

本节内容

二叉树  
先/中/后序  
遍历

王道考研/CSKAOYAN.COM

1

知识总览

二叉树的遍历

先序遍历

中序遍历

后序遍历

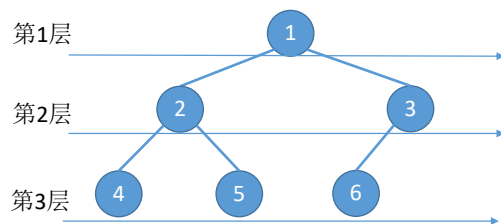
遍历算法的应用举例

王道考研/CSKAOYAN.COM

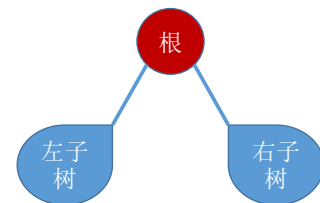
2

## 什么是遍历

遍历：按照某种次序把所有结点都访问一遍



层次遍历：基于树的层次特性确定的次序规则



先/中/后序遍历：基于树的递归特性确定的次序规则

王道考研/CSKAOYAN.COM

3

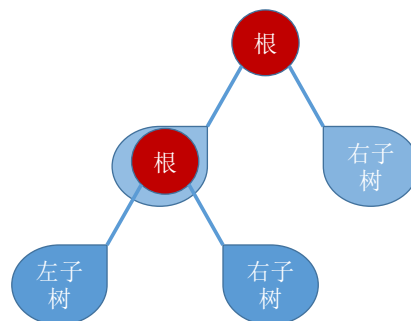
## 二叉树的遍历

二叉树的递归特性：

- ① 要么是个空二叉树
- ② 要么就是由“根节点+左子树+右子树”组成的二叉树



空二叉树



非空二叉树

先序遍历：根左右（NLR）

中序遍历：左根右（LNR）

后序遍历：左右根（LRN）

王道考研/CSKAOYAN.COM

4

### 二叉树的遍历

先序遍历: **A**BC  
中序遍历: B**A**C  
后序遍历: B**C**A

先序遍历: **A**B  
中序遍历: B**A**  
后序遍历: B**A**

右子树为空树

先序遍历: **A**C  
中序遍历: **A**C  
后序遍历: **C**A

左子树为空树

先序遍历: **A** B D E C F G  
中序遍历: D B E **A** F C G  
后序遍历: D E B F G C **A**

先序遍历: 根左右 (NLR)  
中序遍历: 左根右 (LNR)  
后序遍历: 左右根 (LRN)

王道考研/CSKAOYAN.COM

5

### 二叉树的遍历 (手算练习)

先序遍历: 根 左 右  
根 (根 左 右) (根 左)  
根 (根 (根 右) 右) (根 左)

中序遍历: ( 左 根 右) 根 (左 根)  
((根右) 根 右) 根 (左 根)

后序遍历: ( 左 右 根) (左 根) 根  
((右 根) 右 根) (左 根) 根

分支结点逐层展开法...

