总结：DS:逻辑结构,存储结构,运算算法

- **线性**
  - 一般线性表
    - 顺序表 --顺序存储结构
    - 链表 --链式存储结构
  - 特殊线性表
    - 栈:插、删在一端,先进后出。
    - 队:插、删分两端,先进先出。
    - 串:有限字符序列。
    - 数组:可由若干线性表构成。

    （顺序、链式）
    - 广义表:嵌套结构的表。（链式）
- **非线性**
  - 树：层次联系
    - (线索)二叉树、哈夫曼树
    - (平衡)二叉排序树、特殊

    （链式、顺序）
  - 图：任意联系
    - 无、有向图(网络)
    - (强)连通图、完全图

    （邻接表、矩阵）

运算算法：　时间复杂度分析

插、删、遍历、查询、排序等

各 章 要 点

- **第1章 学习要点:**
- 1. 各术语的含义;
- 2. 估算O(f(n))的方法。

- **第2章 学习要点:**
- 1. 顺序表, 链表的特点;
- 2. 单链表的操作算法。

- **第3章 学习要点:**
- 栈、队的特点
- 顺序存储:
  栈与循环队列的方法
- 栈、队常用操作算法

第7章 学习要点:

1. 二叉树的概念及性质
2. 二叉树的遍历算法及其应用
3. 线索二叉树的概念和方法。
4. 哈夫曼树的特点及编码方法

第8章 学习要点:

1. 图的概念、存储结构
2. 深度、广度优先遍历算法
3. 最短路径,拓扑排序算法思路
4. 最小生成树、关键路径方法。

第9，10章 学习要点:

1. 方法(查找/排序过程)、特点
2. 计算ASL、效率O( )、比较

例:分析时间复杂度

```
1. for (i=1;i<=n;i++)
     if  i<=n/2
        for (j=i; j<=n; j++)
           C[i][j]=A[i][j]+5;
```

基本语句

$$T(n)=\sum_{i=1}^{n/2}\sum_{j=i}^{n}1$$

$$=\sum_{i=1}^{n/2}(n-i+1)$$

$$=\frac{n(3n+2)}{8}=O(n^2)$$

- 常用算法时间复杂度：
  - O（1）
  - O（$\log_2 n$）
  - O（n）
  - O（ $n\log_2 n$）
  - O（n^2）
  - O（n^3）
  - O（2^n）

- 2. 时间复杂度最小是哪个？
- (1) $5n+4n\log_2 n$
- (2) $5n\log_2 n+7\log_2 n$
- (3) $2n^2/50-90$

?

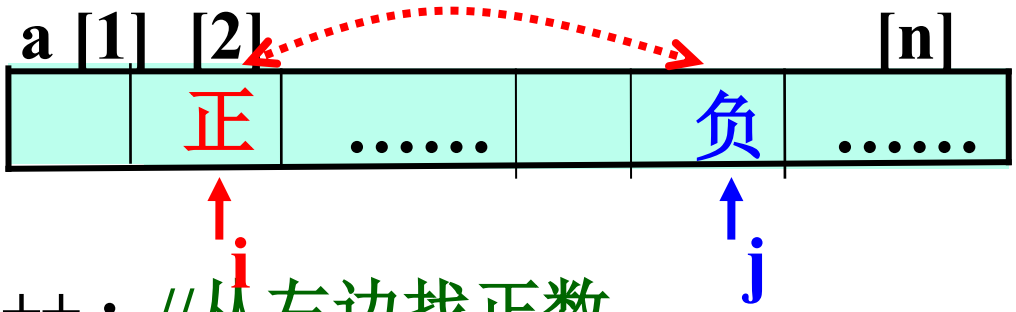
算法设计：

能否调排序算法？

1. 使含正、负整数的顺序表前端为负数、后端为正数。
   要求时间复杂度: O(n),空间复杂度: O(1) 。

[分析]:设置上、下界，分别从表左、右两端查找正、负数，不符合要求则交换，直到上、下界相遇。

```
Void divide ( int R[], int n)
{ int i=1, j=n;   // 设上,下界
    while(i<j)
 { while( (i<j )&&R[i]<0))    i++ ;  //从左边找正数
   while((i<j)&&R[j]>0))      j--;   //
 if(i<j)
  {R[0]=R[i];  R[i]=R[j]; R[j]= R[0];
    i++;    j--; } } }
```



