### 线性表

1线性表的链式存储

带头结点的线性表不可删除头结点，删除其他结点，不改变头指针

不带头结点的线性表删除头结点，需要将头指针指向下一结点，删除其他结点，不改头指针

2有序线性表

若为顺序存储结构，则删除一个元素平均移动（n-1）/2，若插入一个元素，则平均移动n/2

3线性表的链式存储和顺序存储

链式存储结构：随机存储，顺序存取。按随机存入内存，但其头指针从头访问

顺序存储结构：顺序存储，随机存取。按顺序存入内存，随机访问。

4链表的插入与删除

插入：P所指向的结点之后插入S所指向的结点

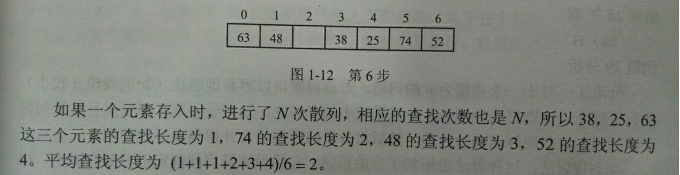
1. >prev=P S->next=P->next P->next=S P->next->prev=S

删除：删除P所指向的结点和Q所指向的结点之间的S所指向的结点

P->next=S->next 或者P->next= S

5平均查找长度

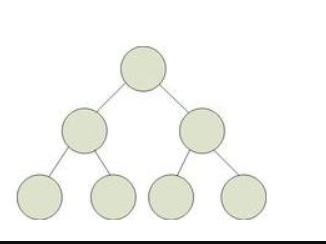
已知一个线性表（38 25 74 63 52 48）假定采用散列函数h(key)=key%7,计算散列地址，存储于A[0 1 2 3 4 5 6],线性探测方法解决冲突，则散列表的等概率成功查找的平均查找长度。



### 树

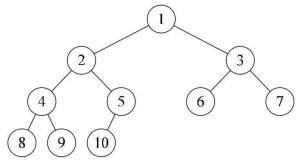
1满二叉树

：除最后一层无任何子节点外，每一层上的所有结点都有两个子结点[二叉树](https://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=111776&ss_c=ssc.citiao.link" \t "https://baike.sogou.com/_blank)。