A B C  
A B D E C F  
A B D G E C F

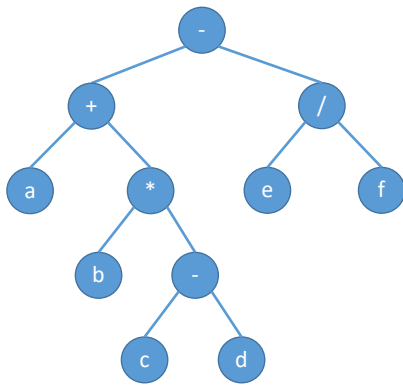
B A C  
D B E A F C  
D G B E A F C

B C A  
D E B F C A  
G D E B F C A

王道考研/CSKAOYAN.COM

6

## 二叉树的遍历（手算练习）



先序遍历:  $-+a*b-cd/ef$

中序遍历:  $a+b*c-d-e/f$

后序遍历:  $abcd-*+ef/-$

先序遍历  $\rightarrow$  前缀表达式

中序遍历  $\rightarrow$  中缀表达式（需要加界限符）

后序遍历  $\rightarrow$  后缀表达式

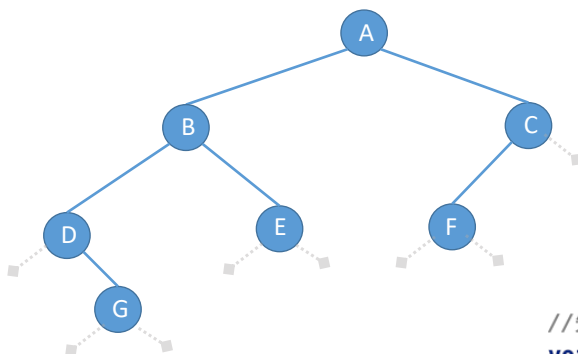
算数表达式的“分析树”

$a + b * (c - d) - e / f$

王道考研/CSKAOYAN.COM

7

## 先序遍历（代码）



先序遍历（PreOrder）的操作过程如下：

1. 若二叉树为空，则什么也不做；

2. 若二叉树非空：

①访问根结点；

②先序遍历左子树；

③先序遍历右子树。

```
typedef struct BiTNode{
    ElemType data;
    struct BiTNode *lchild,*rchild;
}BiTNode,*BiTree;
```

//先序遍历

```
void PreOrder(BiTree T){
```

```
    if(T!=NULL){
```

```
        visit(T);
```

//访问根结点

```
        PreOrder(T->lchild);
```

//递归遍历左子树

```
        PreOrder(T->rchild);
```

//递归遍历右子树

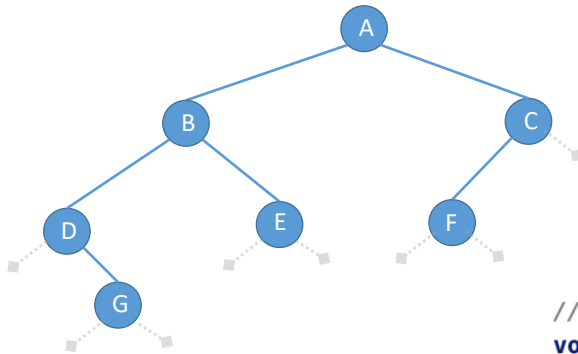
```
    }
```

```
}
```

王道考研/CSKAOYAN.COM

8

### 中序遍历（代码）



中序遍历（InOrder）的操作过程如下：

1. 若二叉树为空，则什么也不做；

2. 若二叉树非空：

①先序遍历左子树；

②访问根结点；

③先序遍历右子树。

//中序遍历

```
void InOrder(BiTree T){
```

```
    if(T!=NULL){
```

```
        InOrder(T->lchild);    //递归遍历左子树
```

```
        visit(T);              //访问根结点
```

```
        InOrder(T->rchild);    //递归遍历右子树
```

```
    }
```

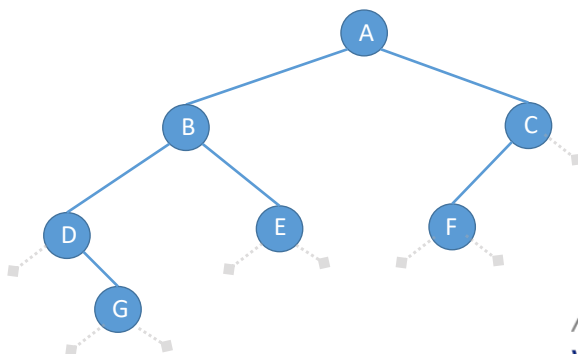
```
}
```

```
typedef struct BiTNode{
    ElemType data;
    struct BiTNode *lchild,*rchild;
}BiTNode,*BiTree;
```

王道考研/CSKAOYAN.COM

9

### 后序遍历（代码）



后序遍历（InOrder）的操作过程如下：

1. 若二叉树为空，则什么也不做；

2. 若二叉树非空：

①先序遍历左子树；

②先序遍历右子树；

③访问根结点。

//后序遍历

```
void PostOrder(BiTree T){
```

```
    if(T!=NULL){
```

```
        PostOrder(T->lchild);    //递归遍历左子树
```

```
        PostOrder(T->rchild);    //递归遍历右子树
```

```
        visit(T);                //访问根结点
```

