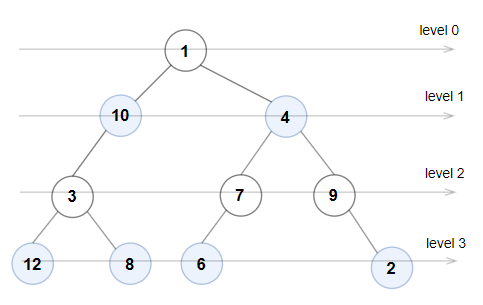
**[Even Odd Tree](https://leetcode.com/problems/even-odd-tree/)**

A binary tree is named **Even-Odd** if it meets the following conditions:

* The root of the binary tree is at level index 0, its children are at level index 1, their children are at level index 2, etc.
* For every **even-indexed** level, all nodes at the level have **odd** integer values in **strictly increasing** order (from left to right).
* For every **odd-indexed** level, all nodes at the level have **even** integer values in **strictly decreasing** order (from left to right).

Given the root of a binary tree, *return*true*if the binary tree is****Even-Odd****, otherwise return*false*.*

**Example 1:**



**Input:** root = [1,10,4,3,null,7,9,12,8,6,null,null,2]

**Output:** true

**Explanation:** The node values on each level are:

Level 0: [1]

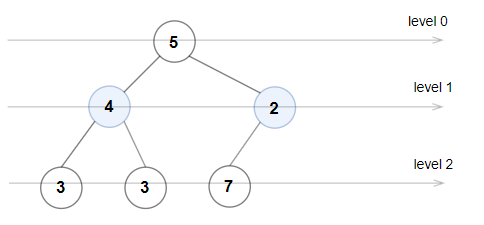
Level 1: [10,4]

Level 2: [3,7,9]

Level 3: [12,8,6,2]

Since levels 0 and 2 are all odd and increasing and levels 1 and 3 are all even and decreasing, the tree is Even-Odd.

**Example 2:**



**Input:** root = [5,4,2,3,3,7]

**Output:** false

**Explanation:** The node values on each level are:

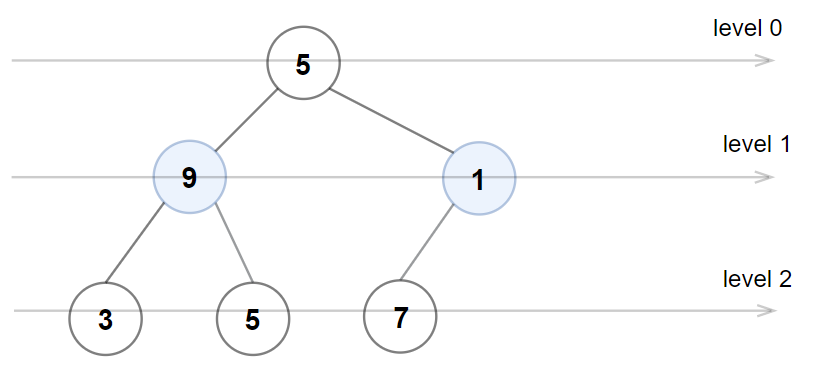
Level 0: [5]

Level 1: [4,2]

Level 2: [3,3,7]

Node values in level 2 must be in strictly increasing order, so the tree is not Even-Odd.

**Example 3:**



**Input:** root = [5,9,1,3,5,7]

**Output:** false

**Explanation:** Node values in the level 1 should be even integers.

**Constraints:**

* The number of nodes in the tree is in the range [1, 105].
* 1 <= Node.val <= 106

class Solution {

public:

    bool isEvenOddTree(TreeNode\* root) {

        if (!root) {

            return true;

        }

        std::queue<TreeNode\*> queue;

        int level = 0;

        queue.push(root);

        while (!queue.empty()) {

            int size = queue.size();

            int prev\_val = (level % 2 == 0) ? std::numeric\_limits<int>::min() : std::numeric\_limits<int>::max();

            for (int i = 0; i < size; ++i) {

                TreeNode\* node = queue.front();

                queue.pop();

                // Check if the values follow the conditions

                if ((level % 2 == 0 && (node->val % 2 == 0 || node->val <= prev\_val)) ||

                    (level % 2 == 1 && (node->val % 2 == 1 || node->val >= prev\_val))) {

                    return false;

                }

                prev\_val = node->val;

                // Add children to the queue

                if (node->left) {

                    queue.push(node->left);

                }

                if (node->right) {

                    queue.push(node->right);

                }

            }

            level++;

        }

        return true;

    }

};

Link : <https://leetcode.com/problems/even-odd-tree/?envType=daily-question&envId=2024-02-29>