

**{121, 112}**

## Upper bound

Always use  $1 \rightarrow 1 \rightarrow 1 \rightarrow \dots$  for “larger” subtree and  $1 \rightarrow 2 \rightarrow 1 \rightarrow 2 \dots$  for “smaller” subtree, so that the alternating paths  $1 \rightarrow 2 \rightarrow 1 \rightarrow 2 \dots$  that you create can be as short as possible. E.g. if you have a long path with some short hairs, the long path would be  $1 \rightarrow 1 \rightarrow 1 \rightarrow \dots$  and only the hairs would need color 2 in a coordinated manner.

Hence  $O(\log n)$

## Lower bound

It is easy to see that in a deterministic setting there is no fast algorithm since, looking at a regular tree, the number of label-2 nodes on level  $x$  completely determines the number of label-2 nodes on levels  $x+1$ ,  $x+2$ , etc.  $\Rightarrow$  we have  $\Omega(\log n)$ .

Moreover, we can argue that randomness does not help here. Let's assume the opposite. Then there is a node  $v$  somewhere in the middle of a large rooted tree  $G$  s.t. the output of  $v$  depends on the random bits in its constant neighbourhood  $C$ . In particular, the output of  $v$  is not completely determined by unique ids of nodes in  $C$  and the tree structure. Then, if we look at the set  $X$  of nodes s.t. they are at the distance of  $d$  from  $v$  and all nodes in  $X$  are descendants of  $v$ . Given that  $d$  is large enough, the number of label-2 nodes in  $X$  has to change whenever  $v$  changes its label. Thus, no matter what's the distribution of labels in set  $X$ , if their random bits are independent of the random bits of  $v$ , with high probability we will have a mismatch somewhere in the tree.

Hence  $\Omega(\log n)$

Thus, we have  $\Theta(\log n)$