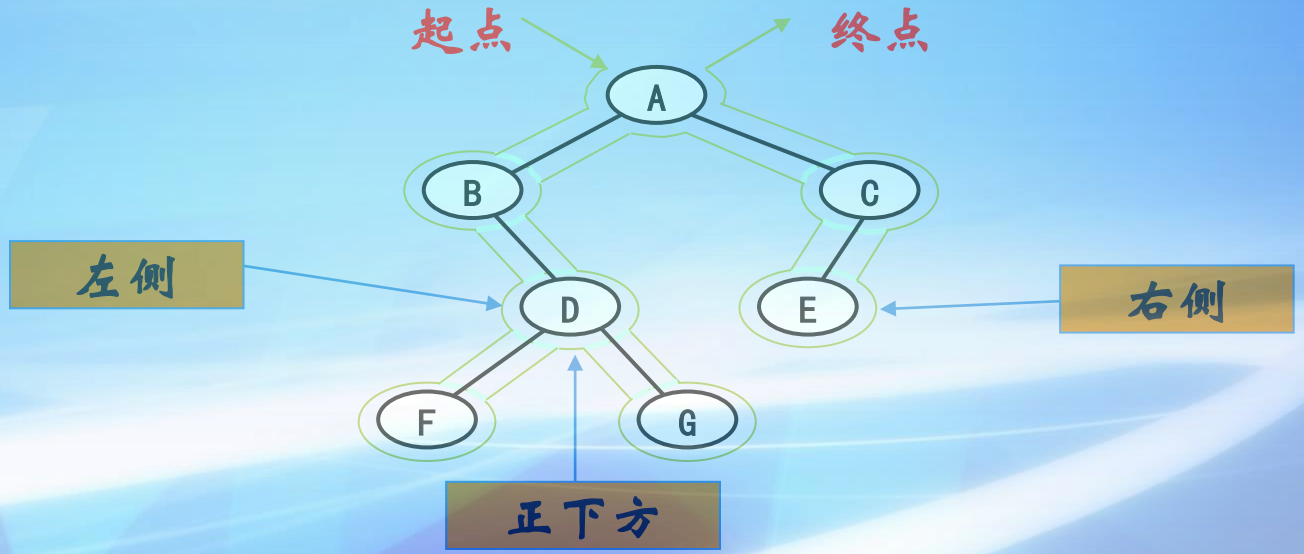
二叉树的遍历

《数据结构》

主讲人：李清



教学目标和要求

1. 能够准确描述前（中、后）序三种递归的遍历算法思想；
2. 能够用C语言编程实现三种递归的遍历算法；
3. 对于给定二叉树，能够图示递归遍历过程中栈的变化。

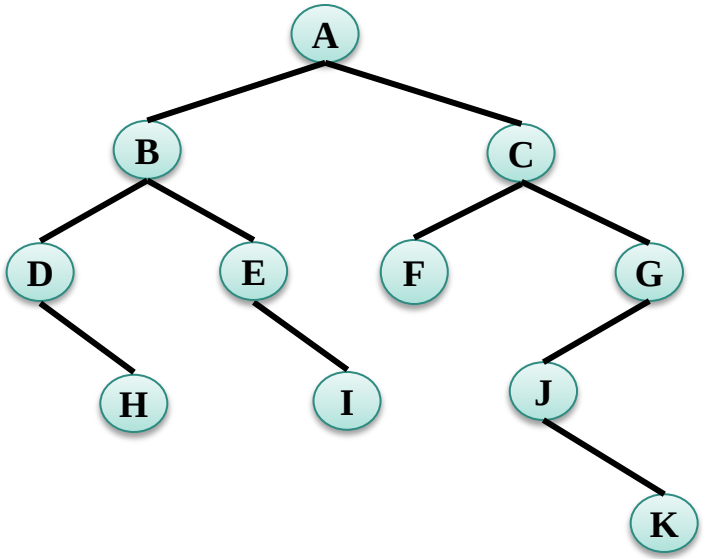
二叉树的遍历

遍历　按一定的规律“访问”二叉树中的每个结点，且只访问一次。

二叉树　遍　历　线性表

先根遍历

先访问二叉树的根，然后访问二叉树的左子树，再访问二叉树的右子树。对二叉树左、右子树的访问也是“先根遍历”。



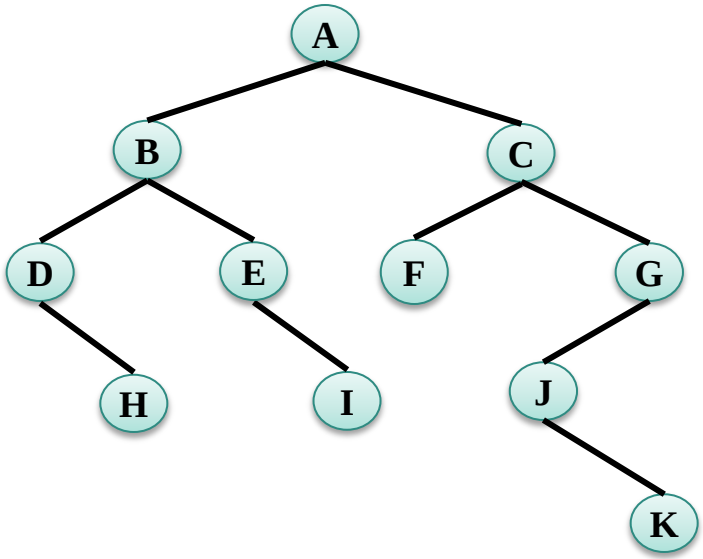
DLR序列：

