representing the sum of $dis(A,B)$ over all ordered pairs of cities (A,B) with $A \neq B$.

Examples

input

```
4
7
2
1
4
```

output

```
7380
```

input

```
8
7F
3F
1F
0C
06
03
11
18
```

output

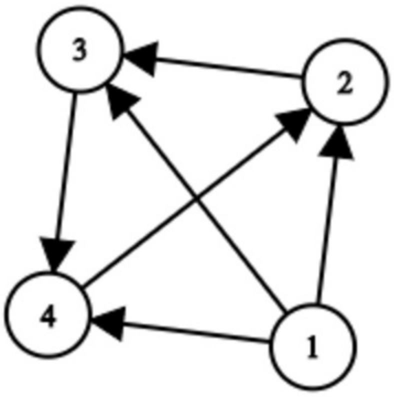
```
88464
```

Note

The first example corresponds to the matrix:

$$\begin{matrix}0111\\0010\\0001\\0100\end{matrix}$$

Which corresponds to this graph:



$dis(1,2) = dis(1,3) = dis(1,4) = dis(2,3) = dis(3,4) = dis(4,2) = 1$

$dis(2,4) = dis(4,3) = dis(3,2) = 2$

$dis(2,1) = dis(3,1) = dis(4,1) = 2456$

Therefore the answer for the diagram is 7380.