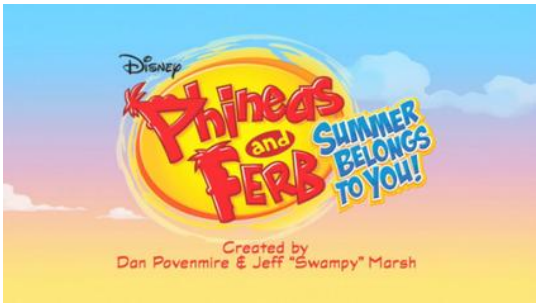
ACM SUMMER CHALLENGE-2K20

EDITORIAL

GRAPH AND TREE ALGORITHMS

Wandering at the end of acm summer challenge:

In this problem you have to find the lexicographically smallest permutation. Here you can traverse the node which is previously traversed. So you can solve it using set(for lexicographically smallest permutation). Start traversing from node 1 using BFS. And make one another array which will contain the recorded nodes on notebook.



Aomine's Last Game:

This question is an example of DFS and how to pass data during DFS recursion.

Key Idea: As described in the question, when a value of a node is modified, then it is certain that it inverts the condition of nodes in its subtree at even levels and doesn't invert the condition of nodes at odd levels. Also, no node condition inversion can affect the condition of its ancestor(s).

So, all you have to do is to maintain two variables while performing recursion that keeps count of modification done at even levels and odd levels respectively. Let's call these variables eVar and oVar respectively.

Now, at any particular node at level i, if i is even, then check whether the condition of node is equal to the required condition of node or not.

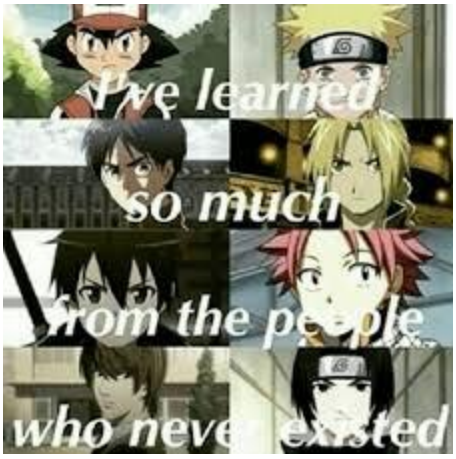
If it is, then check if eVar is even or not. If eVar is even,that means that the node doesn't get its condition modified. If it's odd, it gets its condition modified.

Now, if we don't want to modify the condition of the node (as we checked in the first case) and if eVar is odd (implying that it is modified due to modification of its ancestors), then we need to modify this node too. Otherwise, this node must not be modified.

Similar is the case for nodes at odd levels.

Any node that is modified must be stored in a data structure,preferably, a vector.

This vector and its size, then, contains the answer to our question.



Very Easy:

Consider tree having more than one node and rooted at level 1 .

Now color all nodes at odd level by "color1" and at even level by "color2" then no two adjacent node will have same color and only 2 color is used.

So run a dfs/bfs and count number of nodes at even and odd level, which is the required answer.

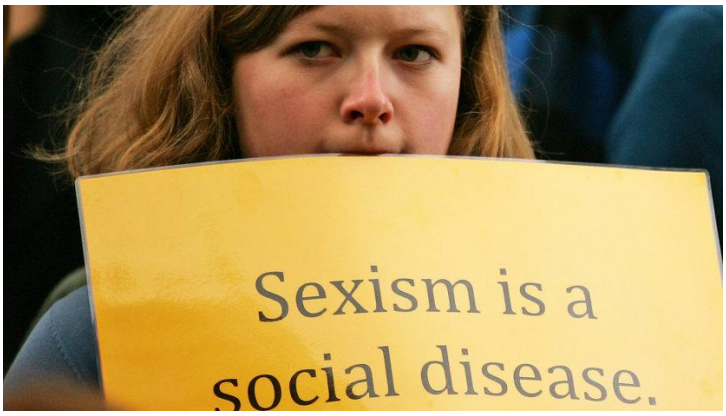
In case of N=1 only 1 color will be needed.

The Big News:

What do we have to find? To total how many aunties this news will spread if only a single aunty knows it initially.

So if we consider an aunty as an node then all the aunties belonging to same subgroup will have an edge pairwise. So with this knowledge our graph is built. After building this graph we have to find out how the spreading goes.

Let's see how the spreading occurs.. the initial aunty will tell all aunties with whom she shares a subgroup with and then all those aunties will tell the news in all the subgroups they belong to and so on until all of them are done spreading in all the subgroups they belong to.



The basic idea we can get from this is a node will spread the news to all such nodes that are connected with some edges(a path of edges exists between them). We have a perfect algorithm for such problems named as Union-Find Algorithm. There is one more way to do it with the help of DFS .