算法设计注意：

1.选择合适结构与方法；

2. 先写方法或步骤(粗算法)再设计算法(可利用算法)；

3.注意是否有效率要求；是否需分析O(?)。

2.阅读算法指出其功能：

逆置单链表！
时间复杂度O(n)

```
void  aaa( LinkList  *&h )
// h为不带头结点的非空单链表
{ LinkList *p,*q,*r ;
   p=h;  q=p->next ;
   while ( q != NULL)
     {  ① r=q->next;
        ② q->next=p ;
        ③ p=q ;
        ④ q=r ;      }
   h->next= NULL ;  h=p;
}
```

遍历单链表！

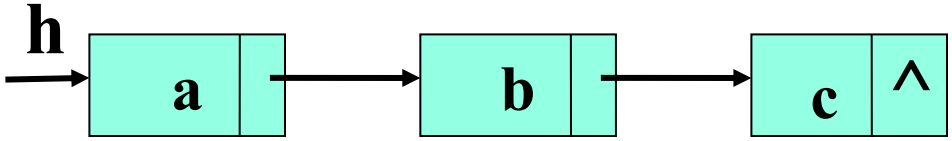
```
p=h;
while ( p )
     { cout<< p->data;
       p=p->next ; }
```

其他的方法！

时间？
空间？

1)设表头,前插法,遍历,删表头

2)读链表存入数组；
再逆向写入链表同时输出！



算法设计：

- 3. 写一算法:将两带头结点的无序单链表合并为一个有序单链表
- 方法1：
- 先连接两单链表; 再排序单链表(模拟简单选择)。
- 方法2：
- 先连接两单链表, 读到数组，排序后写回单链表。

实现复杂！

存储结构转换！

- … p=h->next; n=0 ;
- while (p)
- { n++; R[n]=p->data; p=p->next ; }
- SelectSort(R, n); p=h->next;
- for (i=1; i<=n; ++i)
  {p->data= R[i]; p=p->next ; } …

需实现！

时间复杂度:

$T(n)=n+n^2+n$

$=O(n^2)$

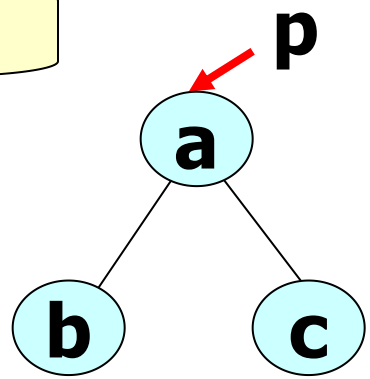
移动量大，有效率要求时不用！

4.阅读并扩展算法：

功能? 先序非递归遍历

```
typedef struct Node
 {  char   data;
    node  *lchild, *rchild;  } ;
```

判断是否满二叉数？




```
void xxx(Node *P)
// P为二叉树的根指针
{Node *s[100];  //栈
  int i= 0;  , sum=0,x=1,h=0; Depth(p,x,h);
  while ( p || i>0 )
  {  if ( p )
      { cout<<p->data;  sum++;
        s[++ i]=p;  //入栈
        p=p->lchild; }
    else { p=s[i--];  //出栈
           p=p->rchild; }
  }  if sum==(pow(2,h)-1) cout<<"yes" else cout<<"no" ;
}
```

- 方法:调求二叉树高度算法,判断$sum==2^h-1$
- 其它方法？

层次遍历,每层均满
(仅一次遍历)

