数据结构重点

堆栈与队列的应用

中缀转后缀/中缀表达式的计算

后缀转中缀/中缀表达式计算运算符关系表

	+	-	*	/	(	)	#
+	>	>	<	<	<	>	>
-	>	>	<	<	<	>	>
*	>	>	>	>	<	>	>
/	>	>	>	>	<	>	>
(	<	<	<	<	<	=	
)	>	>	>	>		>	>
#	<	<	<	<	<		<

中缀转后缀算法：（栈实现）（中缀转前缀类似，只是把字符是数输出压入一个字符串的栈中，操作数弹出的话要先pop两个连接后再压）

- 先压一个'#'
- 遍历字符串：
  - 当前字符是数，输出；
  - 当前字符是运算符：
    - 查表table[栈顶字符，当前字符]
      - 是 '<'：当前字符压栈
      - 是 '='：pop一个（即左括号出栈，拆括号）
      - 是 '>'：pop栈顶，输出，在继续查表判断
- 直到栈顶的'#'遇到了输入字符的'#'

后缀转前缀：（前缀->后缀，后缀->中缀，前缀->中缀类似，不过注意括号）

- 初始化一个装字符串的栈
- 遍历后缀的字符串：
  - 若不是操作符，转成字符串压栈
  - 若是操作符，弹出两个字符串a1，a2，连接成一个新的字符串(e, a1, a2)压栈（e是操作符）
- 直到字符串末尾
- 前缀在栈顶

3.27非递归不会做（

串匹配算法

Brute-Force

KMP

树与二叉树

二叉树的性质

- 性质1：二叉树的第i层至多有2^{i-1}个结点($i \geq 1$)
- 性质2：深度为h的二叉树至多有2^h-1个结点($h \geq 1$)
  思考：性质1和性质2推广到k叉树，结果会如何？
  k^{i-1} $(k^h-1)/(k-1)$
- 性质3：$n_0 = n_2 + 1$
  结点总数 $n = n_0 + n_1 + n_2$
  分支数 $n-1 = n_1 + 2n_2$
  思考：若包含有n个结点的树中只有叶子结点和度为k的结点，则该树中有多少叶子结点？

$n = n_0+n_k, n-1 = kn_k \Rightarrow n_0 = n-(n-1)/k$

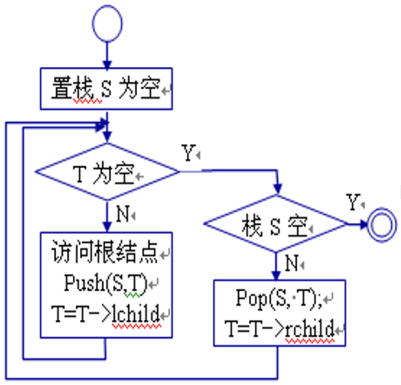
14/106

- 性质4：具有n个结点的完全二叉树的深度为$\lfloor \log_2 n \rfloor + 1$
  由性质2 $2^{k-1}-1 < n \leq 2^k-1$ 或 $2^{k-1} \leq n < 2^k$
  于是 $k-1 \leq \log_2 n < k$

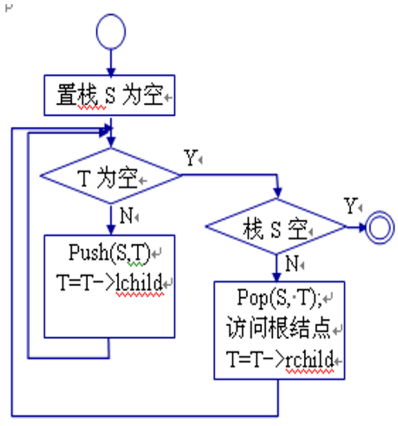
例：若一个完全二叉树有1450个结点，则度为1的结点个数为 1 ，度为2的结点个数为 724 ，叶子结点的个数为 725 ，有 725 个结点有左孩子，有 724 个结点有右孩子；该树的高度为 11 。(性质3、性质4以及完全二叉树的特征)

- 性质5：如果对一棵有n个结点的完全二叉树的结点按层序编号(从第1层到第$\lfloor \log_2 n \rfloor + 1$层，每层从左到右)，则对任一结点$i$ ($1 \leq i \leq n$)，有
  (1) 如果i=1，则结点i是二叉树的根，无双亲；如果i >1，则其双亲是结点$\lfloor i/2 \rfloor$；
  (2) 如果$2i > n$，则结点i无左孩子(结点i为叶子结点)；否则其左孩子是结点$2i$；
  (3) 如果$2i+1 > n$，则结点i无右孩子；否则其右孩子是结点$2i+1$。
  思考：性质5推广到k叉树，结果会如何？

