Kth Largest Sum In A Binary Tree

solved by	Senan
Platform	LeetCode
difficulty	Medium
# Serial	2583
tags	BFS Heaps
language	C++
solved on	@22/10/2024
link	https://leetcode.com/problems/kth-largest-sum-in-a-binary-tree/description/
Completion	☑

Intuition

The problem involves calculating the sum of node values at each level of a binary tree and determining the k-th largest level sum. We can use a breadth-first search (BFS) approach to traverse the tree level by level. The goal is to keep track of the k largest level sums using a min-heap, as it allows efficient management of the k largest elements.

Approach

1. We traverse the tree level by level using BFS.
2. For each level, we compute the sum of the node values.
3. A min-heap is used to store the k largest sums. The heap keeps only k elements at any time, discarding smaller sums as necessary.
4. After finishing the BFS traversal, the top of the heap will hold the k-th largest level sum. If there are fewer than k levels, return -1.

Complexity

Time Complexity:

- **O(n log k)**where `n` is the number of nodes in the tree. This results from traversing each node once **O(n)** and maintaining a heap of size k, which takes **O(log k)** operations for insertion and deletion.

Space Complexity:

- **O(k)** for the priority queue storing the k largest sums.
- **O(n)** for the BFS queue, which in the worst case holds all the nodes of the tree.

Code

```
class Solution {
public:
    long long kthLargestLevelSum(TreeNode* root, int k) {
        priority_queue<long long, vector<long long>, greater<long long>> pq;
```

```
        queue<TreeNode*> q;

        q.push(root);

        while(!q.empty()){
            int size = q.size();
            long long sum = 0;
            for(int i = 0; i < size; i++){
                TreeNode* nd = q.front();
                q.pop();

                sum += (long long)(nd->val);
                if(nd->left) q.push(nd->left);
                if(nd->right) q.push(nd->right);
            }
            if(pq.size() < k) {
                pq.push(sum);
            } else if(pq.top() < sum) {
                pq.pop();
                pq.push(sum);
            }
        }
        return (pq.size() == k) ? pq.top() : -1;
    }
};
```