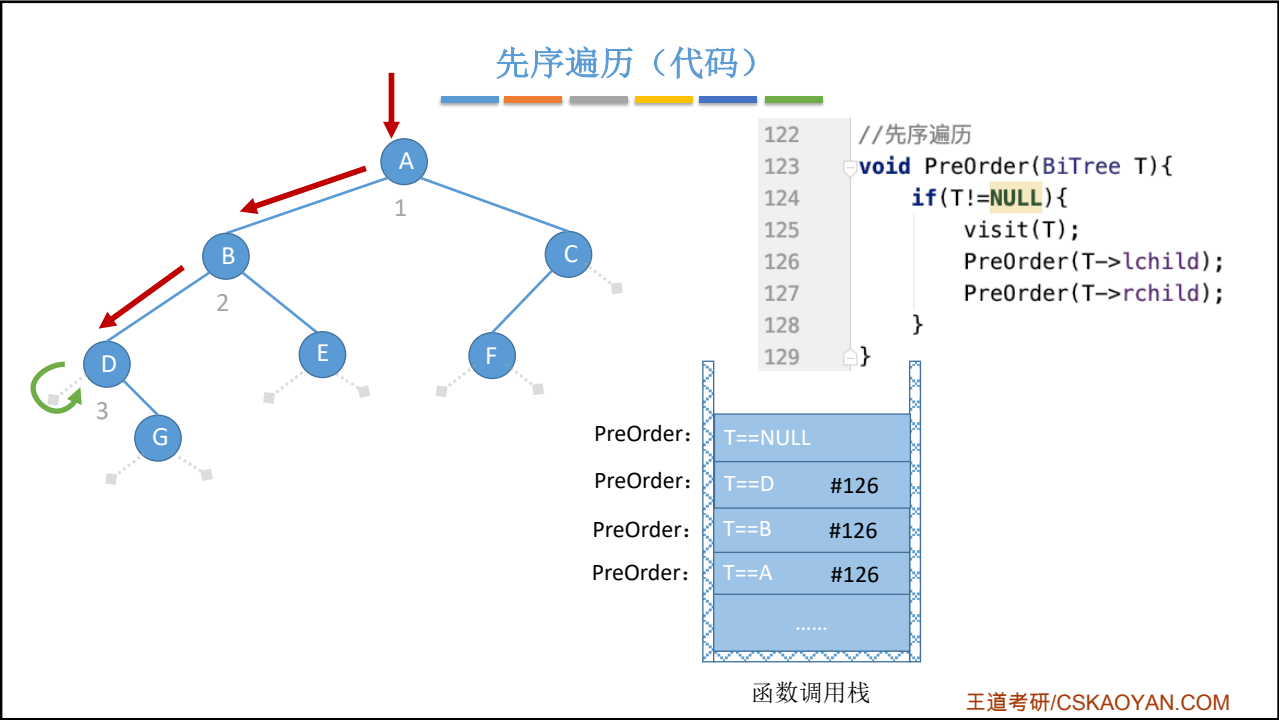
```
    }
```

```
}
```

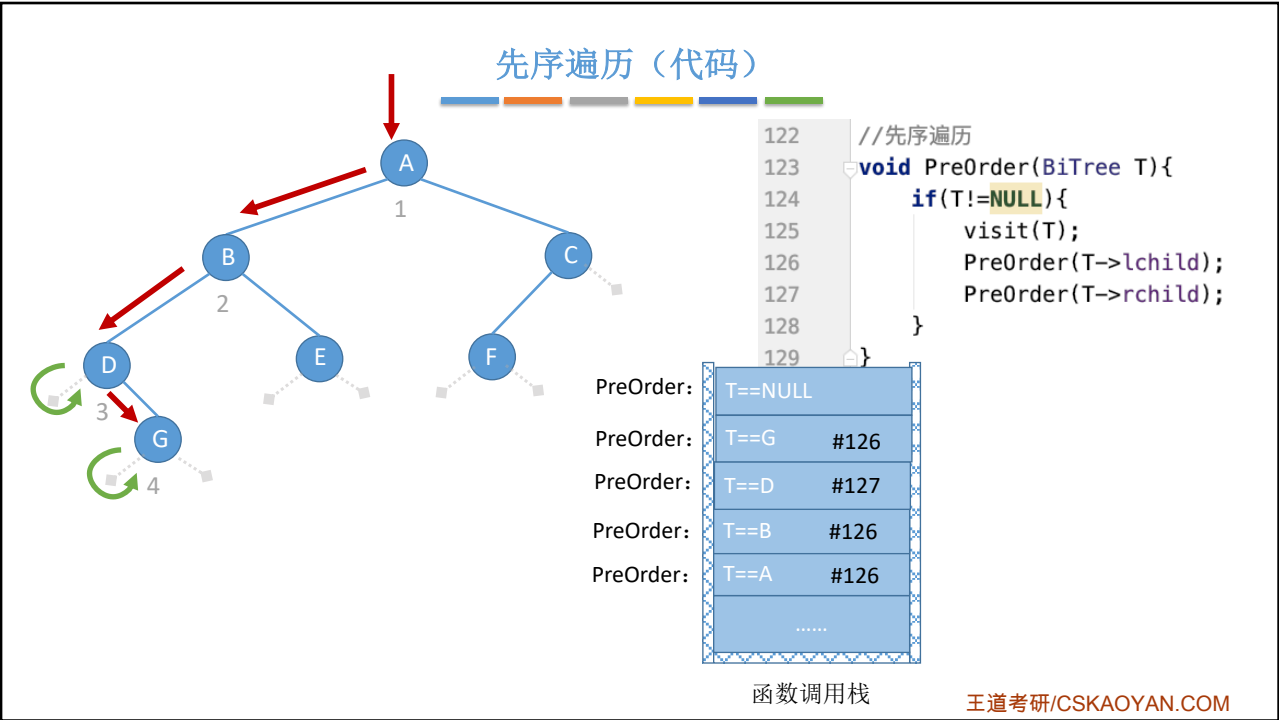
```
typedef struct BiTNode{
    ElemType data;
    struct BiTNode *lchild,*rchild;
}BiTNode,*BiTree;
```

