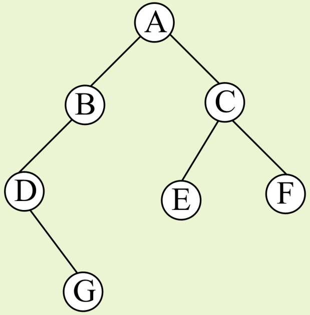
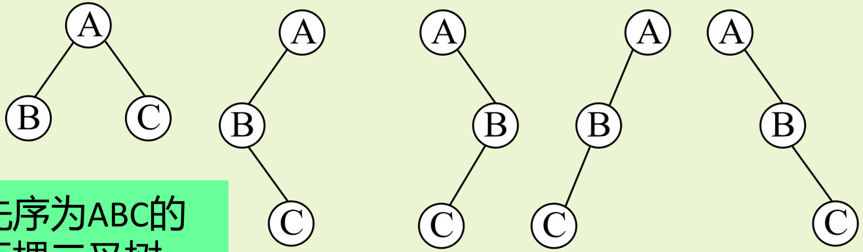
本节主题：

二叉树的构造

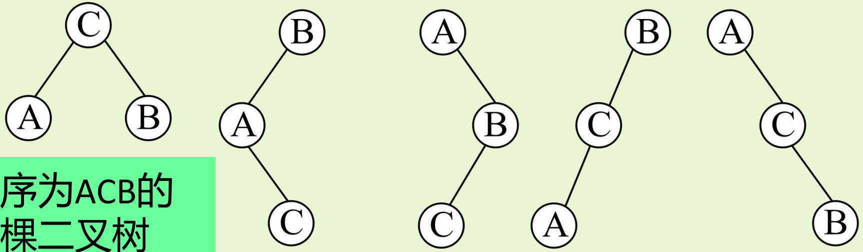
一些直观的认识



- 同一棵二叉树具有唯一先序序列、中序序列和后序序列。
- 不同的二叉树可能具有相同的先序序列、中序序列和后序序列。



先序为ABC的
五棵二叉树



中序为ACB的
五棵二叉树

- 仅由一个先序序列（或中序序列、后序序列），无法确定这棵二叉树的树形！
- 思考：给定先序、中序和后序遍历序列中任意两个，是否可以唯一确定这棵二叉树的树形？

二叉树的构造

- 同时给定一棵二叉树的先序序列和中序序列，就能唯一确定这棵二叉树？
  - 是！
  - 定理1
- 同时给定一棵二叉树的中序序列和后序序列，就能唯一确定这棵二叉树？
  - 是
  - 定理2
- 同时给定一棵二叉树的先序序列和后序序列，就能唯一确定这棵二叉树？
  - 否

定理1

- 定理
  - 任何n（n≥0）个不同节点的二叉树，都可由它的中序序列和先序序列唯一地确定。
- 证明（数学归纳法）
  - 基础：当n=0时，二叉树为空，结论正确。
  - 假设：设节点数小于n的任何二叉树，都可以由其先序序列和中序序列唯一地确定。
  - 归纳：已知某棵二叉树具有n（n＞0）个不同节点，其先序序列是$a_0a_1…a_{n-1}$；中序序列是$b_0b_1…b_{k-1}b_kb_{k+1}…b_{n-1}$。
    - 先序遍历“根-左-右”，a_0必定是二叉树的根节点
    - a_0必然在中序序列中出现，设在中序序列中必有某个b_k（0≤k≤n-1）就是根节点a_0。

根据a_0可以找到b_k

根节点a_0 ⟷ 根节点a_0

先序序列是$a_0a_1…a_{n-1}$

中序序列是$b_0b_1…b_{k-1}b_kb_{k+1}…b_{n-1}$

定理证明(续)

- 由于b_k是根节点，中序遍历“左-根-右”，故中序序列中
  - $b_0b_1...b_{k-1}$必是根节点$b_k(a_0)$左子树的中序序列，即b_k的左子树有k个节点
  - $b_{k+1}...b_{n-1}$必是根节点b_k（a_0）右子树的中序序列，即b_k的右子树有n-k-1个节点。
- 对应先序序列，紧跟在根节点a_0之后的k个节点$a_1...a_k$是左子树的先序序列，$a_{k+1}...a_{n-1}$这n-k-1就是右子树的先序序列。
- 根据归纳假设，子先序序列$a_1...a_k$和子中序序列$b_0b_1...b_{k-1}$可以唯一地确定根节点a_0的左子树，而先序序列$a_{k+1}...a_{n-1}$和子中序序列$b_{k+1}...b_{n-1}$可以唯一地确定根节点a_0的右子树。
- 综上所述，这棵二叉树的根节点已经确定，而且其左、右子树都唯一地确定了，所以整个二叉树也就唯一地确定了。

根节点a_0 $\xleftrightarrow{\text{根据}a_0\text{可以找到}b_k}$ 根节点a_0

先序序列是$a_0\underbrace{a_1...a_k}_{\text{左子树先序序列，有k个结点}}\underbrace{a_{k+1}...a_{n-1}}_{\text{右子树先序序列，有n-k-1个结点}}$

中序序列是$\underbrace{b_0b_1...b_{k-1}}_{\text{左子树中序序列，有k个结点}}b_k\underbrace{b_{k+1}...b_{n-1}}_{\text{右子树中序序列，有n-k-1个结点}}$