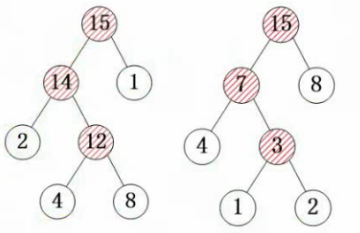
2完全二叉树

：除最后一层，为完全二叉树，最后一层左边按序存在，右边可缺失



3哈夫曼树

：最有二叉树，一般求树的带权路径长度

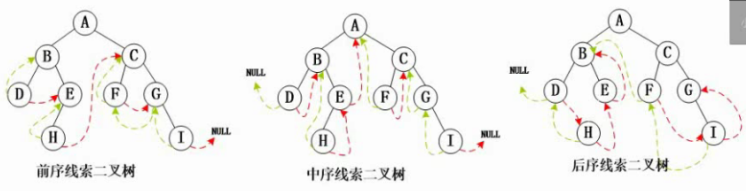


2\*2+4\*3+8\*3+1\*1=41

4\*2+1\*3+2\*3+8\*1=25

4线索二叉树

：给二叉树，根据前序中序后序画线索二叉树图



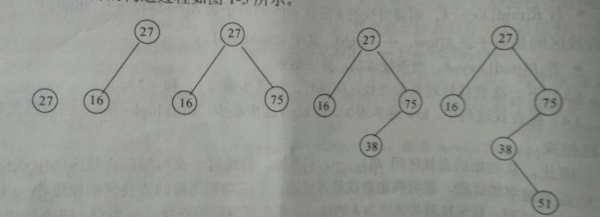
在线索二叉树中，先序的前驱，和后续的后继不能改变当前问题，即不能解决问题

而先序的后继 中序的前序 中序的后继 后续的前驱可以解决问题

5平衡二叉树

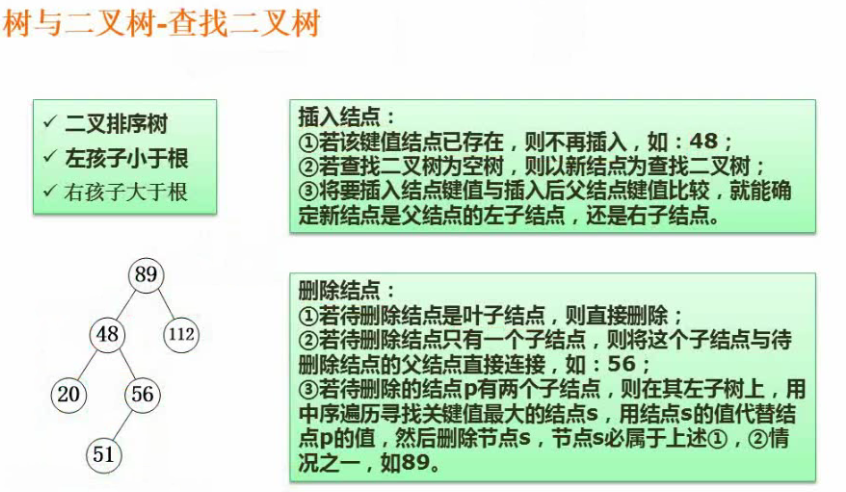
：给元素序列，来构造二叉树，找最小不平衡子树的根

例如元素序列（27 16 75 38 51 ）



最小不平衡子树的根为75

6查找二叉树



删除结点：

1叶子结点 如51直接删除

2一个子结点 如56删除后将51连接与48上

3两个子结点 如89删除后，左子树里选择最大值放于89位置，其它按1和2步骤

7转二叉树

：对每一个结点，孩子结点作为左子树，兄弟节点为右子树

### 结点计算

1叶子结点的计算

已知树的每一度的节点个数，则可以算出叶子结点 度为2的结点等等

