二叉树简单实现：

```
class Node:
    def __init__(self,item):
        self.item = item
        self.child1 = None
        self.child2 = None


class Tree:
    def __init__(self):
        self.root = None

    def add(self, item):
        node = Node(item)
        if self.root is None:
            self.root = node
        else:
            q = [self.root]

            while True:
                pop_node = q.pop(0)
                if pop_node.child1 is None:
                    pop_node.child1 = node
                    return
                elif pop_node.child2 is None:
                    pop_node.child2 = node
                    return
                else:
                    q.append(pop_node.child1)
                    q.append(pop_node.child2)

    def traverse(self):  # 层次遍历
        if self.root is None:
            return None
        q = [self.root]
        res = [self.root.item]
        while q != []:
            pop_node = q.pop(0)
            if pop_node.child1 is not None:
                q.append(pop_node.child1)
                res.append(pop_node.child1.item)

            if pop_node.child2 is not None:
                q.append(pop_node.child2)
                res.append(pop_node.child2.item)
        return res

    def preorder(self,root):  # 先序遍历
        if root is None:
            return []
        result = [root.item]
        left_item = self.preorder(root.child1)
        right_item = self.preorder(root.child2)
        return result + left_item + right_item
    def inorder(self,root):  # 中序序遍历
        if root is None:
            return []
        result = [root.item]
        left_item = self.inorder(root.child1)
        right_item = self.inorder(root.child2)
        return left_item + result + right_item

    def postorder(self,root):  # 后序遍历
        if root is None:
            return []
        result = [root.item]
        left_item = self.postorder(root.child1)
        right_item = self.postorder(root.child2)
        return left_item + right_item + result

t = Tree()
for i in range(10):
    t.add(i)
print('层序遍历:',t.traverse())
print('先序遍历:',t.preorder(t.root))
print('中序遍历:',t.inorder(t.root))
print('后序遍历:',t.postorder(t.root))
```

输出结果：

```
层次遍历: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
先次遍历: [0, 1, 3, 7, 8, 4, 9, 2, 5, 6]
中次遍历: [7, 3, 8, 1, 9, 4, 0, 5, 2, 6]
后次遍历: [7, 8, 3, 9, 4, 1, 5, 6, 2, 0]
```