5. 求二叉树b(根指针)中指定结点x(值)的层次数。

思路：先序遍历查找结点x，找的过程中求层次。

- 1) 当前根b==NULL时　　x结点的层次h=0 (找不到)
- 2) 当前根的值==x时　　h=当前层次lh
- 3) 在左子树中查找
- 4) 在右子树中查找 (左子树中无时)

初值0　初值1

```
void Level(BTNode *b, ElemType x, int &h, int lh )
{    if (b==NULL)   h=0;     //空树时返回0
  →  else  if (b->data ==x)   h=lh;     //找到时
  →  else
     {Level(b->lchild, x, h , lh+1 );    //在左子树中查找
   →  if (h==0)                  //左子树中无时在右子树中查
      Level(b->rchild, x, h, lh+1);    }
}
```

两个出口

6. 判断: 顺序存储的满二叉树是否是二叉排序树

方法:利用中序递归遍历,输出序列的前后值比较

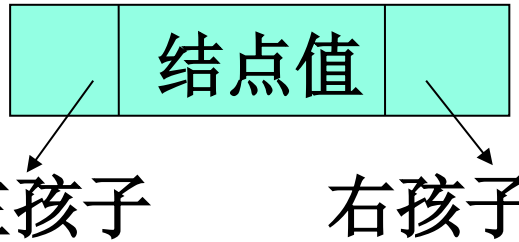
```
bool test1(int A[], int i, int n)
                        // 设A[1:n], 根i=1
{  if (i<n)
   {   test1(A, 2*i, n);          //进入左子树
       if (pre>A[i])              //当前结点比前驱值小
            return(false)         //非二叉排序树
       pre= A[i];                 //修改前驱值
       test1(A, 2*i+1, n); } }    //进入右子树
```

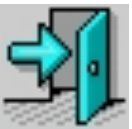
设全局变量:
前驱值pre= -maxint

用函数名返回，避免加参数!

```
算法: bool test(Node *T)
{if (T) { test(T->lchild);
     if   pre>T->data   return(false);
          pre= T->data ;
          test(T->rchild); }}
```

Node型:
lchild data rchild





7. 设计算法:查图G中是否有结点k, 并求连通分量数。

1) 方法:利用图的DFS遍历

2) 设: 邻接表存储图G,
全局变量: k, b=0 ,
visited[n]数组。

利用非连通图的遍历:

```
int  DFS1(ALGraph *g)
{ int i , sum=0;
   for (i=0; i<g->n; i++)
     if (visited[i]==0)
       { DFS(g,i); sum++; }
   return sum;}
```

3)算法:

```
void DFS(ALGraph *G, int v  )
 { ArcNode *p; visited[v]=1;  //置已访问标记
  cout<<v;    if  v==k { cout<<"ok!"; b=1};
   p=G->adjlist[v].firstarc;    //p指向v的第一条弧头结点
    while (p!=NULL)
     {  if (visited[p->adjvex]==0)   DFS(G, p->adjvex);
                                //若p所指顶点未访问,递归访问它
        p=p->nextarc;   //p指向v的下一条弧头结点} }
```

