比较好的总结以及完成题目: <https://lfool.github.io/LFool-Notes/algorithm/%E4%BA%8C%E5%8F%89%E6%A0%91--%E7%BA%B2%E9%A2%86%E7%AF%87.html>

说后序位置之前，先简单说下中序和前序: 「中序位置」主要用在 BST 场景中，你完全可以把 BST 的中序遍历认为是遍历有序数组

「前序位置」本身其实没有什么特别的性质，之所以发现好像很多题都是在「前序位置」写代码，实际上是因为我们习惯把那些对前中后序位置不敏感的代码写在「前序位置」罢了

可以发现，「前序遍历」的代码执行是「自顶向下」的，而「后序遍历」的代码执行是「自底向上」的。这不奇怪，因为本文开头就说了前序位置是刚刚进入节点的时刻，后序位置是即将离开节点的时刻。但这里面大有玄妙，意味着「前序位置」的代码只能从函数参数中获取父节点传递来的数据；而「后序位置」的代码不仅可以获取参数数据，还可以获取到子树通过函数返回值传递回来的数据

无论是前、中、后序遍历，**都是先访问根节点，再访问它的左子树，再访问它的右子树。**

那它们之间的区别在哪里？比如中序遍历，它是将 do something with root（处理当前节点）放在了访问完它的左子树之后。

图示

描述已自动生成

前、中、后序遍历都是基于DFS，节点的访问顺序如上图所示，每个节点有三个不同的驻留阶段，即每个节点会被经过三次：

1. 在递归它的左子树之前。

2. 在递归完它的左子树之后，在递归它的右子树之前。

3. 在递归完它的右子树之后。

我们将 do something with root 这个操作，放在这三个时间点之一，就分别对应：前、中、后序遍历。所以，它们的唯一区别是：在什么时间点去处理节点，去拿他做文章。

Preorder, Postorder and Inorder are all based on DFS. The only difference is: During the traversal, what time they will access the content of a node. Because a node is actually visited 3 times for binary tree. They include: the time before the first DFS call, and the times after each DFS call.

* The preorder traversal accesses the content of a node when it is first visited (before the DFS on his left child). It is implemented as follows:

文本

描述已自动生成

* The postorder traversal accesses the content of a node when it is last visited (after the DFS on both children). It is implemented as follows :

图形用户界面, 文本

描述已自动生成

* The inorder traversal accesses the content of a node just before we visit right child. It is implemented as follows

图形用户界面, 应用程序

描述已自动生成