例如一棵度为4的树，20个度为4的结点，10个度为3的结点，1个度为2的结点，10个度为1的结点。

叶子结点算法 （20\*4+10\*3+1\*2+10\*1+1（根节点））---（20+10+1+10）===82

例如一棵度为3的树，100个度为3的结点，200个度为2的结点，

叶子结点算法 （3\*100+2\*200+1（根节点））---（100+200）==401

例如一个具有967个节点的完全二叉树，其叶子结点个数为484

(奇数加一 偶数不变最后除以二即可)N2+N1+N0=957 N0=N2+1

### 算法

1分治算法

：对于一个规模为n的问题，若问题小则可以直接解决，否则将问题划分为若干个小的子问题，子问题互相独立且与原问题形式相同，递归解决子问题，最后解决原问题

2动态规划算法

：将问题实例分解更小的子问题，并存储子问题的解，从而避免计算重复的子问题。而且其也运用分治法

3贪心算法

：追求当前最优解，但不一定是最终最优解。不包含回溯算法。

4回溯算法

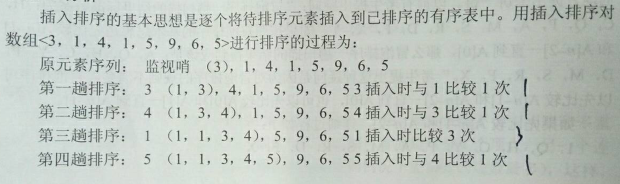
：系统搜索问题的方法，能进则进，不进则退，换路而行。递归实现

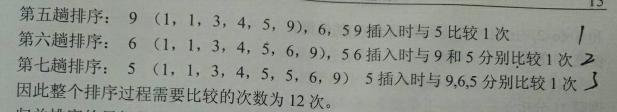
5递归算法

：分成递推和回归两个阶段

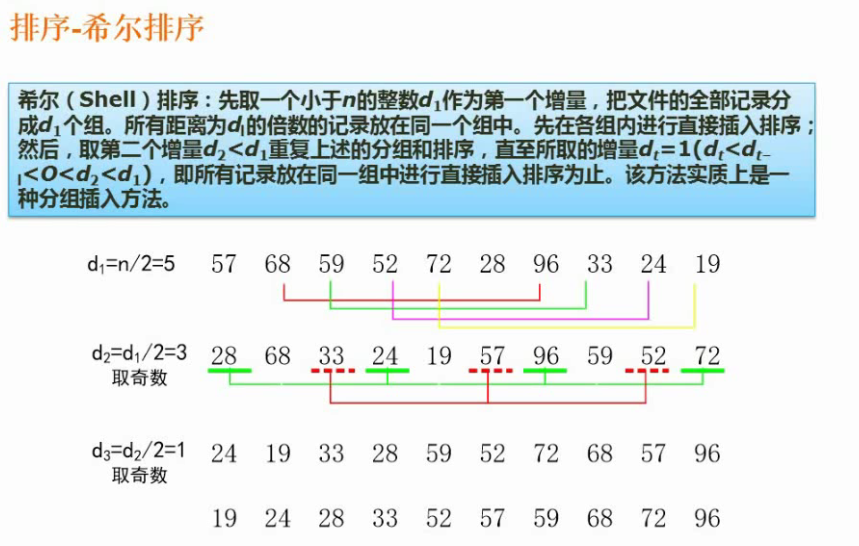
### 排序及时间复杂度

1插入排序

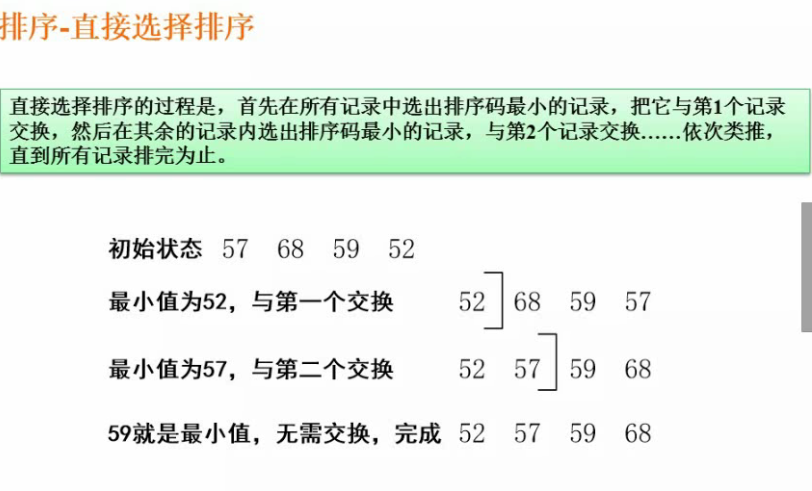