王道考研/CSKAOYAN.COM

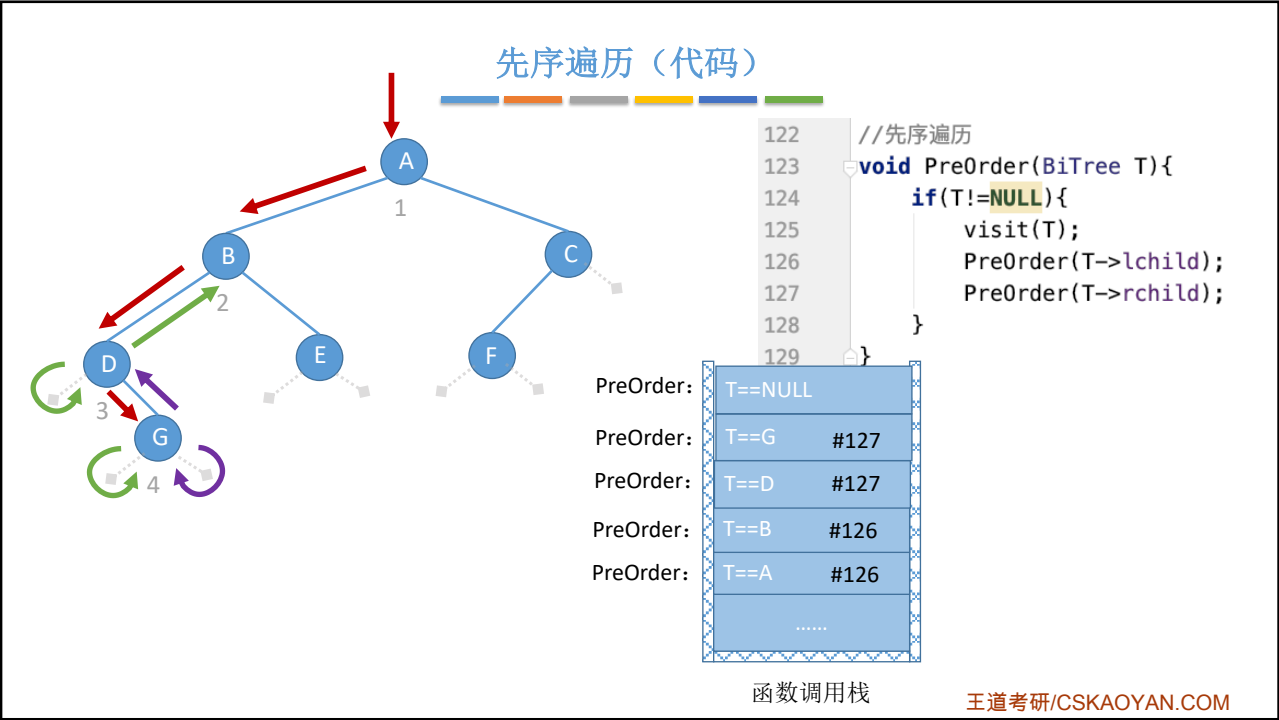
10



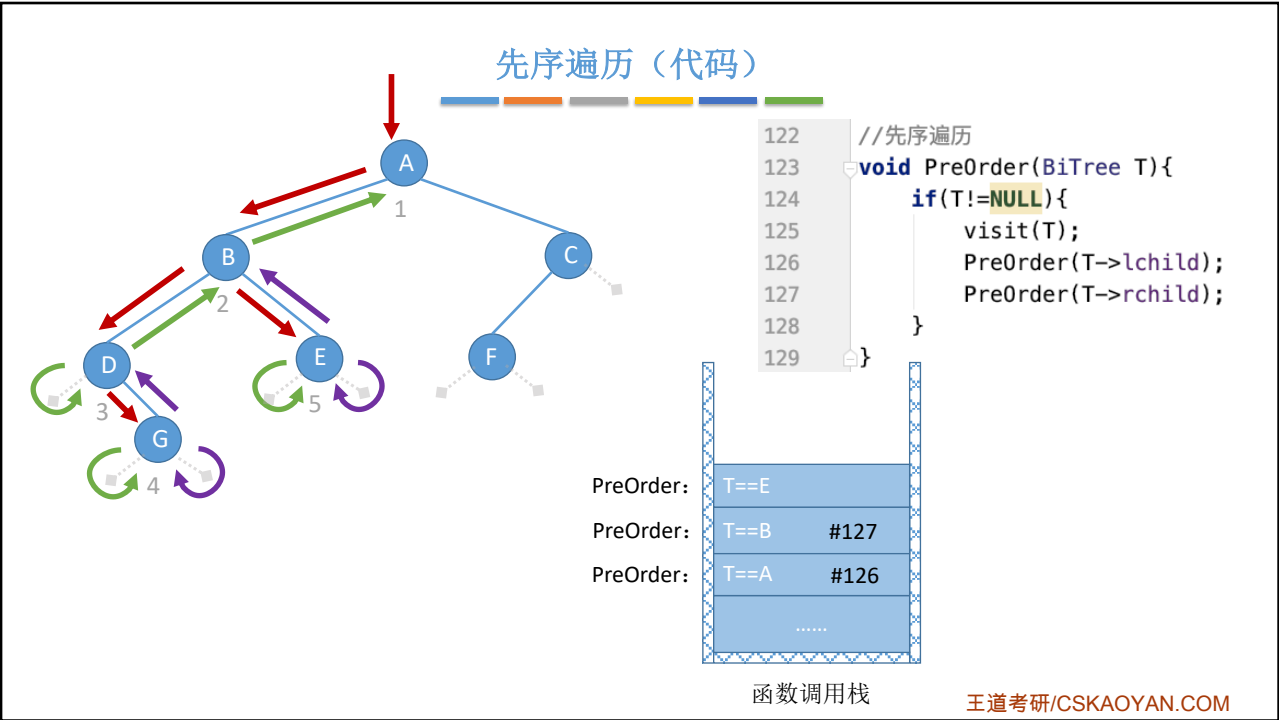
11



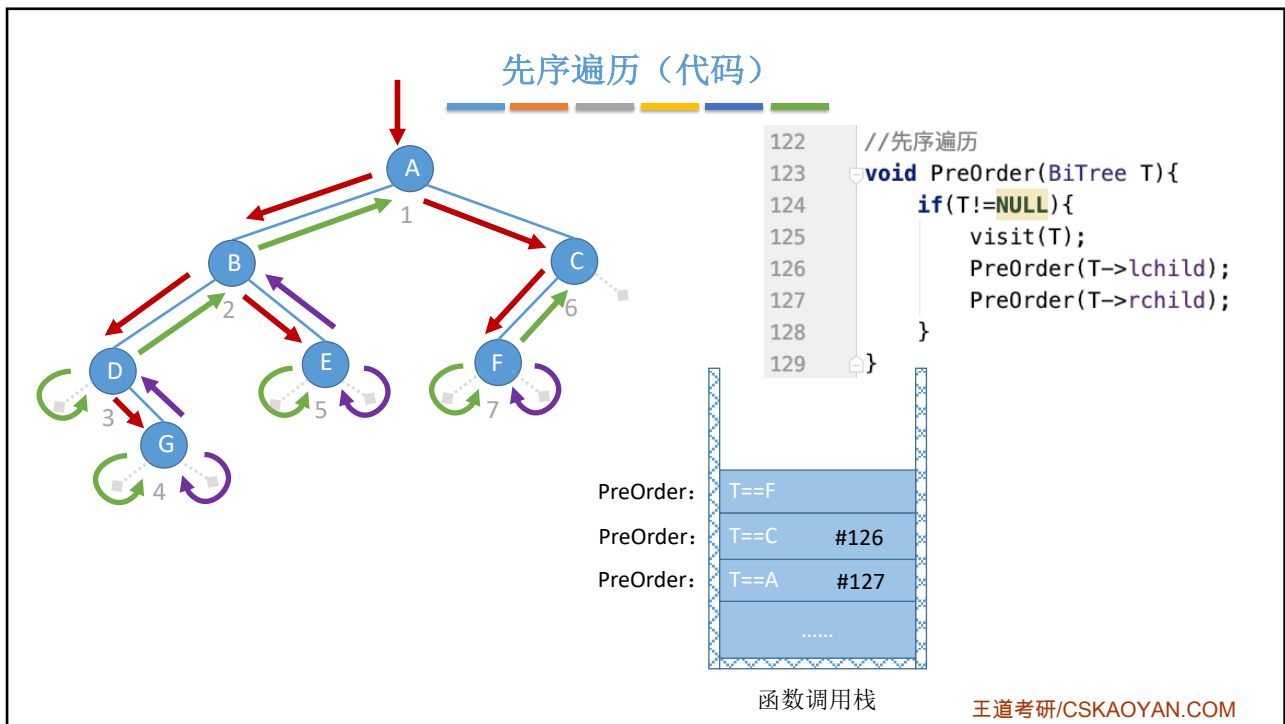
12



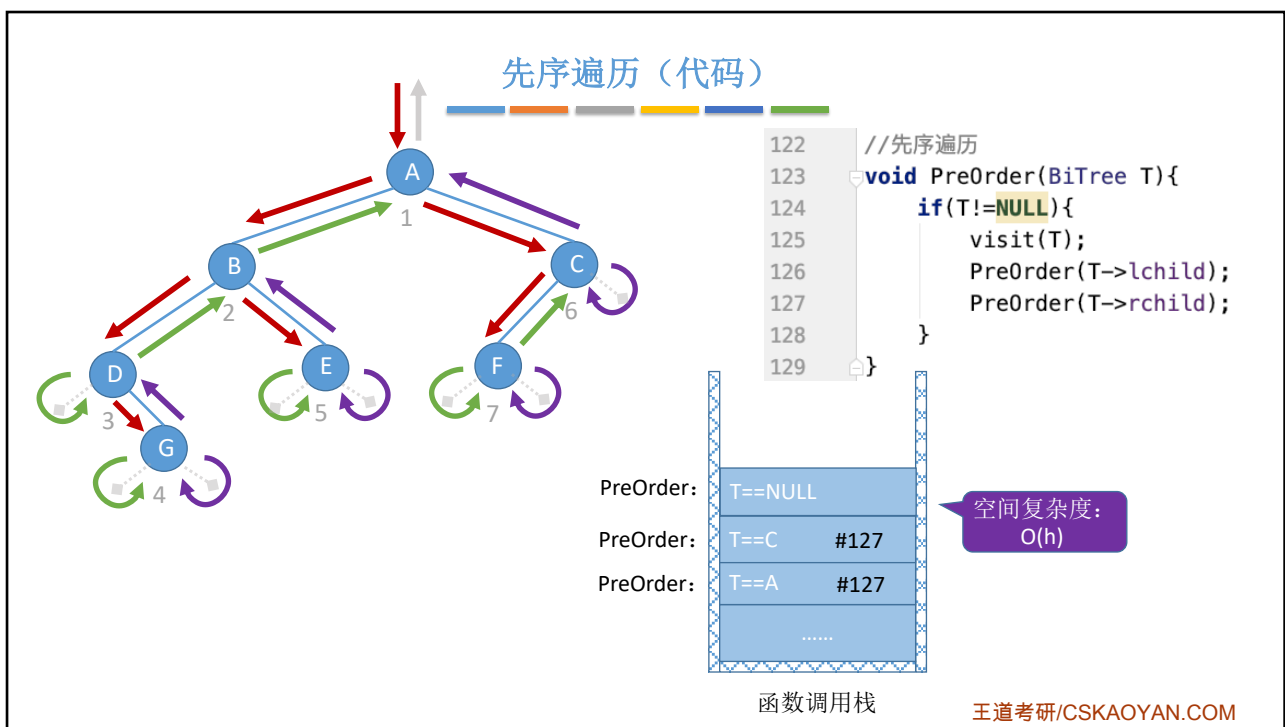
13



14



15



16



### 求先序遍历序列

图示说明:  
第一次路过 →  
第二次路过 →  
第三次路过 →

每个结点都会被路过3次