ABDHEICFGJK

第一个结点是根结点

中根遍历

先访问二叉树的左子树，然后访问二叉树的根，再访问二叉树的右子树。对二叉树左、右子树的访问也是“中根遍历”。

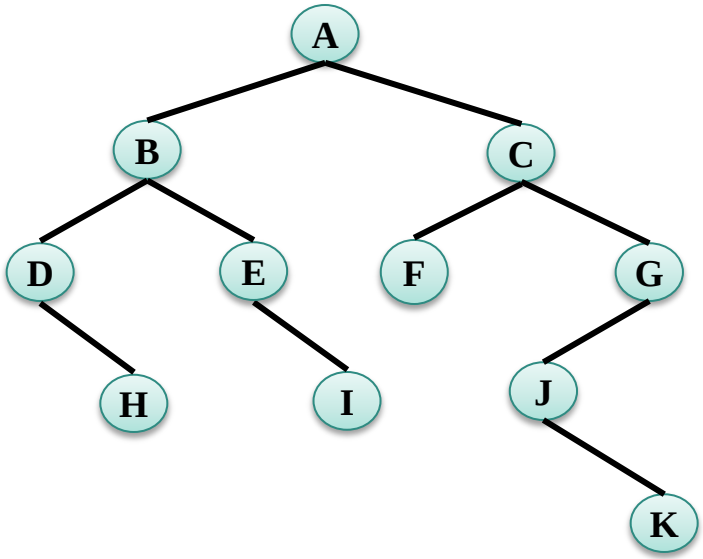


LDR序列：

DHBEIAFCJKG

后根遍历

先访问二叉树的左子树，然后访问二叉树的右子树，最后访问二叉树的根。对二叉树左、右子树的访问也是“后根遍历”。



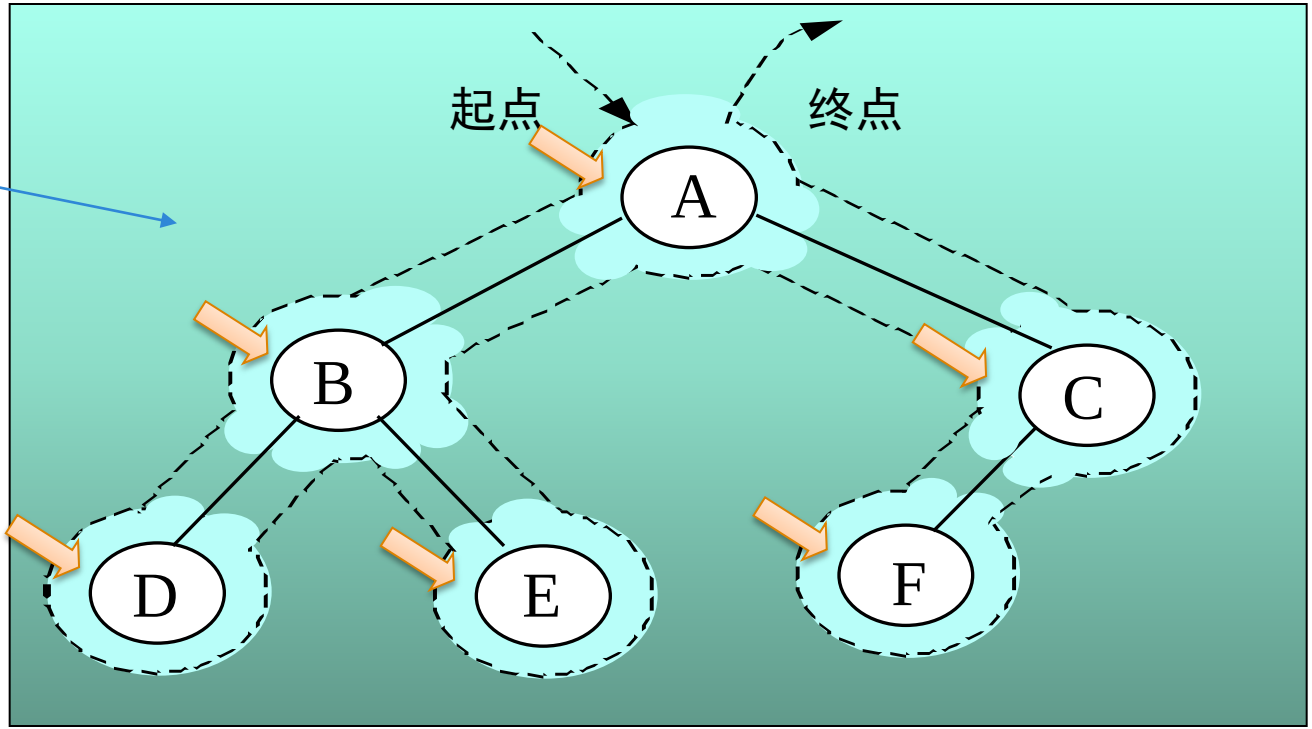
LRD序列：

HDIEBFKJGCA

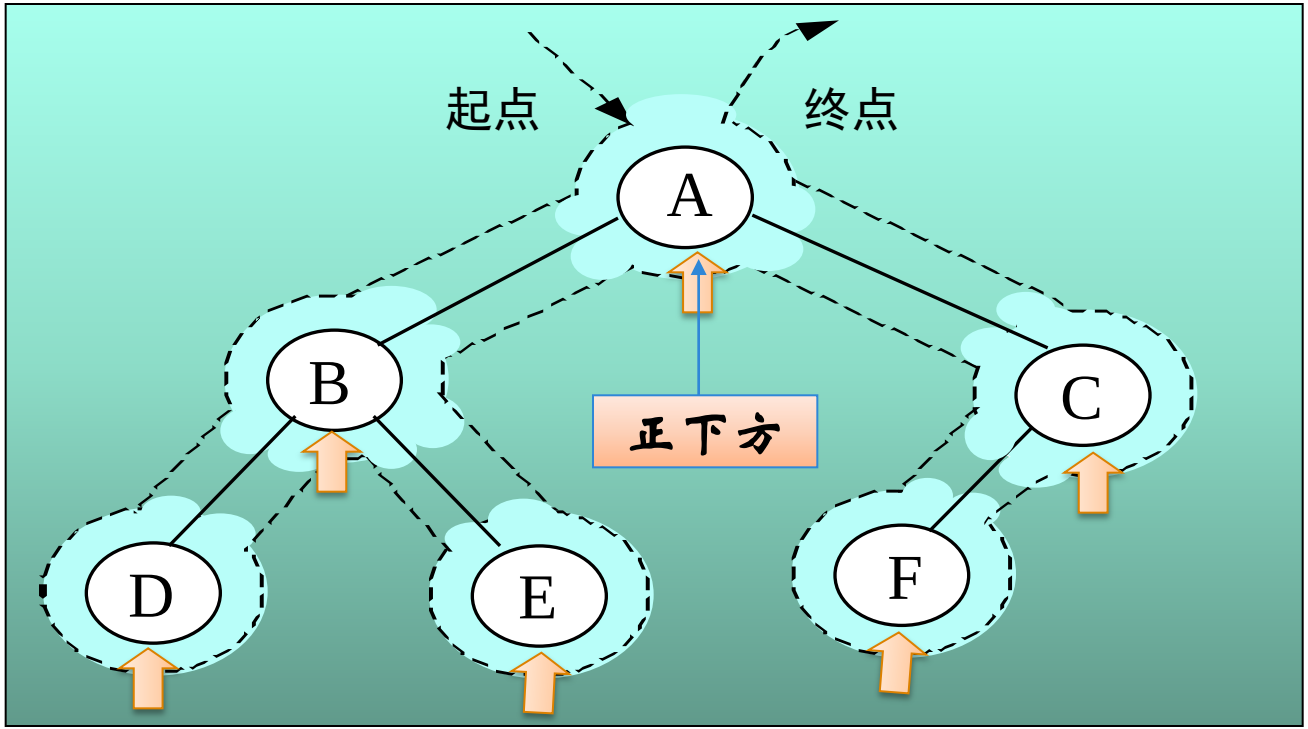
最后一个结点是根结点

二叉树的遍历

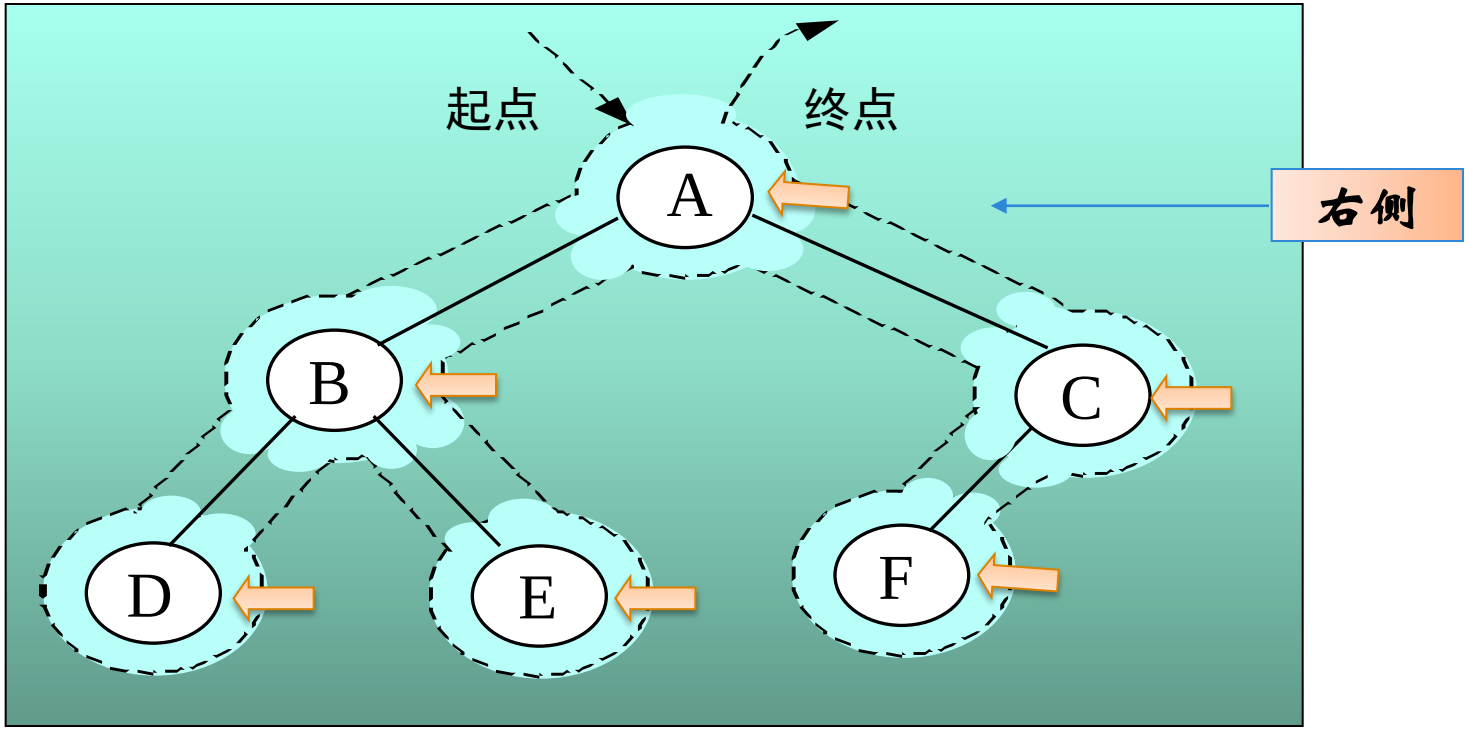