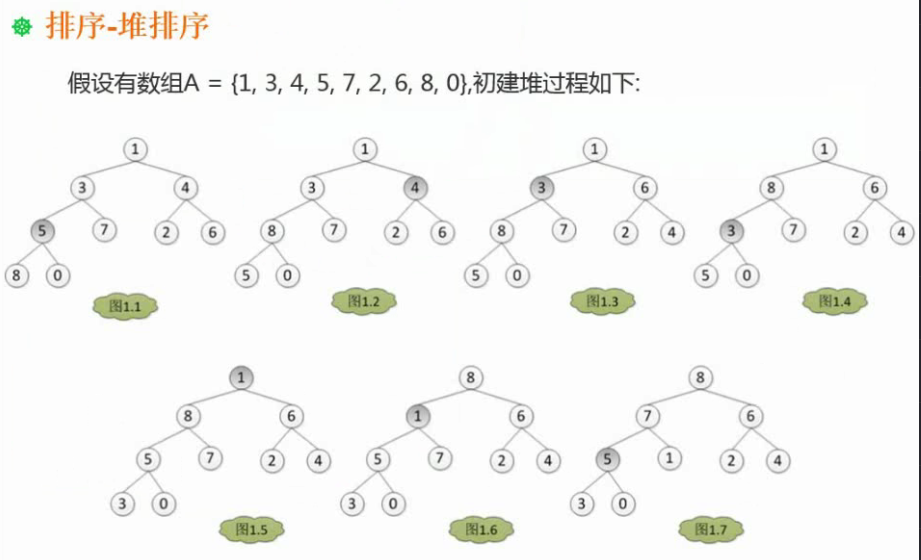
2希尔排序



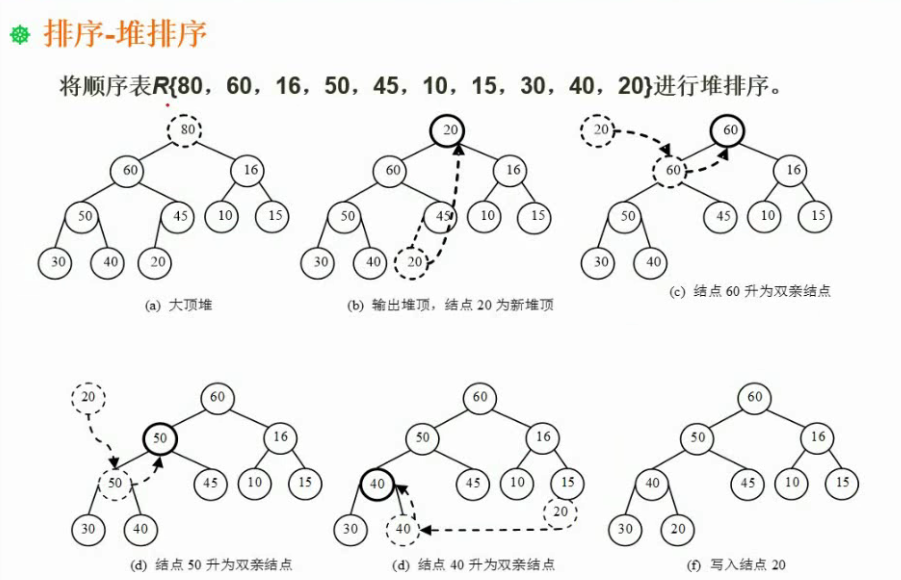
3直接选择排序



4堆排序

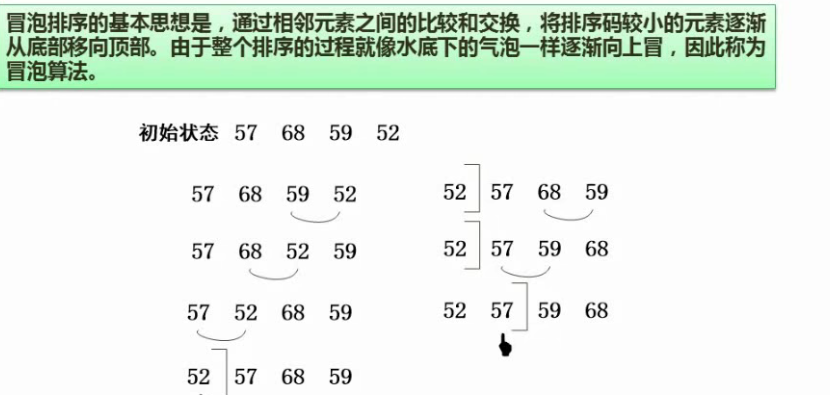


先按顺序进行排列为完全二叉树，然后从最后一个非叶子结点开始与子结点对比大小，并进行调整，完成过程中父节点改变，则非叶子结点的子结点也将再次改变。



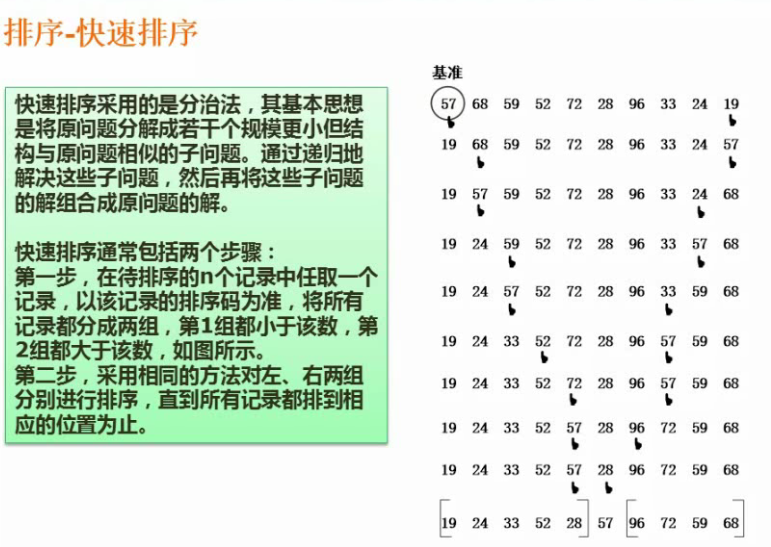
完成初步堆排序之后，取走根节点，将最后以为结点放于根节点位置，并与子结点对比排位，调整位置一直到排序完成。然后再次取走根节点，循环以上操作即为堆排序。

5冒泡排序



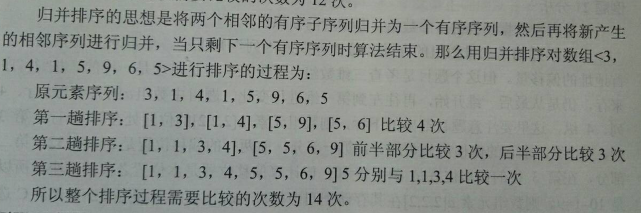
从最后一位开始向前面依次对比调整，为第一轮，则确定最小的第一位（这里是52），依次循环，进行排序。

6快速排序



基准改变端，则指针加一，一直到一轮对比结束，则完成一轮排序，然后循环步骤。

7归并排序



8基数排序

根据个位 十位 百位等关键码进行分配和收集。

9时间复杂度



堆排序和快速排序和归并排序为O（log2n） 冒泡排序插入排序，选择排序为Ο(n2)

顺序查找时间复杂度 O（n） 二分（折半）查找时间复杂度O（log2n） 基数O（log2n）



