Editorial

badry

March 2018

1 Winter is Here

1.1 Prerequisites

- trees.
- Data structures.

1.2 Complexity : $O(NLog^2N + QLogN)$

1.3 Solution:

It is easy to realize that the answer is to find 2 deepest nodes in the subtree of V with indices between L and R but these 2 nodes have to be in 2 different children of V. For simplicity lets first try to find only the deepest node in the subtree of V with index between L and R and then see how we can extend this to solve our main problem. First, Lets convert the tree into array using the Euler tour trick.After doing this we will get all the subtree of V in a continuous segment lets assume it is $[Lv, Rv]$. Each cell in this array represent a node with 2 values (d, i) where d is its depth and i is its index. Our problem now reduced to the following : giving range $[x, y]$ in the array find the cell containing i between L and R and its d is the maximum. There are many ways to solve the above mentioned problem, one of them is to use segment tree where each node in it contains a small sparse table. the segment tree is to handle the $[x, y]$ and the sparse table is to handle the $[L, R]$ range. also in each node we have to compress the values of i in it so that their range became the same as the range this node covers.By doing so we can answer any query in $LogN$ time. Ok now we have the deepest node in the subtree of V what should we do to find the other one?. It is simple just identify the child of V that contains this node and assume its range in the euler array is $[Lc, Rc]$ we can exclude its range for the range of V and do exactly the same with the reaming part.In other words we can query the range $[Lv, Lc - 1]$ and the range $[L, Lc - 1]$. We can know this child using binary lifting