左侧



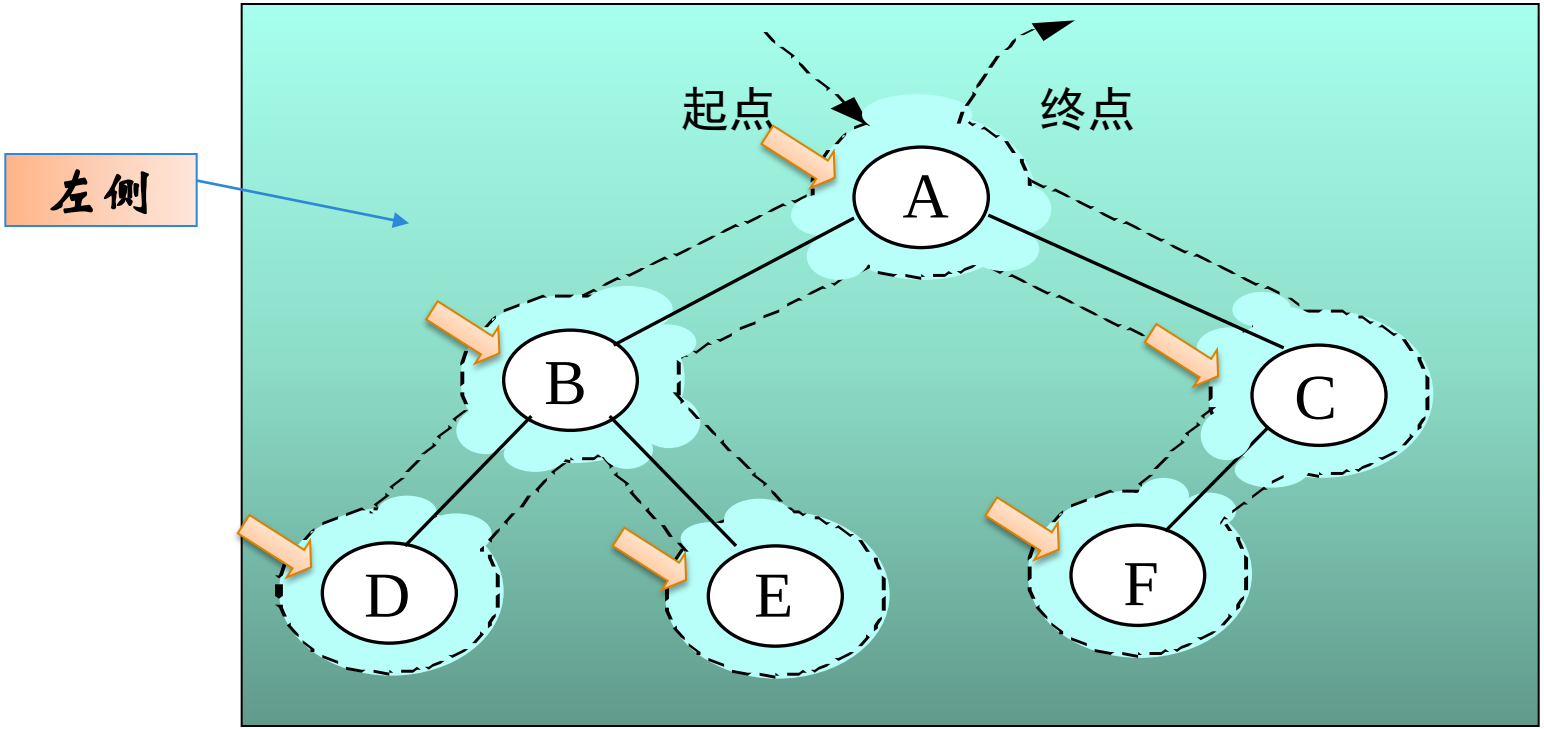
二叉树的遍历



二叉树的遍历

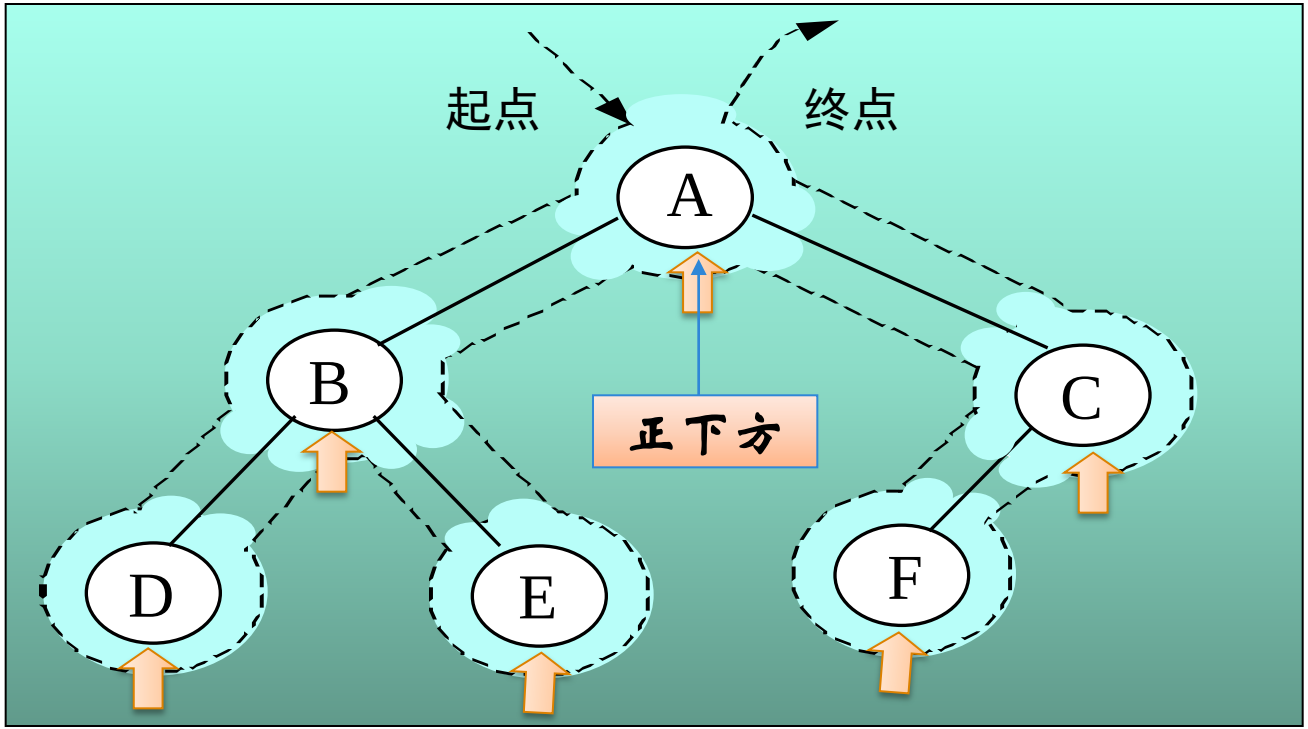


二叉树的遍历



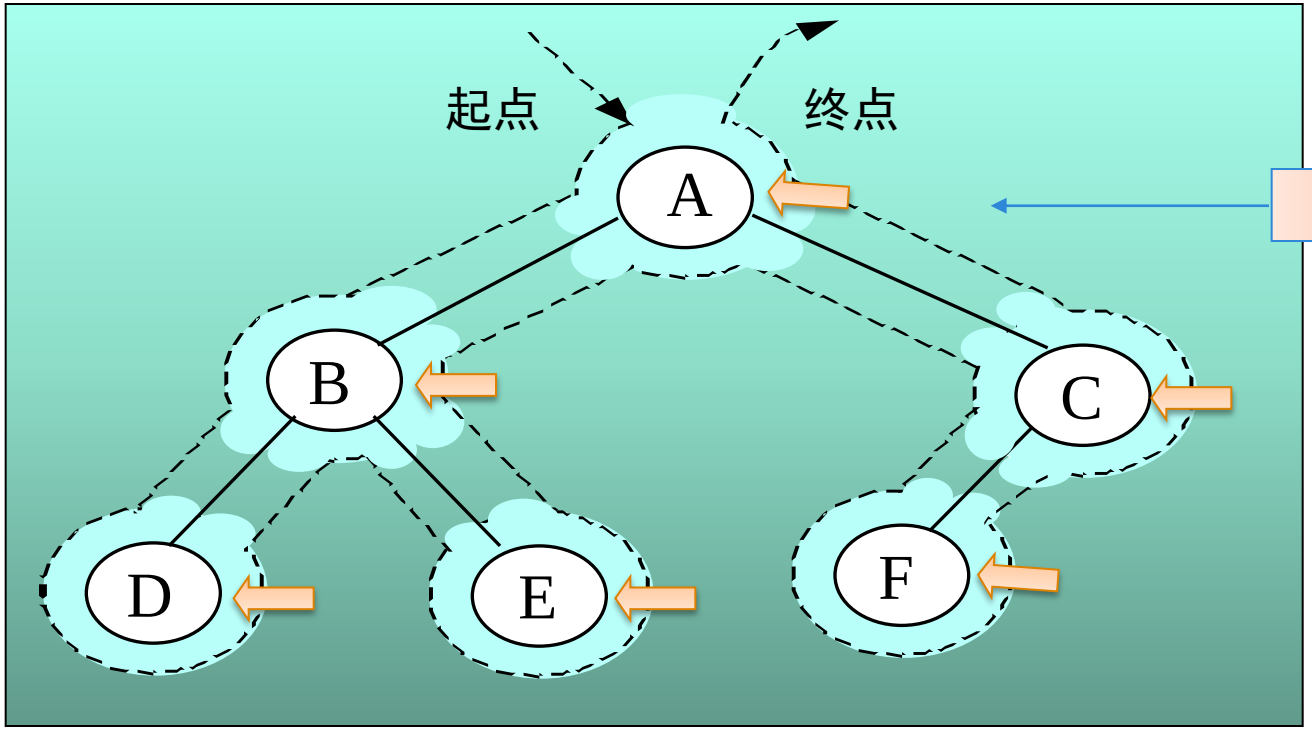
先序序列为：　A　B　D　E　C　F

二叉树的遍历



中序序列为：　　D B E A F C

二叉树的遍历



后序序列为：　　D E B F C A

二叉树的遍历----递归遍历算法思想

先序遍历算法

如果当前正在遍历的二叉树（包括子树）为空，则什么也不做，否则，

（D）访问该二叉树的根结点；

（L）递归地遍历根的左子树；

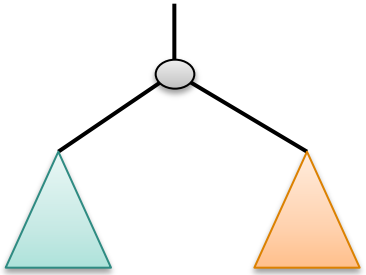
