二叉树的遍历是指按某条搜索路径访问二叉树中的每个节点一次且只有一次。

按照根、左子树和右子树的访问先后顺序不同，二叉树的遍历可以有6种方案：DLR、LDR、LRD、DRL、RDL、RLD。如果限定先左后右（先左子树后右子树），则只有前3种遍历方案：DLR、LDR、LRD。按照根的访问顺序不同，根在前面称为先序遍历（DLR），根在中间称为中序遍历（LDR），根在最后称为后序遍历（LRD）。

一、先序遍历

先序遍历是指先访问根，然后先序遍历左子树，再先序遍历右子树。

算法步骤：

如果二叉树为空，则空操作，否则：

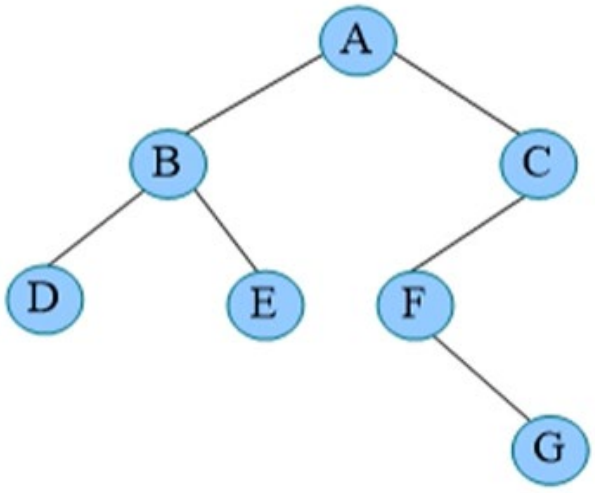
1）访问根节点；

2）先序遍历左子树；

3）先序遍历右子树。

先序遍历秘籍：访问根，先序遍历左子树，左子树为空或已遍历才可以遍历右子树。

对下面二叉树进行先序遍历：




```
void preorder(Btree T){//先序遍历
    if(T){
        cout<<T->data<<"  ";
        preorder(T->lchild);
        preorder(T->rchild);
    }
}
```

二、中序遍历

中序遍历是指中序遍历左子树，然后访问根，再中序遍历右子树。

算法步骤：

如果二叉树为空，则空操作，否则：

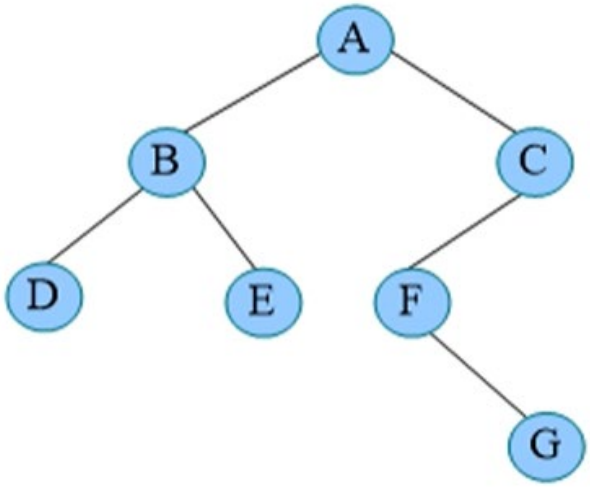
1）中序遍历左子树；

2）访问根节点；

3）中序遍历右子树。

中序遍历秘籍：中序遍历左子树，左子树为空或已遍历才可以访问根，中序遍历右子树。

对下面二叉树进行中序遍历：




```
void inorder(Btree T){//中序遍历
    if(T){
        inorder(T->lchild);
        cout<<T->data<<"  ";
        inorder(T->rchild);
    }
}
```

三、后序遍历

后序遍历是指后序遍历左子树，后序遍历右子树，然后访问根。

算法步骤：

如果二叉树为空，则空操作，否则：

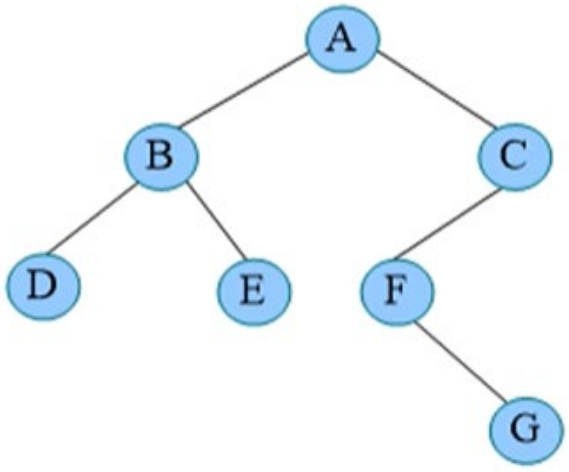
1）后序遍历左子树；

2）后序遍历右子树；

3）访问根节点。

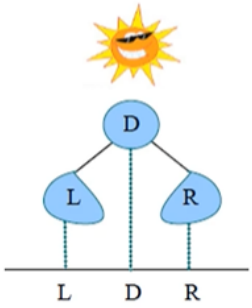
后序遍历秘籍：后序遍历左子树，后序遍历右子树，左子树、右子树为空或已遍历才可以访问根。

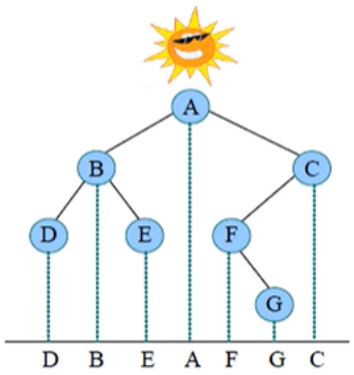
对下面二叉树进行后序遍历：



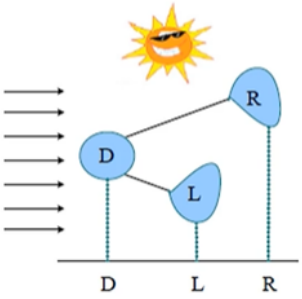
```
void posorder(Btree T){//后序遍历
    if(T){
        posorder(T->lchild);
        posorder(T->rchild);
        cout<<T->data<<"  ";
    }
}
```

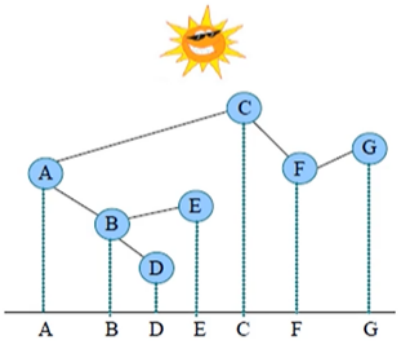
中序遍历就像在无风的情况下，遍历顺序为左子树、根、右子树，太阳直射，将所有的节点投影到地上。





先序遍历就像在左边大风的情况下，将二叉树的树枝刮向右方，且顺序为根、左子树、右子树，太阳直射，将所有的节点投影到地上。





后序遍历就像在右边大风的情况下，将二叉树的树枝刮向左方，且顺序为左子树、右子树、根，太阳直射，将所有的节点投影到地上。