先序遍历 (PreOrder) 的操作过程如下:  
1. 若二叉树为空, 则什么也不做;  
2. 若二叉树非空:  
    ①访问根结点;  
    ②先序遍历左子树;  
    ③先序遍历右子树。

脑补空结点, 从根节点出发, 画一条路:  
如果左边还有没走的路, 优先往左边走  
走到路的尽头 (空结点) 就往回走  
如果左边没路了, 就往右边走  
如果左、右都没路了, 则往上面走  
先序遍历——第一次路过时访问结点

王道考研/CSKAOYAN.COM

17

### 求中序遍历序列

图示说明:  
第一次路过 →  
第二次路过 →  
第三次路过 →

每个结点都会被路过3次

中序遍历 (InOrder) 的操作过程如下:  
1. 若二叉树为空, 则什么也不做;  
2. 若二叉树非空:  
    ①先序遍历左子树;  
    ②访问根结点;  
    ③先序遍历右子树。

脑补空结点, 从根节点出发, 画一条路:  
如果左边还有没走的路, 优先往左边走  
走到路的尽头 (空结点) 就往回走  
如果左边没路了, 就往右边走  
如果左、右都没路了, 则往上面走  
中序遍历——第二次路过时访问结点

王道考研/CSKAOYAN.COM

18

从你的全世界路过法...