课堂练习：

同: 均为线性结构(均有顺序、链式存储)
异: 栈实现后进先出,队实现先进先出

一、简答：

- 1. 比较栈和队列的异同点

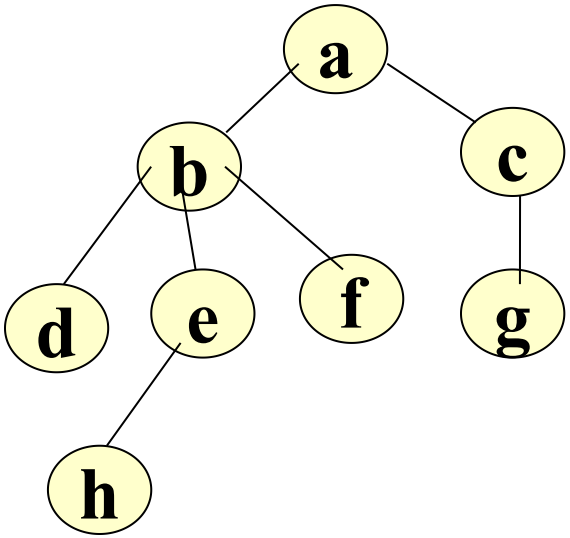
零元素：不分配存储空间；
对称的非零元素：共享存储空间

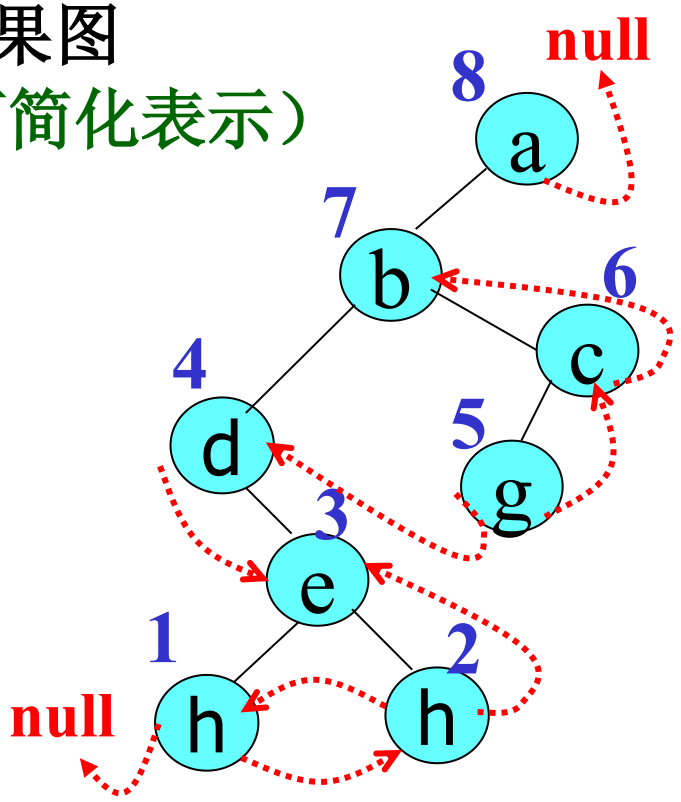
- 2. 特殊矩阵的压缩存储策略是什么？ “xyzt”共有多少子串？

共有11个子串

二、分析、证明与设计

- 1. 如图：1)画出树转二叉树结果图
  2)加上后序线索（可简化表示）





课堂练习：

2. 证明:对任一满二叉树T中的分支数B满足：$B=2(n_0-1)$

- 证明:
- 由二叉树性质①:

  $n_0=n_2+1$　即：$n_2=n_0-1$
- ∵满二叉树无度为1的结点

  ∴边数:

  $B=2\times n_2+1\times n_1+0\times n_0$

  $=2\times n_2+1\times 0+0\times n_0$

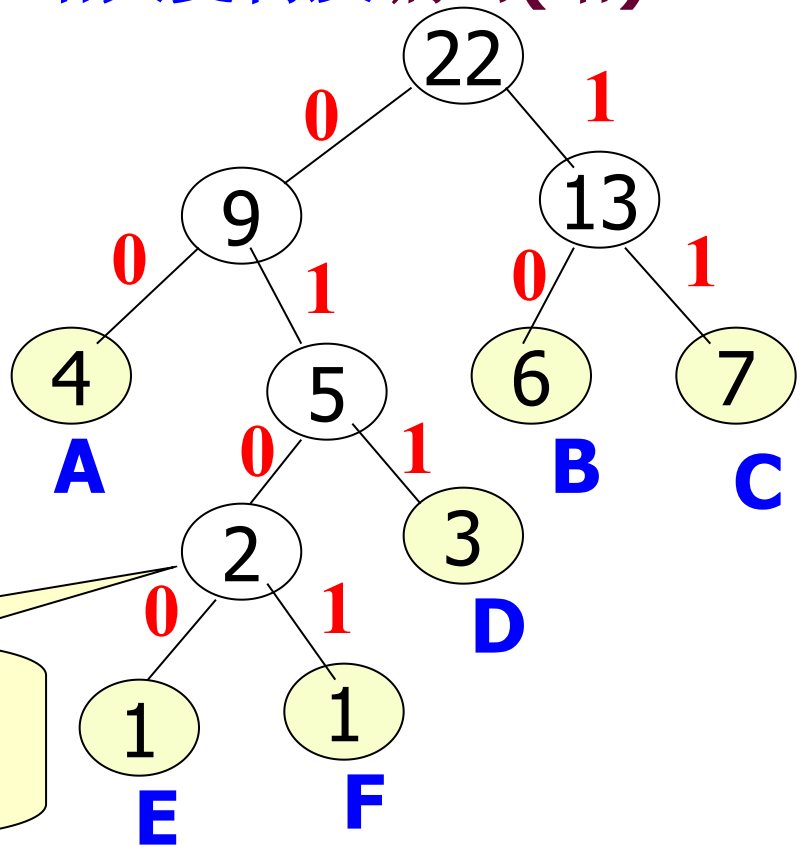
  $=2n_2$

  $=2(n_0-1)$
- 得证。

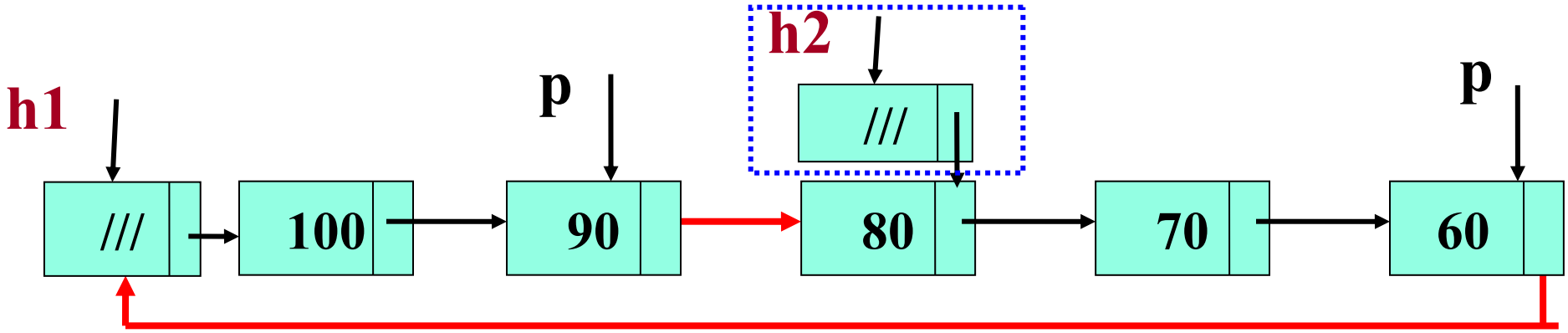
n个叶结点的哈夫曼树有2n-1个结点！

3. 设电文:AAAABBBBCCCCDDDBBCCCEF,设计哈夫曼编码,并给出图示。解:

- c={A, B, C, D, E, F}
- w={4, 6 , 7, 3, 1, 1}
- 哈夫曼树及编码(略)



三、算法设计　1、合并两个带表头结点的单链表，求合并后单链表的结点个数，并将其变为循环单链表。




```
int com-list(LinkList *&h1, LinkList *h2 )
{   LinkList *p=h1 ;  int n=0;
     while (p->next != NULL)
                 {   p=p->next ; n=n+1; }    }
     p->next=h2->next;  free(h2) ;
     while (p->next != NULL)
                 { p=p->next ; n=n+1;   }
     p->next=h1;   return(n) ;
}
```

