Balanced Tree

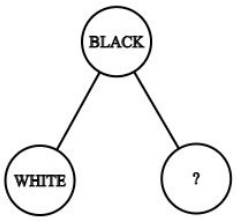
-editorial-

Author: Tatomir Alex & Oncescu Costin
Preparation: Tatomir Alex & Oncescu Costin

Task: Find a coloring for a given tree such that D is minimized. The tree has to be D-Balanced.

10 points: Try every possible coloring - $\Theta(2^N * N)$ or $\Theta(2^N * N^2)$ per test.
13 points: Dynamic programming on tree - $\Theta(N)$ per test.
26 points: Any variation of the official solution that works on paths/chains.
94 or 100 points: The following solution solves the problem in $\Theta(N \log N)$ or $\Theta(N)$.

First of all, let's decide if we can find a coloring such that the tree is D-Balanced. This can be achieved only if one of the following conditions holds:



- All nodes are colored only in black or only in white.
- There are at least 2 white nodes and at least 2 black nodes.

Otherwise, the answer for this test case is '-1'. Be careful to check the tricky case with 3 nodes of different colours (white, black, and undecided).

From now on, we will try to solve the following task : *"For a given tree, a given partial coloring and a positive integer D, check if you can make the tree D-Balanced."*
In order to solve this task we need to define some notions for a given rooted (sub)tree :

- We say that a node is satisfied if it has at least one other node of the same colour at distance at most D. Otherwise, it is unsatisfied.
- We say that a colour is satisfied if all nodes(possibly none) of that colour are satisfied. Otherwise, it is unsatisfied.
- We say that a node is a '*top node'* of its colour if it has the smallest depth compared to the other nodes of the same colour. Note that there may be more '*top nodes*' for each colour.
- We say that a node is a 'tail node' of its colour if it is unsatisfied and has the largest depth compared to the other unsatisfied nodes of that colour. Note that there may be more '*tail nodes*' for each colour.

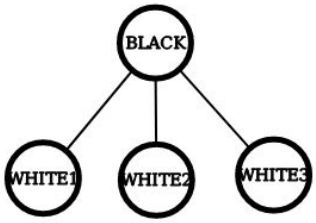
Now it is time to find the relevant information about a fully colored subtree :

- If both colours are satisfied, then we are interested only in the depth of the top nodes of black and white. Considering that the root is either black or white, we are interested only in the depth of the top node of the other colour.
- If both colours are unsatisfied, then we have to analyze only one special case(see case B).

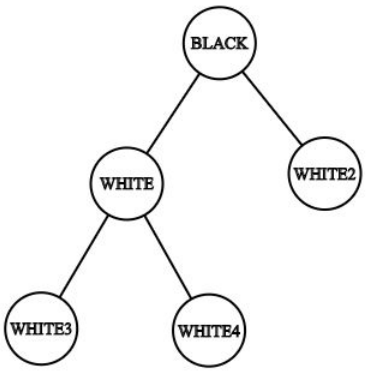
- If only one of the colours is satisfied, then we need to know the depth of the tail nodes of the unsatisfied colour and the colour of the actual root. In order to prove that the head nodes are irrelevant for the further computations, we need to consider the node T (it is outside the current subtree) that satisfies the tail nodes. T will be closer to the upcoming unsatisfied nodes (of the same colour) than any other node inside the subtree. Hence, no other information is required to describe this coloring.

Let's define the following groups of colorings for a subtree :

A. The coloring of a subtree with only one node. It is uniquely determined by the colour of root.
B. The coloring of a subtree which has both colours unsatisfied and has a chance to be fully satisfied in the future. (i.e. any unsatisfied node is at depth at most D)
   If D equals 1 then the subtree is similar to the one in the first picture(left).
   If D is greater than 1 then the subtree is similar to the one in the second picture(right).
   There is no other configuration with both colours unsatisfied. For a given subtree, there may be at most one valid coloring in this group.



C. The colorings of a subtree with both colours satisfied. As proved before, we are interested only in the depth of the top node of both colours. Consequently, we can say that a coloring is better (i.e. the best choice no matter the structure and colours of the nodes outside the subtree) than another if the depth of the head node of the colour different from root's colour is smaller in the first coloring compared to the second one. Hence, we can choose the best coloring according to this rule and ignore the others (from this group).
D. The colorings that have the colour of the root satisfied and the other unsatisfied. We are now interested only in the depth of the tail nodes (it was proven before that all the other information is not useful). As in case C, we can say that a coloring is better that another if the tail nodes have smaller depth. Hence, we can choose the best coloring according to this rule and ignore the others (form this group).
E. The coloring that has the colour of the root unsatisfied and the other satisfied. This is a tricky case because it can be easily confused with D. The difference is that the root is not a satisfied top node as before. The child nodes are now top nodes for the satisfied colour. The following picture may clarify the structure of this kind of coloring. Note that there must be no other node of the same colour as root at distance at most D.



For simplicity, we can integrate cases A and B in C, D, E.

Now we can define the following dynamic programming :
DP[root of subtree][group A/B/C/D/E] = the best coloring (according to the group) of the rooted subtree (only the relevant information). The recurrence is constant in time and is mainly based on case analysis. We say that a the tree can be made D-Balanced if there is any valid coloring in DP[1][C].

- *Solution for 94 points*

We can binary search on the D value and check if the tree can be made D-Balanced at each step.
Time complexity : $\Theta(N \ log \ N)$
Memory complexity : $\Theta(N)$

- *Solution for 100 points*

We are only a few steps apart from the solution. The trick here is to make the search interval for D as small as possible. This can be done in 2 steps.

1. Find a lower bound for the answer
2. Prove that the upper bound is c more than the lower bound (c is a constant that has to be determined)

1. Colour all undetermined nodes in white and compute the minimum D needed to satisfy all initial white nodes. Let this value be estimated_white.
   Colour all undetermined nodes in black and compute the minimum D needed to satisfy all initial black nodes. Let this value be estimated_black.
   The lower bound is maximum of estimated_white and estimated_black. These values can be computed with a standard dynamic programming technique.
2. It can be proven (intuitively or with case analysis) that there is always a way to colour the nodes such that the answer is at most the lower bound + 3. For example, a good way to colour a big zone of undetermined nodes is to choose colours alternatively. There are some limit cases when we must reach the constant of 3.

Using the computed lower and upper bound in the binary search, the time complexity reduces significantly to $\Theta(N)$. You can also use this approximation to get partial score in the 4th subtask.