G - Leaf Color Editorial by en_translator

There are several solutions for this problem; in this editorial, we introduce a technique so-called Auxiliary Tree.

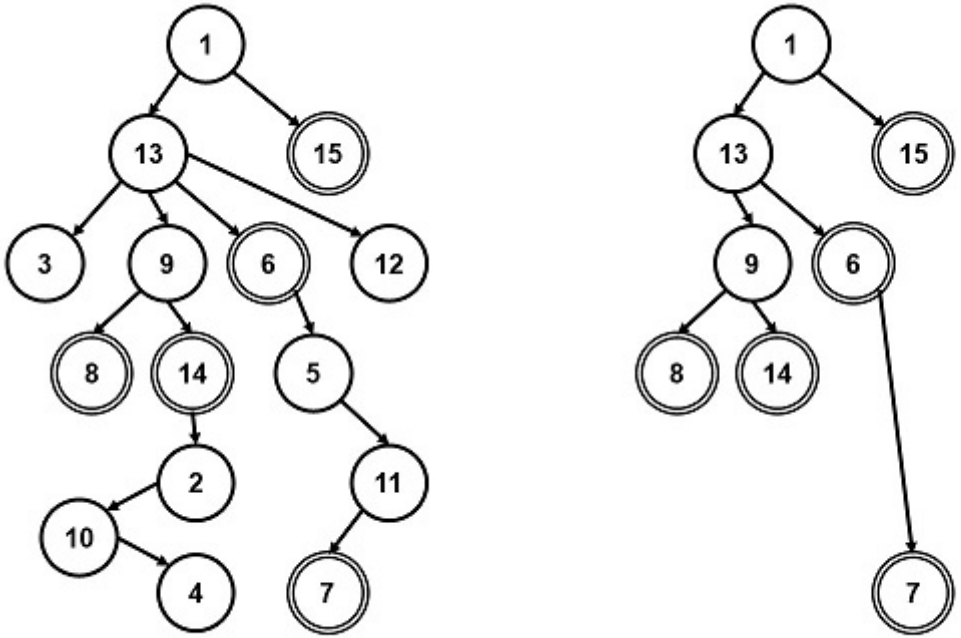
Approach of this editorial

We first explain the approach of solving this problem.

Fix a color c of degree-one vertex. Then the problem for the fixed color can be solved in $\mathrm{O}(N)$ time with tree DP (Dynamic Programming). (Although not easy, it is a good exercise for blue coders, so we recommend you to think about it.) However, solving it for all color costs $\mathrm{O}(N^2)$ time, leading to TLE (Time Limit Exceeded).

The problem for a fixed color c costs $\mathrm{O}(N)$ time because the tree has N vertices, which in fact includes essentially unnecessary vertices such as those that can never be used; caring such vertices when performing DP for color c is waste of time. Can we somehow compress the tree?

Consider which vertices we need to retain when compressing. Besides color-c vertices, it seems sufficient to retain vertices that are LCA (Lowest Common Ancestor) of color-c vertices. (Here, we regard tree T as a rooted tree rooted at vertex 1.) After such vertices are chosen, one adds an edge between each ancestor-descendant vertex pair, so that the tree on the left transforms to the right in the figure below. It seems that we can perform the DP on the tree on the right.



(Figure: compression when vertices $6, 7, 8, 14$, and 15 are painted with color c)

We call the tree resulting from the compression **an auxiliary tree obtained by compressing a tree so that LCA relations of specified vertices are preserved**. (This is informally called auxiliary tree, virtual tree, or compression tree (in Japan and China etc.). Dropping "obtained by…" part, we simply call it an **Auxiliary Tree** in this editorial.)

- Regarding the name of this tree, see also the following article. Auxiliary Tree に関する Kyopro Encyclopedia of Algorithms の記事 (Japanese article on Auxiliary Tree in "Kyopro (Competitive Programming) Encyclopedia of Algorithms")

Property of Auxiliary Tree

The formal definition of Auxiliary Tree is as follows. Given a rooted tree T and a vertex set X, the Auxiliary Tree is a rooted tree whose vertex set $A(X)$ is

$$A(X) = \{\mathrm{lca}(x, y) | x \in X, y \in X\}$$

and edges exist between every pair of vertices in which one is an ancestor of the other in T.

We introduce a property of Auxiliary Tree.

> Property 1
>
> $|A(X)| \leq 2|X| - 1.$

(Proof) Take any pre-order of a DFS (Depth-First Search) of the tree T. Let $x_1, x_2, \ldots, x_m$ be the vertices in X arranged in the pre-order.
Let $l = \mathrm{lca}(x_1, x_2)$; then we have the following fact:

> Lemma 1
>
> For an integer n greater than or equal to 3, if $\mathrm{lca}(x_1, x_n) \neq l$, then $\mathrm{lca}(x_1, x_n) = \mathrm{lca}(x_2, x_n)$.

