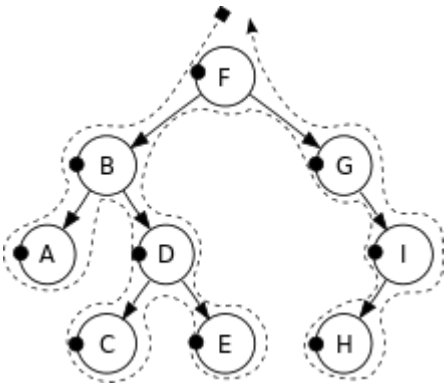
144. 二叉树的前序遍历

题目：144. Binary Tree Preorder Traversal 语言: python3 英文版链接：https://leetcode.com/problems/binary-tree-preorder-traversal/description/ 中文版链接：https://leetcode-cn.com/problems/binary-tree-preorder-traversal/description/

题目分析

前序遍历，指先访问根，然后访问子树的遍历方式

根结点 ---> 左子树 ---> 右子树



遍历结果：F, B, A, D, C, E, G, I, H

层次遍历使用 BFS 实现，利用的就是 BFS 一层一层遍历的特性；而前序、中序、后序遍历利用了 DFS 实现。

前序、中序、后序遍只是在对节点访问的顺序有一点不同，其它都相同。

① 前序

```
void dfs(TreeNode root) {
    visit(root);
    dfs(root.left);
    dfs(root.right);
}
```

② 中序

```
void dfs(TreeNode root) {
    dfs(root.left);
    visit(root);
    dfs(root.right);
}
```

③ 后序

```
void dfs(TreeNode root) {
    dfs(root.left);
    dfs(root.right);
    visit(root);
}
```

答案

递归版本

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

class Solution:
    def preorderTraversal(self, root: TreeNode) -> List[int]:
        result = []
        if not root:
            return result
        result.append(root.val)
        result += self.preorderTraversal(root.left)
        result += self.preorderTraversal(root.right)
        return result
```

非递归版本

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

class Solution:
    def preorderTraversal(self, root: TreeNode) -> List[int]:
        result, s = [], []
        if not root:
            return result
        s.append(root)
        while s:
            root = s.pop()
            result.append(root.val)
            if root.right:
                s.append(root.right)
            if root.left:
                s.append(root.left)
        return result
```