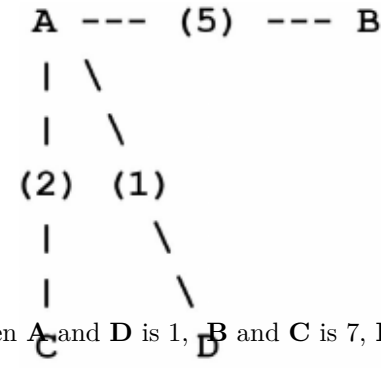
You are given an unrooted weighted tree. The weights on the edges are 16 bit unsigned integers, that is they are between 0 to $2^{16}-1$ (inclusive). For every integer x in the range 0 to $2^{16}-1$ (inclusive) find out how many pairs (unordered) of distinct nodes in the given tree have distance x. Distance between two nodes in the tree is defined as the bitwise xor of the edge weights on the path between these two nodes.



For example consider the tree on the right:

There are four nodes **A**, **B**, **C** and **D** in this tree. The edge weights are: **AB (5)**, **AC (2)** and **AD (1)**. So the distance between **A** and **D** is 1, **B** and **C** is 7, **B** and **D** is 4 etc.

Input

First line of the input contains a positive integer T ($T \leq 10$) denoting the number of test cases. Hence T cases follow. Each case starts with a positive integer n ($n \leq 100000$) denoting the number of nodes in the tree. Hence $n-1$ lines follow with the format 'u v w' meaning there is an edge between u and v ($1 \leq u, v \leq n$) with the weight w.

Output

For each test case output the case number (no trailing space after '`Case` x`:`') followed by the number of paths with the distance x for every x in the range 0 to $2^{16}-1$ (inclusive). There should **NOT** be empty line(s) between two cases. Please see the Sample Input output for the details.

Note for the Sample: Please note, the output below is truncated intentionally to save the trees, electricity, ram consumption, network bandwidth and so on. In fact, till $2^{16}-9$, '`0`'s follow.

Sample Input

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

Sample Output

```
Case 1:
0
1
1
1
1
1
0
1
0
```