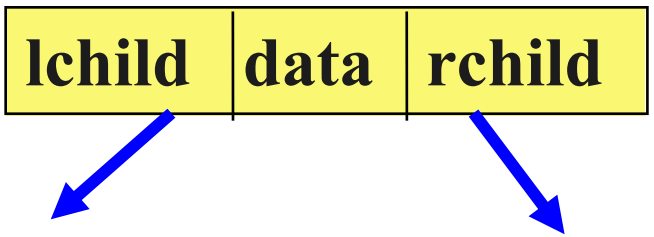
空表时适用！

三、算法设计

- 2.写出将二叉树T所有结点的右孩子data域置零并求单分支结点数的算法：
- void set-sum(BTNode *&T , int &sum)

```
                                  // T为根指针, sum=0
{   if  (T)
   { if  (T->rchild)  T->rchild=0 ;
     if  (T->lchild  && (！ T->rchild) OR
          (！ T->lchild)  && T->rchild )
     sum++;
     set-sum (T->lchild,  sum );
     set-sum (T->rchild,  sum );  }
}
```

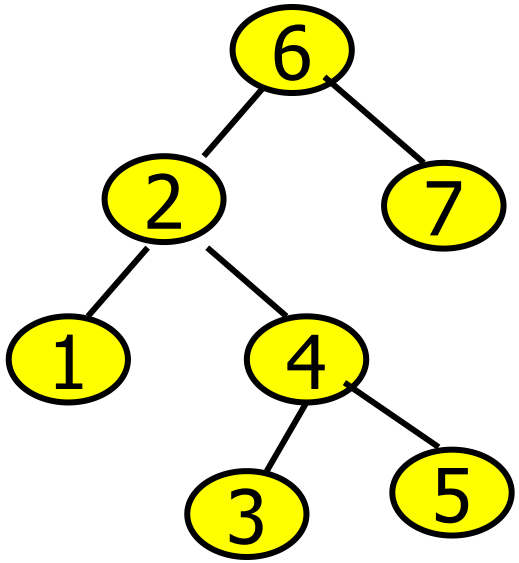
先序遍历的扩展！



掌握方法：

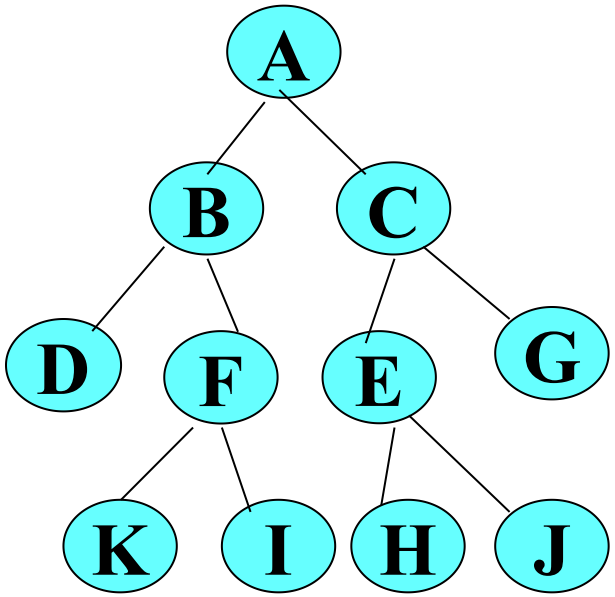
问题： 由二叉排序树的后序序列：

1 3 5 4 2 7 6 构造一颗二叉排序树？



- 填空并画出该二叉树。
- 先序：_B_F_ICEH_G
- 中序：D_KFIA_EJC_
- 后序：_K_FBHJ_G_A

- 【解答】
- 先序：A B D F K I C E H J G
- 中序：D B K F I A H E J C G
- 后序：D K I F B H J E G C A



掌握方法：

连通无向网

- 构造最小生成树：

  - prim法：选n-1条权值之和最小的边对应的点
  - kruskal法：选n-1条权值之和最小且不形成回路的边

各边权值之和最小的生成树

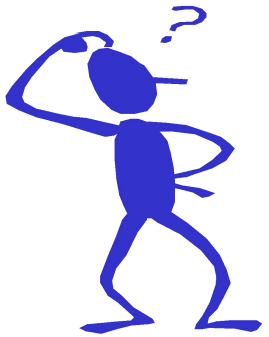
有向网

最短路径：从源点到终点边的权值之和最小的路径

- 求单源最短路径-- Dijkstra算法
- 求多源最短路径-- Flyd算法

AOE网(有向无回路网)