(Proof of Lemma) We prove it by contradiction. Suppose there exists n such that $\mathrm{lca}(x_1, x_n) \neq l$ and $\mathrm{lca}(x_1, x_n) \neq \mathrm{lca}(x_2, x_n)$. If $\mathrm{lca}(x_1, x_n) = m$, then m is a descendant or ancestor of l, but if it is an ancestor, then $\mathrm{lca}(x_1, x_n) = \mathrm{lca}(x_2, x_n) = m$, which never happens. So m is a descendant of l, but then the pre-order should occur in this order: x_1, x_n, x_2. This violates the definition of x. Thus, this is a contradiction. (End of proof of lemma)

As lemma shows, $\mathrm{lca}(x_1, x_n)$ is one of: (1) x_1 itself, (2) $\mathrm{lca}(x_1, x_2)$, and (3) $\mathrm{lca}(x_2, x_n)$. Thus, $A(x)$ can be represented as

$$A(\{x_1, \dots, x_m\}) \\ = A(\{x_2, \dots, x_m\}) \cup \{x_1, \mathrm{lca}(x_1, x_2)\}.$$

This equation implies

$$|A(X)| \leq |A(\{x_2, \dots, x_m\})| + 2;$$

by induction on the size of vertex set, it is shown that $|A(X)| \leq 2|X| - 1$. (End of proof)

- By the way, we can also prove that the inequality is strict: when T is a perfect binary tree and X consists of its leaves, we can indeed achieve $|A(X)| = 2|X| - 1$.

This property implies that an Auxiliary Tree has $\mathrm{O}(|X|)$ vertices, which can be bounded by a constant times the size of the specified vertex set. We exploit this fact to reduce the computational complexity in the original problem.

Construction of Auxiliary Tree

Given a vertex set X, one can construct the Auxiliary Tree in $\mathrm{O}(|X|(\log |X| + \log N))$ time (if the original tree has N vertices. Also, a precomputation on the tree T of cost $\mathrm{O}(N \log N)$ is required).

We explain the procedure of construction. We use the following property:

> Property 2 Take any pre-order of a DFS of the tree T. Let $x_1, x_2, \dots, x_m$ be the vertices in X arranged in the pre-order.
> Then $A(X)$ has the following property:
>
> $$A(X) = X \cup \{\mathrm{lca}(x_i, x_{i+1}) | 1 \leq i < m\}.$$

(Proof)

Using the equation

$$A(\{x_1, \dots, x_m\}) \\ = A(\{x_2, \dots, x_m\}) \cup \{x_1, \mathrm{lca}(x_1, x_2)\}$$

shown in the proof of property 1, it turns out that

$$
\begin{aligned}
&A(\{x_1, \ldots, x_m\}) \\
&= A(\{x_2, \ldots, x_m\}) \cup \{x_1, \mathrm{lca}(x_1, x_2)\} \\
&= A(\{x_3, \ldots, x_m\}) \cup \{x_1, \mathrm{lca}(x_1, x_2), x_2, \mathrm{lca}(x_2, x_3)\} \\
&\vdots \\
&= A(\{x_m\}) \cup \{x_1, \mathrm{lca}(x_1, x_2), x_2, \ldots, x_{m-1}, \mathrm{lca}(x_{m-1}, x_m)\} \\
&= \{x_1, \mathrm{lca}(x_1, x_2), x_2, \ldots, x_{m-1}, \mathrm{lca}(x_{m-1}, x_m), x_m\},
\end{aligned}
$$

which is what we want. (End of proof)

Using property 2, one can construct a pre-order of the vertex set of $A(X)$ as follows:

- Precalculate doubling or HLD (Heavy-Light Decomposition) so that LCA of any two vertices on T can be obtained fast.
- Take a pre-order $x_1, x_2, \ldots, x_m$ of the vertices in X.
- Obtain $\mathrm{lca}(x_1, x_2)$, …, and $\mathrm{lca}(x_{m-1}, x_m)$.
- Sort $x_1, x_2, \ldots, x_m, \mathrm{lca}(x_1, x_2), \ldots, \mathrm{lca}(x_{m-1}, x_m)$ in the pre-order and remove duplicates.

Once we obtain the vertices that occur, all that left is to scan this sequence while maintaining a stack. (We do not explain in detail. For the actual procedure, see an article by yaketake08 (in Japanese), chapter “ソート2回の方法”. (Translator’s note: one can also refer to ICL1705 - Editorial - editorial - CodeChef Discuss, which is mentioned in the Japanese article too.)

Based on the technique introduced so far, one can generate the auxiliary tree from vertex set painted in each color and perform tree DP, so that the problem is solved in a total of $\mathrm{O}(N \log N)$, which is fast enough.