求后序遍历序列

后序遍历 (InOrder) 的操作过程如下:

1. 若二叉树为空, 则什么也不做;
2. 若二叉树非空:
  - ①先序遍历左子树;
  - ②先序遍历右子树;
  - ③访问根结点。

脑补空结点, 从根节点出发, 画一条路:  
如果左边还有没走的路, 优先往左边走  
走到路的尽头 (空结点) 就往回走  
如果左边没路了, 就往右边走  
如果左、右都没路了, 则往上面走  
后序遍历——第三次路过时访问结点

图示说明:  
第一次路过 →  
第二次路过 →  
第三次路过 →

每个结点都会被路过3次

王道考研/CSKAOYAN.COM

19

二叉树的遍历 (手算练习)

先序遍历:  $-+a*b-cd/ef$   
中序遍历:  $a+b*c-d-e/f$   
后序遍历:  $abcd-*+ef/-$

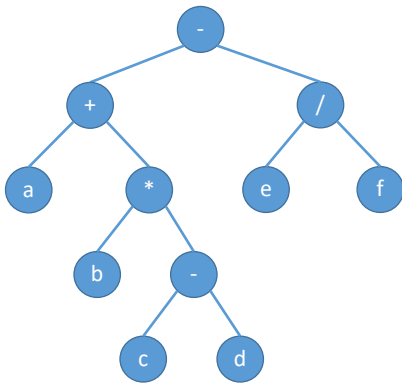
先序遍历 → 前缀表达式  
中序遍历 → 中缀表达式 (需要加界限符)  
后序遍历 → 后缀表达式

算数表达式的“分析树”  
 $a + b * (c - d) - e / f$

王道考研/CSKAOYAN.COM

20

## 例：求树的深度（应用）



```

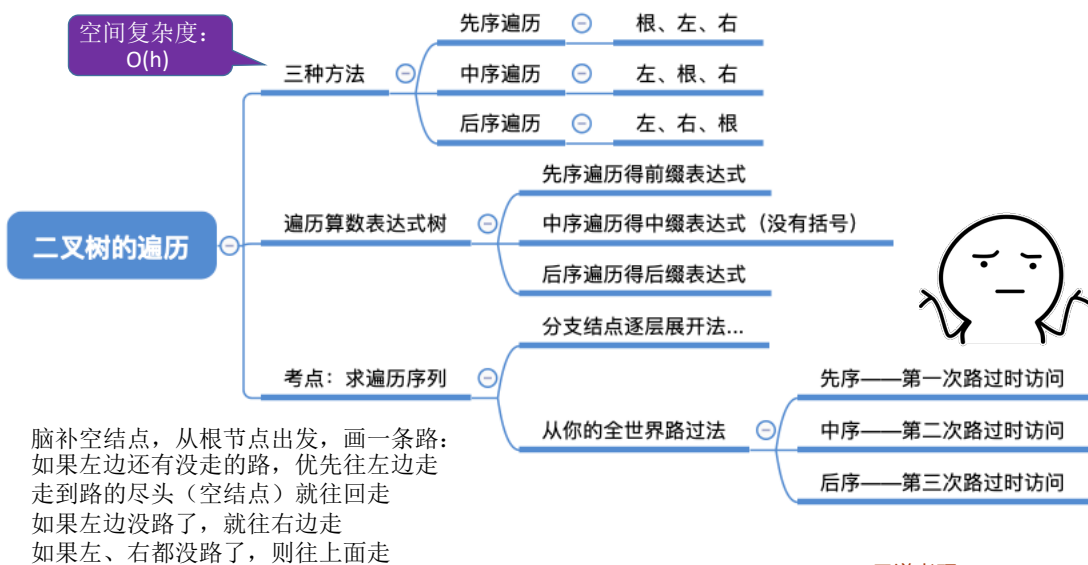
int treeDepth(BiTree T){
    if (T == NULL) {
        return 0;
    }
    else {
        int l = treeDepth(T->lchild);
        int r = treeDepth(T->rchild);
        //树的深度=Max(左子树深度, 右子树深度)+1
        return l>r ? l+1 : r+1;
    }
}

```

王道考研/CSKAOYAN.COM

21

## 知识回顾与重要考点



王道考研/CSKAOYAN.COM

22