- 关键路径：AOE网上的从源点到汇点的最长路径。



掌握方法：

- 查找例：给定key序列：11, 78, 10, 1, 3, 2, 4, 21
- 试分别用顺序、二分、二叉排序树、平衡二叉排序树、散列(线性探测法和拉链法)查找方法，画出其对应结构图(顺序表，判定树，二叉排序树、平衡二叉排序树及两种散列表, 设H(k)=k mod 11）。
- 求等概率下每种查找成功时的ASL 。

1) 顺序查找的顺序表 ：

0	1	2	3	4	5	6	7	8	9	10
11	78	10	1	3	2	4	21			

顺序查找：

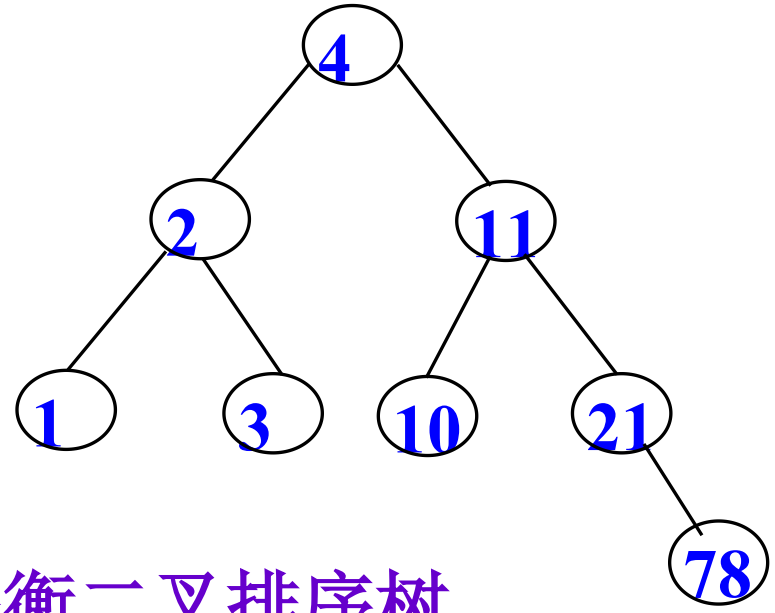
ASL=(1+2+3+4+5+6+7+8)/8=4.5

关键字序列：11，78，10，1，3，2，4，21

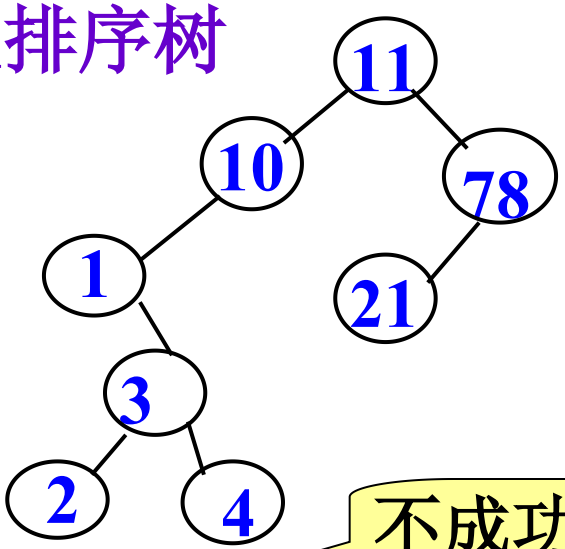
2)二分查找的判定树：

ASL=(1+2*2+3*4+4)/8=2.625;