（R）递归地遍历根的右子树；

对该二叉树（包括子二叉树）的遍历结束。

将上述算法中的（D）移到（L）的后面，得到中序遍历：（L）（D）（R）；

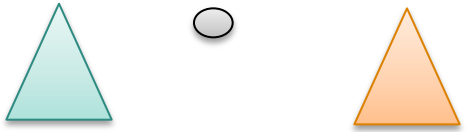
将上述算法中的（D）移到（R）的后面，得到后序遍历：（L）（R）（D）。

二叉树的遍历----递归遍历算法思想



先序遍历

中序遍历



后序遍历

问题

如何编程实现二叉树的遍历？

二叉树的存储描述

定义双链表结构

```
typedef     int  element_type ;

typedef     struct  Bnode
 {  element_type   data;
     struct  Bnode  *Lson, *Rson;
 }  Bnode, *Bptr;
```

二叉树的遍历----递归遍历算法实现

```
void preorder(Bptr p)//先序遍历DLR
{
    if(!P) return;
    visit(p);
    preorder(p->Lson);
    preorder(p->Rson);
}//T(n)=O(n)
```

空返回

递归

```
main()
{  ……
   preorder(root);
   ……
}
```

计算对n个结点的二叉树遍历所需时间

能否改进？

- 每个结点都要访问一次，共有n次（非空）调用；
- 每个空子树也要递归调用一次，共有n+1次空调用；
- 每次递归调用最多执行4条语句，所以算法总的执行语句是O(n+1+4n)条。

时间复杂性： T(n)=O(n)

