2583. Kth Largest Sum in a Binary Tree

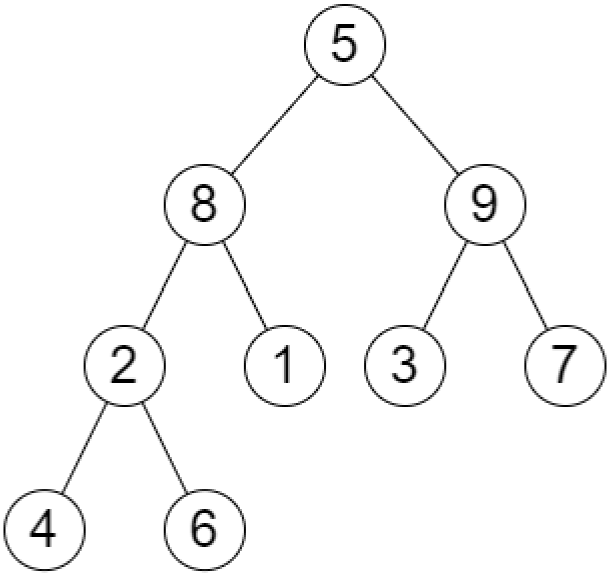
You are given the `root` of a binary tree and a positive integer `k`.

The **level sum** in the tree is the sum of the values of the nodes that are on the **same** level.

Return *the* `k`th ***largest*** *level sum in the tree (not necessarily distinct)*. If there are fewer than `k` levels in the tree, return `-1`.

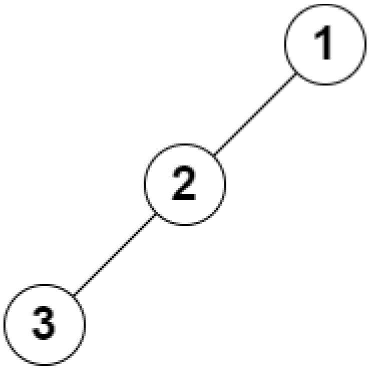
Note that two nodes are on the same level if they have the same distance from the root.

Example 1:



```
Input: root = [5,8,9,2,1,3,7,4,6], k = 2
Output: 13
Explanation: The level sums are the
following:
- Level 1: 5.
- Level 2: 8 + 9 = 17.
- Level 3: 2 + 1 + 3 + 7 = 13.
- Level 4: 4 + 6 = 10.
The 2nd largest level sum is 13.
```

Example 2:



```
Input: root = [1,2,null,3], k = 1
Output: 3
Explanation: The largest level sum is 3.
```

Constraints:

- The number of nodes in the tree is `n`.
- `2 <= n <= 105`
- `1 <= Node.val <= 106`
- `1 <= k <= n`

2583. Kth Largest Sum in a Binary Tree

```
/*
    Level traversel+array sorting
    Time complexity: $\Omega(n+h\log h), O(n+nlogn)$
    Space complexity: $O(n+h)$
    $n$ : total number of nodes in the tree
     $h$ : height of the binary tree= $\log_2(n)$
*/
class Solution{
public:
    long long kthLargestLevelSum(TreeNode* root, int k) {
        std::vector<long long> levels_sums;
        int level=0;

        auto level_traversal=[&](TreeNode* root)->void{
            if (!root) return;
            std::queue<TreeNode*> q;
            q.push(root);
            while(!q.empty()){
                int cur_subtree_size=q.size();
                long long s=0;
                while(cur_subtree_size>0){
                    TreeNode* cur_node=q.front();
                    q.pop();
                    cur_subtree_size--;
                    s+=cur_node->val;
                    if(cur_node->left) q.push(cur_node->left);
                    if(cur_node->right) q.push(cur_node->right);
                }
                levels_sums.push_back(s);
                level++;
            }
        };

        level_traversal(root);
        if(k>level) return -1;
        sort(levels_sums.begin(),levels_sums.end(),std::greater<long long>());
        return levels_sums[k-1];
    }
};
```

2583. Kth Largest Sum in a Binary Tree

```
/*
    Level traversel+Min heap
    Time complexity: $\Omega(n+h\log k), O(n+\log n\log k)$
    Space complexity: $O(n+k)$
  $n$ : total number of nodes in the tree
   $h$ : height of the binary tree= $\log_2(n)$
*/
class Solution{
public:
    long long kthLargestLevelSum(TreeNode* root, int k) {
        std::priority_queue<long long,std::vector<long long>,std::greater<long long>> min_heap;
        auto level_traversal=[&](TreeNode* root)->void{
            if (!root) return;
            std::queue<TreeNode*> q;
            q.push(root);
            while(!q.empty()){
                int cur_subtree_size=q.size();
                long long s=0;
                while(cur_subtree_size>0){
                    TreeNode* cur_node=q.front();
                    q.pop();
                    cur_subtree_size--;
                    s+=cur_node->val;
                    if(cur_node->left) q.push(cur_node->left);
                    if(cur_node->right) q.push(cur_node->right);
                }
                min_heap.push(s);
                if(min_heap.size()>k) min_heap.pop();
            }
        };
        level_traversal(root);
        if(k>min_heap.size()) return -1;
        return min_heap.top();
    }
};
```