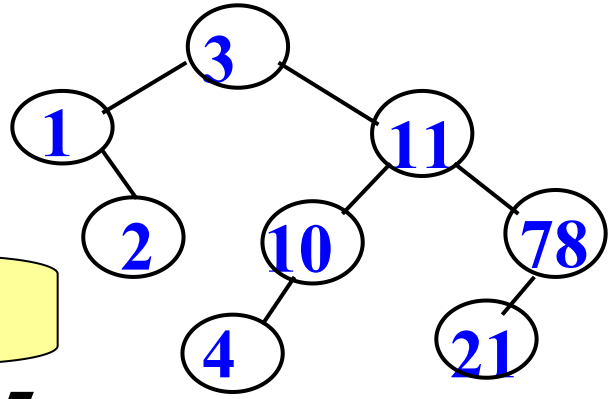
因关键字及顺序确定，故判定树、平衡及二叉排序树均唯一



3)二叉排序树



4) 平衡二叉排序树



不成功时的ASL！

3) ASL=(1+2*2+3*2+4+5*2)=3.125;

4) ASL=(1+2*2+3*3+4*2)/8=2.75;

NEXT

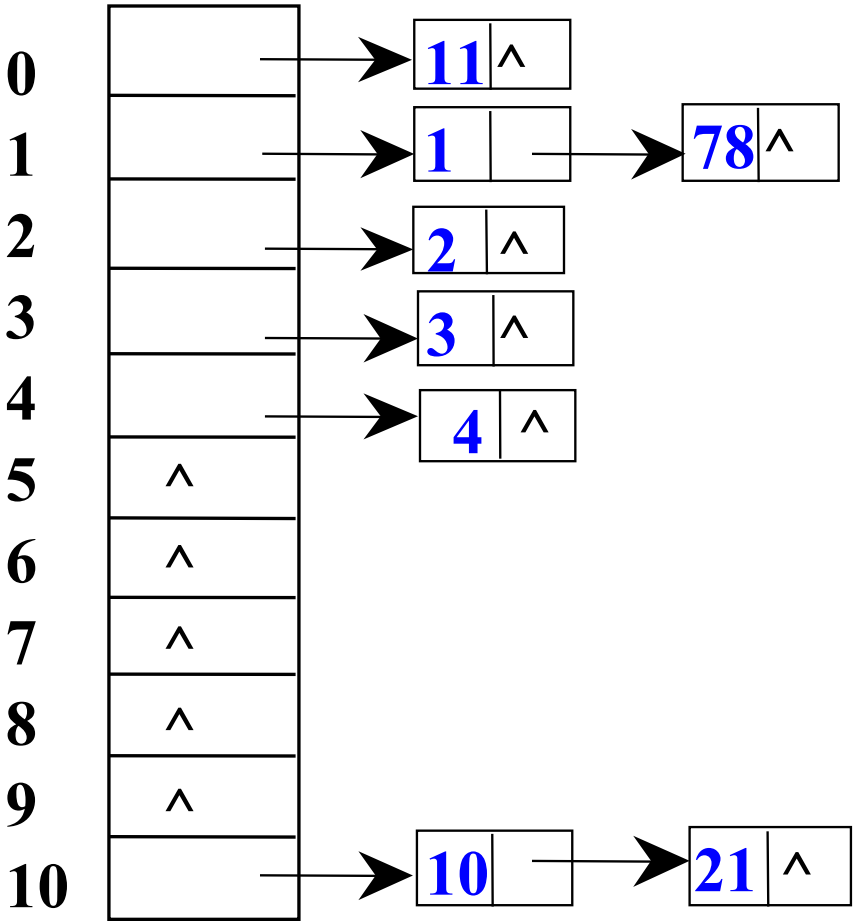
关键字序列：11,78,10,1,3,2,4,21 设:H(k)=k mod 11

5) 线性探测法的散列表：

0	1	2	3	4	5	6	7	8	9	10
11	78	1	3	2	4	21				10

ASL=（1+1+2+1+3+2+1+8 /8=2.375

6) 拉链法的散列表：



ASL=
(1*6+2*2)/8
=1.25