二叉树的遍历----递归遍历算法实现

```
void preorder(Bptr p)//先序遍历DLR
{
    if(!P) return;
    visit(p);
    if(p->Lson)preorder(p->Lson);
    if(p->Rson)preorder(p->Rson);
}//T(n)=O(n)
```

消除空调用

```
main()
{ ……
  if(root)preorder(p->Rson);
  ……
}
```

二叉树的遍历----递归遍历算法实现

消除空调用

```
void preorder_1(Bptr p)//DLR
{  visit(p);                                  //1
   if(p->Lson)preorder_1(p->Lson);   //2
   if(p->Rson)preorder_1(p->Rson);  //3
}//T(n)=O(n)                                 //4
```

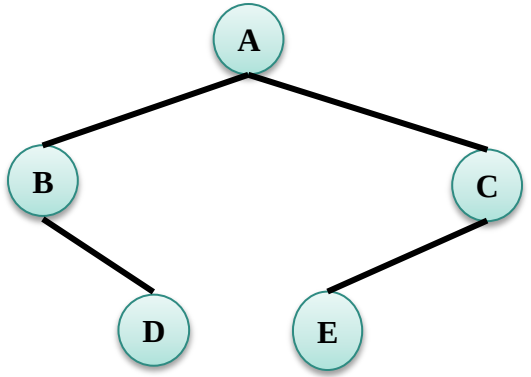
问题

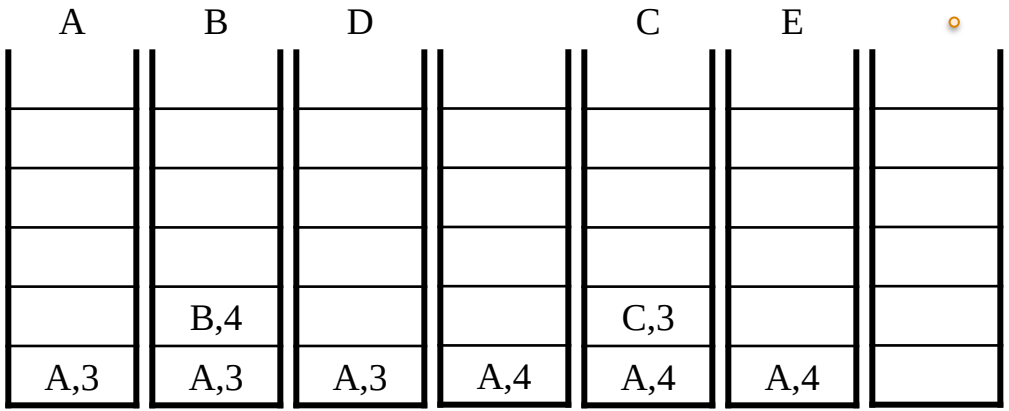
递归遍历算法是如何实现的呢？

二叉树的遍历----递归遍历算法实现

栈

```
void preorder_1(Bptr p)//DLR
{ 1. visit(p);
   2. if(p->Lson)preorder_1(p->Lson);
   3. if(p->Rson)preorder_1(p->Rson);
4.}//T(n)=O(n)
```

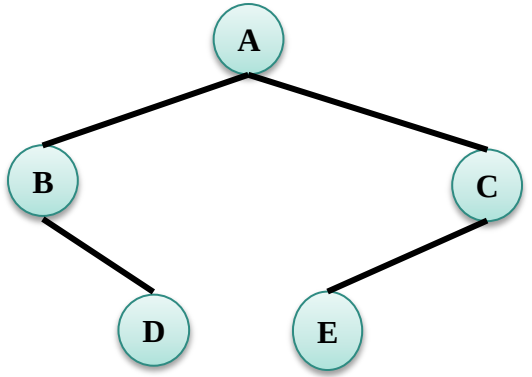


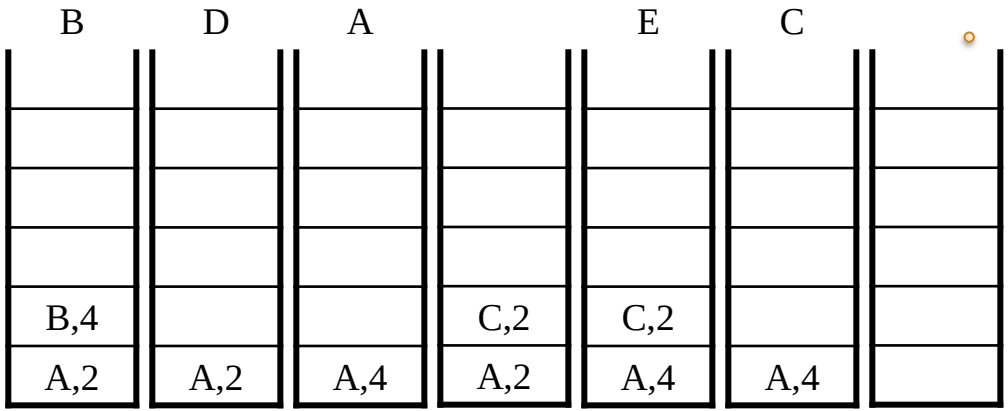


先序遍历序列

二叉树的遍历----递归遍历算法实现

```
void inorder_1(Bptr p)//LDR
{  if(p->Lson)inorder_1(p->Lson);   //1
   visit(p);                        //2
   if(p->Rson)inorder_1(p->Rson);  //3
}//T(n)=O(n)                       //4
```

