题目地址（129. 求根到叶子节点数字之和）

https://leetcode-cn.com/problems/sum-root-to-leaf-numbers/

入选理由

1. 路径是一种经典的树的题目，掌握路径是树必须的技能之一

标签

- 树
- BFS
- DFS

难度

- 中等

题目描述

给定一个二叉树，它的每个结点都存放一个 0-9 的数字，每条从根到叶子节点的路径都代表一个数字。

例如，从根到叶子节点路径 1->2->3 代表数字 123。

计算从根到叶子节点生成的所有数字之和。

说明: 叶子节点是指没有子节点的节点。

示例 1:

```
输入: [1,2,3]
    1
   / \
  2   3
输出: 25
解释:
从根到叶子节点路径 1->2 代表数字 12.
从根到叶子节点路径 1->3 代表数字 13.
因此，数字总和 = 12 + 13 = 25.
示例 2:
```

```
输入: [4,9,0,5,1]
    4
   / \
  9   0
 / \
5   1
输出: 1026
解释:
从根到叶子节点路径 4->9->5 代表数字 495.
从根到叶子节点路径 4->9->1 代表数字 491.
从根到叶子节点路径 4->0 代表数字 40.
因此，数字总和 = 495 + 491 + 40 = 1026.
```

前置知识

- DFS
- BFS
- 前序遍历

DFS

思路

求从根到叶子的路径之和，那我们只需要把每条根到叶子的路径找出来，并求和即可，这里用 DFS 去解，DFS 也是最容易想到的。

代码

代码支持：JS，Java，C++，Python

C++ Code:

```cpp
class Solution {
public:
    int sum = 0;
    int sumNumbers(TreeNode* root) {
        dfs(root, 0);
        return sum;
    }

    void dfs(TreeNode* root, int num) {
        if (!root) return;
        if (!root->left && !root->right) {
            sum += num * 10 + root->val;
            return;
```

```
        }
        dfs(root->left, num * 10 + root->val);
        dfs(root->right, num * 10 + root->val);
    }
};
```

Java Code:

```
class Solution {
    public int ans;

    public int sumNumbers(TreeNode root) {
        dfs(root, 0);
        return ans;
    }

    public void dfs(TreeNode root, int last){
        if(root == null) return;
        if(root.left == null && root.right == null) {
            ans += last * 10 + root.val;
            return;
        }
        dfs(root.left, last * 10 + root.val);
        dfs(root.right, last * 10 + root.val);
    }
}
```

Python Code:

```
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, x):
#         self.val = x
#         self.left = None
#         self.right = None

class Solution:
    def sumNumbers(self, root: TreeNode) -> int:
        def dfs(root, cur):
            if not root: return 0
            if not root.left and not root.right: return cur * 10 + root.val
            return dfs(root.left, cur * 10 + root.val) + dfs(root.right, cur * 10 + root.val)
        return dfs(root, 0)
```

JS Code:

```
function sumNumbers1(root) {
  let sum = 0;
  function dfs(root, cur) {
    if (!root) {
      return;
    }
    let curSum = cur * 10 + root.val;
    if (!root.left && !root.right) {
      sum += curSum;
      return;
    }
    dfs(root.left, curSum);
    dfs(root.right, curSum);
  }
  dfs(root, 0);
  return sum;
}
```

复杂度分析

令 n 为节点总数， h 为树的高度。

- 时间复杂度：$O(n)$
- 空间复杂度：$O(h)$

BFS

思路

如果说 DFS 是孤军独入，取敌将首级，那么 BFS 就是堂堂正正，车马摆开，层层推进。BFS 可能没那么优雅，但是掌握模板之后简直就是神器。

要求根到的叶子的路径的和，那我们把中间每一层对应的值都求出来，当前层的节点是叶子节点，把对应值相加即可。

代码

代码支持：JS，Java,Python3,CPP

JS Code：

```
function sumNumbers(root) {
  let sum = 0;
  let curLevel = [];
  if (root) {
```

```
    curLevel.push(root);
  }
  while (curLevel.length) {
    let nextLevel = [];
    for (let i = 0; i < curLevel.length; i++) {
      let cur = curLevel[i];
      if (cur.left) {
        cur.left.val = cur.val * 10 + cur.left.val;
        nextLevel.push(cur.left);
      }
      if (cur.right) {
        cur.right.val = cur.val * 10 + cur.right.val;
        nextLevel.push(cur.right);
      }
      if (!cur.left && !cur.right) {
        sum += cur.val;
      }
      curLevel = nextLevel;
    }
  }
  return sum;
}
```

Java Code：

```
class Solution {
    public int sumNumbers(TreeNode root) {
        Queue<TreeNode> queue = new LinkedList<>();
        queue.offer(root);
        int sum = 0;

        while (!queue.isEmpty()) {
            int size = queue.size();
            for (int i = 0; i < size; i++) {
                TreeNode cur = queue.poll();

                if (cur.left == null && cur.right == null) {
                    sum = sum + cur.val;
                }

                if (cur.left != null) {
                    cur.left.val = cur.val * 10 + cur.left.val;
                    queue.offer(cur.left);
                }

                if (cur.right != null) {
                    cur.right.val = cur.val * 10 + cur.right.val;
                    queue.offer(cur.right);
                }
            }
        }
```

```
        return sum;
    }
}
```

Python3 Code:

```
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def sumNumbers(self, root: TreeNode) -> int:
        res = 0
        q = deque()
        q.append((root,0))
        while q:
            node,value = q.popleft()
            if node.left:
                q.append((node.left,value*10+node.val))
            if node.right:
                q.append((node.right,value*10+node.val))
            if not node.left and not node.right:
                res += value*10+node.val
        return res
```

CPP Code:

```
class Solution {
public:
    int sumNumbers(TreeNode* root) {
        if (!root) return 0;
        queue<TreeNode*> qt;
        qt.push(root);

        int sum = 0;

        while (!qt.empty()) {
            int size = qt.size();
            while (size--) {
                TreeNode* node = qt.front();
                qt.pop();
```

```
                if (node->left) {
                    node->left->val += 10 * node->val;
                    qt.push(node->left);
                }

                if (node->right) {
                    node->right->val += 10 * node->val;
                    qt.push(node->right);
                }

                if (!node->left && !node->right) { // leaf node
                    sum += node->val;
                }
            }
        }

        return sum;
    }
};
```

复杂度分析

令 n 为节点总数， q 为队列长度。

- 时间复杂度：$O(n)$
- 空间复杂度：$O(q)$。最坏的情况是满二叉树，此时和 n 同阶。

为什么空间复杂度和 n 同阶呢？这是因为叶子节点的数目在极端情况下是完全二叉树，此时的叶子节点的数目差不多和 $n/2$ 相等。具体推导过程如下。

令 h 为树高度。

$k = h - 1$

$n = sum_{i=0}^{k} 2^i$ ①

$n/2 = sum_{i=-1}^{k-1} 2^i$ ②

$n/2 = 2^k - \frac{1}{2}$ ① - ②

$T(n) = 2^k = n/2 + \frac{1}{2}$ = O(n)

上一页 下一页