例：先序ABDGCEF，中序DGBAECF，二叉树？

根节点：A
左先序：BDG；左中序：DGB
右先序：CEF；右中序：ECF

根节点：B
左先序:DG 左中序:DG
右先序:空 右中序:空

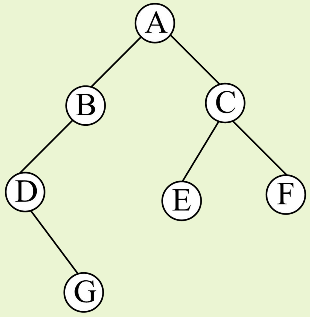
根节点：C
左先序:E 左中序:E
右先序:F 右中序:F

根节点：D
左先序:空 左中序:空
右先序:G 右中序:G

根节点：E
左先序:空 左中序:空
右先序:空 右中序:空

根节点：F
左先序:空 左中序:空
右先序:空 右中序:空

根节点：G
左先序:空 左中序:空
右先序:空 右中序:空



算法实现

```
BTNode *CreateBT1(char *pre, char *in, int n)
{
   BTNode *s;
   char *p;
   int k;
   if (n<=0) return NULL;
   s=(BTNode *)malloc(sizeof(BTNode));
   s->data=*pre;
   //在中序中找根节点的位置k
   //构造左、右子树
   return s;
}
```

pre	A	B	D	G	C	E	F	
in	D	G	B	A	E	C	F	

```
for (p=in; p<in+n; p++)
   if (*p==*pre)
      break;
k=p-in;
```

```
s->lchild=CreateBT1(pre+1,in,k);
s->rchild=CreateBT1(pre+k+1,p+1,n-k-1);
```

根节点

先序序列是$a_0a_1 \ldots a_k a_{k+1} \ldots a_{n-1}$

左子树先序序列，有k个结点

右子树先序序列，有n-k-1个结点

根节点

中序序列是$b_0b_1 \ldots b_{k-1} b_k b_{k+1} \ldots b_{n-1}$

左子树中序序列，有k个结点

右子树中序序列，有n-k-1个结点

定理2

- 定理
  - 任何n（n＞0）个不同节点的二叉树，都可由它的中序序列和后序序列唯一地确定。
- 证明
  - （略）
- 算法
  - （略）

根据a_{n-1}可以找到b_k

根节点

后序序列是$a_0 \dots a_{k-1}\ a_k \dots a_{n-2}\ a_{n-1}$

左子树后序序列，有k个结点

右子树后序序列，有n-k-1个结点

根节点

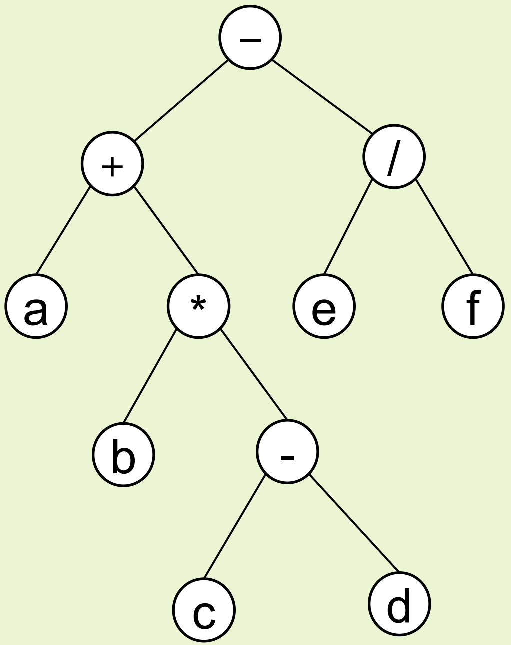
中序序列是$b_0 b_1 \dots b_{k-1} b_k b_{k+1} \dots b_{n-1}$

左子树中序序列，有k个结点

右子树中序序列，有n-k-1个结点

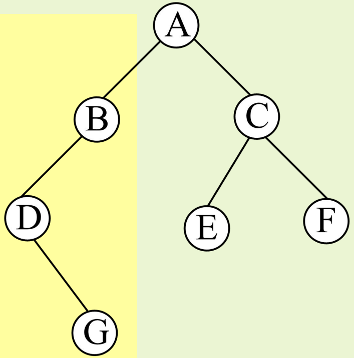
应用(例)

- 问题
  - 算法表达式：(a+b*(c-d)-e/f)
- 方案
  - 用二叉树存储
- 实现
  - 先序序列为：-+a*b-cd/ef
  - 中序序列为：a+b*c-d-e/f
  - 后序序列为：abcd-*+ef/-
- 必要性
  - 人们习惯中缀形式的算术表达式，对于计算机，使用后缀易于求值。



其他转换：由顺序存储结构转为二叉链存储结构

```
BTNode *trans(SqBTree a,int i)
{
    BTNode *b;
    if (i>MaxSize)
        return(NULL);
    if ([i]=='#') return(NULL);
    b=(BTNode *)malloc(sizeof(BTNode));
    b->data=a[i];
    b->lchild=trans(a,2*i);
    b->rchild=trans(a,2*i+1);
    return(b);
}
```



f(a,i)= {
NULL， 当i大于MaxSize
NULL，当i对应的节点为空
b(值为a[i]);
b->lchild = f(a, 2*i);
b->rchild = f(a, 2*i+1) ，其他情况

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	…
s	--	A	B	C	D	#	E	F	#	G	#	#

```
int main()
{
    BTNode *b;
    ElemType s[]="0ABCD#EF##G#";
    b=trans(s,1);
    ……
}
```