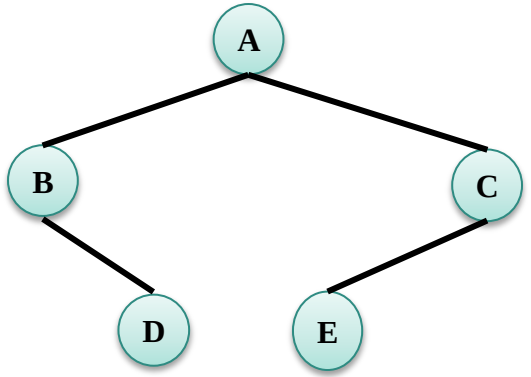


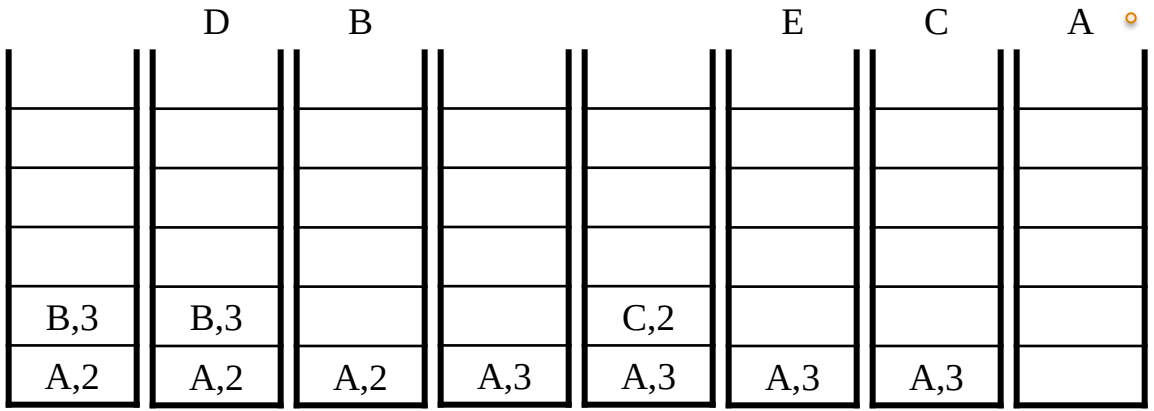


中序遍历序列

二叉树的遍历----递归遍历算法实现

```
void suorder_1(Bptr p)//LRD
{  if(p->Lson)suorder_1(p->Lson);   //1
   if(p->Rson)suorder_1(p->Rson);  //2
   visit(p);                        //3
}//T(n)=O(n)                       //4
```



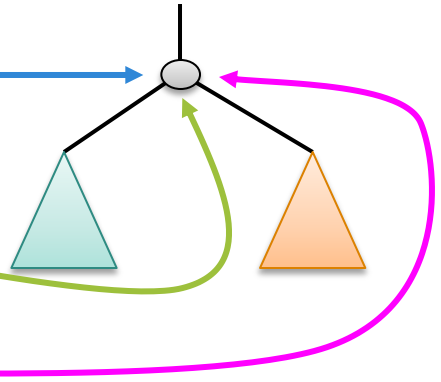


后序遍历序列

二叉树的遍历----递归遍历算法实现（含空调用）

```
void traversal(Bptr p)
  {
      if(p= =NULL) return;
      先序访问(p);
      traversal(p->Lson);
      中序访问(p);
      traversal(p->Rson);
      后序访问(p);
  }
```

主调语句：
    traversal(root);



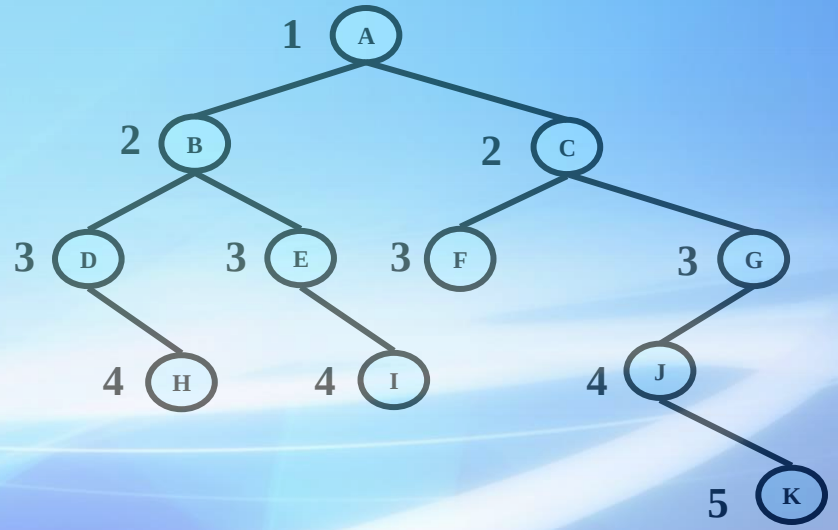
小结

- 二叉树的遍历通常有：先序、中序、后序三种遍历；
- 二叉树的三种遍历都可用递归算法编程实现；
- 不含空调用的递归遍历算法可减少递归调用次数。

二叉树遍历的应用

主讲人：李清

```
void level(Bptr p,int i)
{  if(!p) return;
   p->layer=i;
   level(p->Lson,i+1);
   level(p->Rson,i+1);
}
//求二叉树的结点的高
main()
{  ……
   level(root,1);
   ……
}
```



问题

二叉树的遍历能做什么？

教学目标和要求

1. 了解二叉树遍历的应用；
2. 能够通过二叉树的遍历编程实现求二叉树的叶子数，结点的度、层数、高度、子孙等。

求二叉树各结点的层数

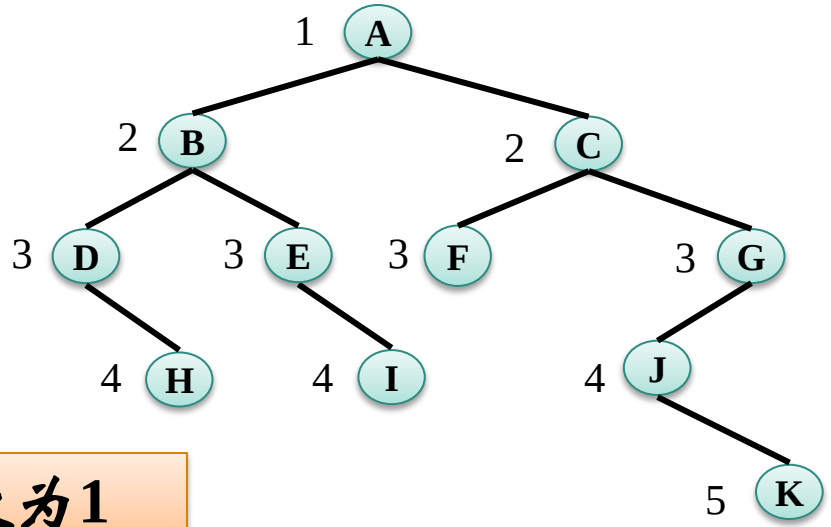
```
void level(Bptr p,int i)
{  if(!p) return;
   p->layer=i;
   level(p->Lson,i+1);
   level(p->Rson,i+1);
}

main()
{  ……
   level(root,1);
   ……
}
```

