**1361. Validate Binary Tree Nodes: -**

Medium Accepted: 67.8K Submissions: 164.6K Acceptance Rate: 41.2%

You have n binary tree nodes numbered from 0 to n - 1 where node i has two children leftChild[i] and rightChild[i], return true if and only if **all** the given nodes form **exactly one** valid binary tree.

If node i has no left child then leftChild[i] will equal -1, similarly for the right child.

Note that the nodes have no values and that we only use the node numbers in this problem.

**Example 1:**

A diagram of a diagram

Description automatically generated

**Input:** n = 4, leftChild = [1,-1,3,-1], rightChild = [2,-1,-1,-1]

**Output:** true

**Example 2:**

A diagram of a triangle with arrows and circles

Description automatically generated

**Input:** n = 4, leftChild = [1,-1,3,-1], rightChild = [2,3,-1,-1]

**Output:** false

**Example 3:**

A black arrow pointing upwards

Description automatically generated

**Input:** n = 2, leftChild = [1,0], rightChild = [-1,-1]

**Output:** false

**Constraints:**

* n == leftChild.length == rightChild.length
* 1 <= n <= 104
* -1 <= leftChild[i], rightChild[i] <= n - 1

**Code: -**

class Solution {

public:

    bool helper(int n, int ind, vector<int>& left, vector<int>& right, unordered\_map<int,bool> &parent){

        if(ind >= n)

            return true;

        if(ind < n and parent[ind] == true)

            return false;

        parent[ind] = true;

        bool leftcheck = false, rightcheck = false;

        if(left[ind] != -1){

            leftcheck = helper(n, left[ind], left, right, parent);

            if(leftcheck == false)

                return false;

        }

        if(right[ind] != -1){

            rightcheck = helper(n, right[ind], left, right, parent);

            if(rightcheck == false)

                return false;

        }

        return true;

    }

    bool validateBinaryTreeNodes(int n, vector<int>& left, vector<int>& right) {

        vector<bool> par(n, false);

        for(int i = 0; i<n; ++i){

            if(left[i] != -1){

                if(par[left[i]] == true)    return false;

                par[left[i]] = true;

            }

            if(right[i] != -1){

                if(par[right[i]] == true)   return false;

                par[right[i]] = true;

            }

        }

        int head = -1;

        for(int i = 0; i<n; ++i){

            if(par[i] == false){

                if(head == -1)      head = i;

                else                return false;

            }

        }

        if(head == -1)      return false;

        cout << head;

        unordered\_map<int,bool> parent;

        bool result = helper(n, head, left, right, parent);

        if(parent.size() == n)

            return result;

        return false;

    }

};

**T.C: - O(N)**

**S.C: - O(N)**