

Problem B. Simple Tree Problem

Time limit: 20 seconds

Memory limit: 512 Megabytes

There is an undirected tree with n vertices and $n - 1$ edges.

The i -th vertex has a positive integer value of $a_i (i = 1, 2, \dots, n)$.

The i -th edge has a positive integer value of $k_i (i = 1, 2, \dots, n - 1)$.

We define $f(x, T)$ as the total number of vertices in tree T with value equal to x .

We define $g(y, T)$ as the maximum x that satisfies $f(x, T)$ is not less than y , if there is no x that satisfies the condition, then $g(y, T)$ is equal to 0.

For the i -th edge, **if** we remove it, the original tree will be divided into two new trees, denoted as T_{i_1} and T_{i_2} , respectively.

For the i -th edge, you need to calculate $\max(g(k_i, T_{i_1}), g(k_i, T_{i_2})) (i = 1, 2, \dots, n - 1)$.

Please note that for each edge, we will not really remove it.

Please pay attention to the time complexity of your program.

Input

Each test contains multiple test cases. The first line of input contains a single integer $t (1 \leq t \leq 10^6)$ — the number of test cases. The description of test cases follows.

The first line of each test case contains a single integer $n (1 \leq n \leq 10^6)$ — the number of vertices.

The second line of each test case contains n integers $a_i (1 \leq a_i \leq 10^9)$ — indicating the value of each vertex.

The following $n - 1$ lines of each test case contains three integers u_i, v_i and $k_i (1 \leq u_i, v_i, k_i \leq n, u_i \neq v_i)$ — indicating an edge with value k_i between vertices u_i and v_i . It is guaranteed that the given edges form a tree.

It is guaranteed that the sum of n does not exceed 3×10^6 .

Output

For each testcase, output $n - 1$ lines, where the i -th line contains an integer representing the answer to the i -th edge.

Notes: In this problem, you may need more stack space to pass this problem. We suggest you to add the following code into your main function if you use C++.

```
int main() {
    int size(512<<20); // 512M
    __asm__ ( "movq %0, %%rsp\n"::"r"((char*)malloc(size)+size));
    // YOUR CODE
    ...
    exit(0);
}
```

And if you use the code above please **DON'T forget to add `exit(0)`** in the end of your main function, otherwise your code may get RE.

Example

standard input	standard output
3	2
5	5
5 2 1 2 2	5
3 4 2	5
3 2 1	5
2 1 1	1
2 5 1	1
5	0
2 1 3 1 5	
2 4 1	
2 1 2	
1 3 2	
1 5 3	
1	
3	