传值

先序

根结点的层数为1

Lson	data	layer	Rson



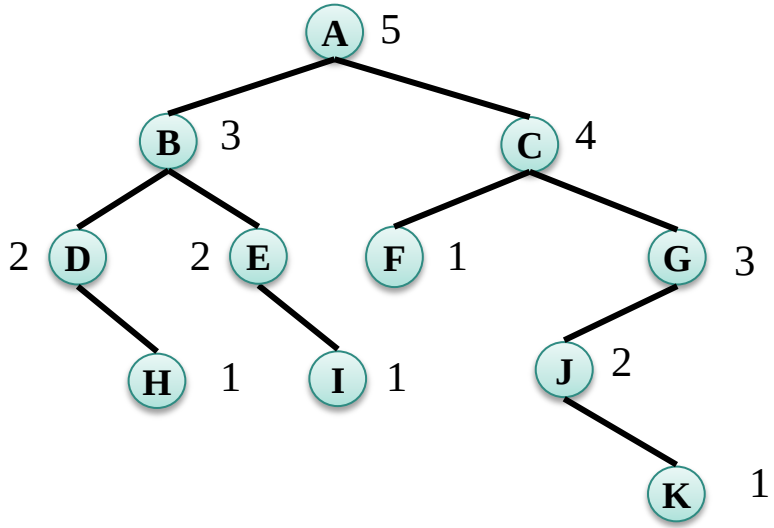
求二叉树各结点的高度

```
int high(Bptr p)
{  int i,j;
   if(!p) return 0;
   i=high(p->Lson);
   j=high(p->Rson);
   p->height=(i>j)?i+1;j+1;
   return (p->height);
}
main()
{  ……
   h=high(root);
   ……
}
```

空树的高度为0

后序

Lson	data	height	Rson



求二叉树中结点a子孙

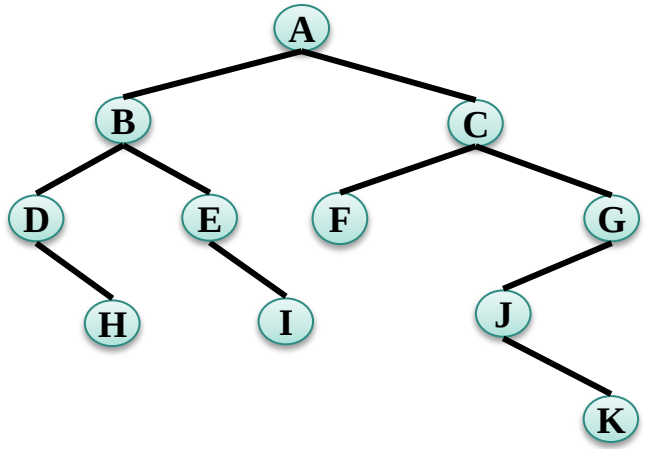
```
void descents(Bptr p,element_type a))
{  if(!p) return;
   if(p->data==a) found=1;// a子孙开始
   if(found==1) printf("%4d",p->data);
   descents(p->Lson,a);
   descents(p->Rson,a);
   if(p->data==a) found=0;// a子孙结束
}
main()
{  ……
   found=0;descents(root,a);
   ……
}
```

以a='B'为例

先序

后序

Lson	data	Rson



遍历序列ABDHEICFGJK

输出序列BDHEI

求二叉树中结点a子孙

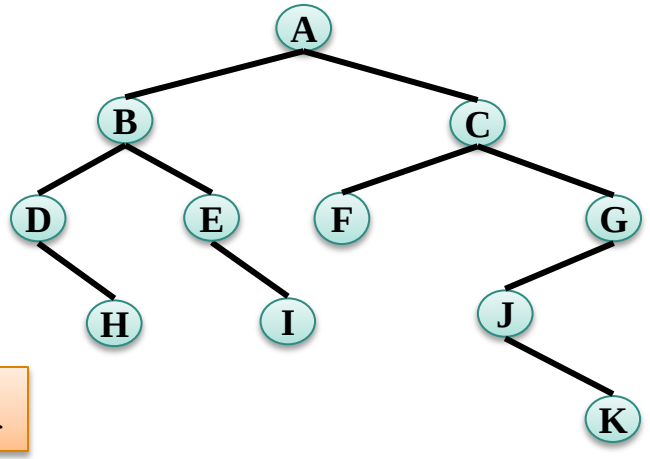
```
void descents(Bptr p,element_type a))
{  if(!p) return;
   if(p->data==a) found=1;// a子孙开始
   if(found ==1)  printf("%4d",p->data);
   descents(p->Lson,a);
   descents(p->Rson,a);
   if(p->data==a) found=0;// a子孙结束
}
main()
{  ……
   found= 2 ;descents(root,a);
   ……
}
```

if(found==0)return;

遍历完a的所有子孙后结束

以a='B'为例

Lson	data	Rson



遍历序列ABDHEI

输出序列BDHEI

小结

- 先序遍历可求二叉树各结点层数
- 后序遍历求二叉树各结点的高度
- 先序和后序相结合求给定结点的所有子孙
- 采用何种遍历取决于对二叉树进行的操作