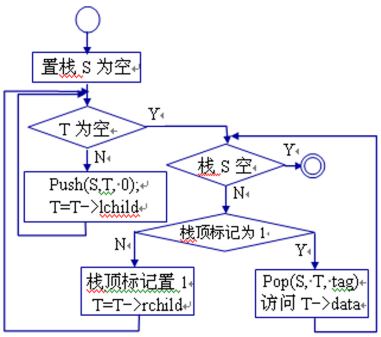
先序遍历非递归算法：



中序遍历非递归算法：



后序遍历非递归算法：



算法形式：

```
InitStack(S)
while(T != null || !StackEmpty(S)){
    while(T != null){
        Push(S, T)
        T = T->lchild
    }
    if(!StackEmpty(S)){
        ...
    }
}
```

中序线索二叉树：

- 找后继

某结点的后继：
a) 若该结点有右孩子，其后继为其右子树中最左下的结点；
b) 若该结点无右孩子，其后继由rchild指向。
其后继为满足以下条件的最小子树的根r：该结点为r的左子树中最右下的结点。

后序线索二叉树：

- 找前驱

如何在后序线索二叉树上找结点的前驱？
a) 若该结点无左孩子，则其前驱由lchild指向。
b) 若该结点有左孩子且有右孩子，则其前驱为其右孩子；
c) 若该结点有左孩子且无右孩子，则其前驱为其左孩子。

- 找后继

a) 若该结点是二叉树的根，则其后继为空；
b) 若该结点是其双亲的右孩子或是其双亲的左孩子且其双亲没有右子树，则其后继为其双亲；
c) 若该结点是其双亲的左孩子且其双亲有右子树，则其后继为其双亲的右子树中的后序遍历列出的第一个结点。

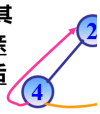
先序线索二叉树：

- 找后继

如何在先序线索二叉树上找结点的后继？
a) 若该结点无右孩子，则其后继由rchild指向。
b) 若该结点有右孩子且有左孩子，则其后继为其左孩子；
c) 若该结点有右孩子且无左孩子，则其后继为其右孩子。

- 找前驱

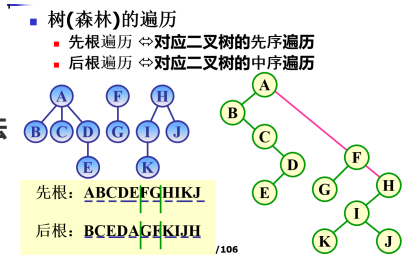
a) 若该结点是二叉树的根，则其前驱为空；
b) 若该结点是其双亲的左孩子或是其双亲的右孩子且其双亲没有左子树，则其前驱为其双亲；
c) 若该结点是其双亲的右孩子且其双亲有左子树，则其前驱为其双亲的左子树中的先序遍历列出的最后一个结点。



二叉树的层次遍历算法：利用队列，可以用来判断是不是完全二叉树

由完全二叉树的定义，对完全二叉树按层次遍历应满足：
1）若某结点没有左孩子，则它一定无右孩子；
2）若某结点缺孩子，则其后继必无孩子。
利用层次遍历，需要附加一个标志量bFlag反映是否已扫描过的结点均有左右孩子。在第一次遇到缺孩子的结点时，可将bFlag置为FALSE；此时存在以下两种情况：
·若它满足1），需要继续扫描后继结点，以判断是否满足2）；
·若不满足1），则说明不是完全二叉树。

森林的孩子兄弟表示法



赫夫曼编码算法

赫夫曼算法

1) 根据给定的n个权值构成n棵二叉树的集合F，其中每棵二叉树中只有一个带权值的结点；
2) 在F中选取两棵根结点的权值最小的树作为左右子树构造一棵新的二叉树，且置新的二叉树的根结点的权值为其左、右子树上根结点的权值之和；
3) 在F中删除这两棵树，同时将新得到的二叉树加入到F中；
4) 重复2)和3)，直到F中只含一棵树为止。

86/106

图

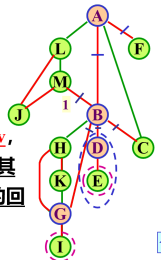
判断关节点

先找到DFS生成树，在树中找符合以下条件的结点：

关节点的特性

若生成树的根有两棵或两棵以上的子树，则此根顶点必是关节点。

若生成树中某个非叶子顶点v，其某棵子树的根和子树中的其它顶点均没有指向v的祖先的回边，则v为关节点。



84/106

查找

n个数据的顺序查找：

一次不成功，比较长度是n+1

Date

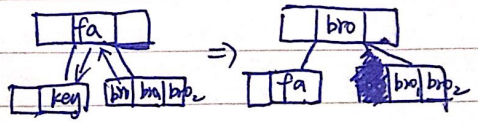
B-树删除

① 被删key所在结点，关键字数 $\geq \lceil \frac{m}{2} \rceil$，直接删

② 关键字数 $= \lceil \frac{m}{2} \rceil - 1$：

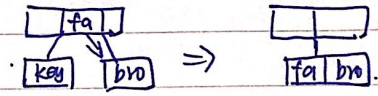
1) 被删key所在结点的左/右兄弟结点，关键字数 $> \lceil \frac{m}{2} \rceil$.

向兄弟借



2) 关键字数 $= \lceil \frac{m}{2} \rceil - 1$

向父亲借



③ 根被删了：找右子树中min值替换，转成上3种.

eg.