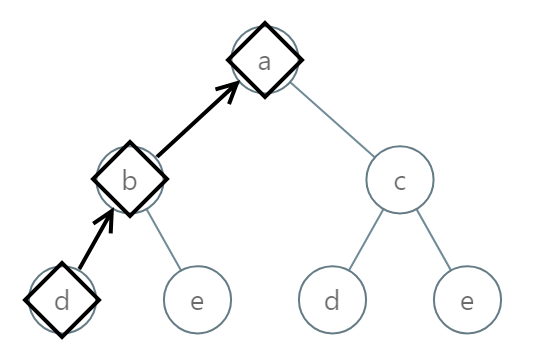
# 题目

给定一颗根结点为 root 的二叉树，树中的每一个结点都有一个从 0到 25 的值，分别代表字母 'a' 到 'z'：值 0代表 'a'，值 1 代表 'b'，依此类推。

找出按字典序最小的字符串，该字符串从这棵树的一个叶结点开始，到根结点结束。

（小贴士：字符串中任何较短的前缀在字典序上都是较小的：例如，在字典序上"ab"比"aba"要小。叶结点是指没有子结点的结点。）

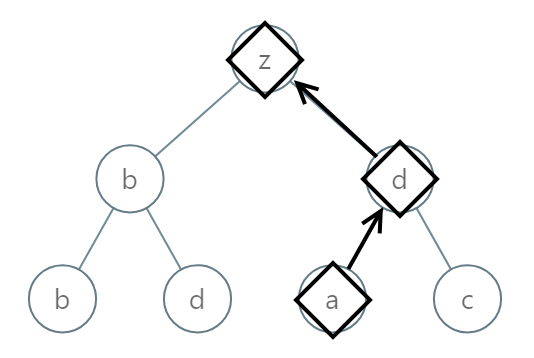
示例 1：



输入：[0,1,2,3,4,3,4]

输出："dba"

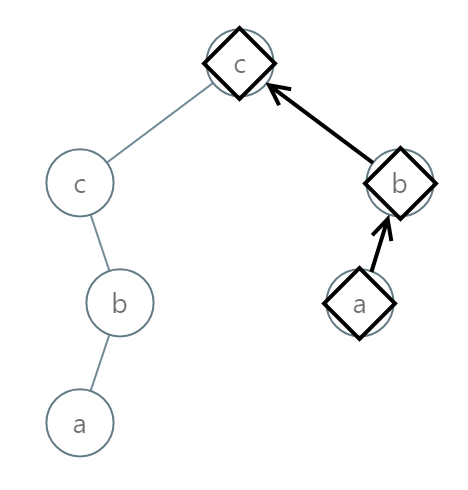
示例 2：



输入：[25,1,3,1,3,0,2]

输出："adz"

示例 3：



输入：[2,2,1,null,1,0,null,0]

输出："abc"

提示：

给定树的结点数介于 1 和 8500 之间。

树中的每个结点都有一个介于 0 和 25 之间的值。

# 分析

/\*\*

 \* Definition for a binary tree node.

 \* struct TreeNode {

 \*     int val;

 \*     TreeNode \*left;

 \*     TreeNode \*right;

 \*     TreeNode() : val(0), left(nullptr), right(nullptr) {}

 \*     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}

 \*     TreeNode(int x, TreeNode \*left, TreeNode \*right) : val(x), left(left), right(right) {}

 \* };

 \*/

class Solution {

public:

    vector<string> path;

    string smallestFromLeaf(TreeNode \*root)

    {

        dfs(root, "");

        sort(path.begin(), path.end()); //升序排序

        return path[0];

    }

    void dfs(TreeNode \*root, string s)

    {

        if (!root)

            return;

        s += 'a' + root->val;

        if (!root->left && !root->right)

        {

            reverse(s.begin(), s.end()); //题目要求从根节点到叶节点，因此反转

            path.push\_back(s);

            return;

        }

        dfs(root->left, s);

        dfs(root->right, s);

    }

};