查找：

在一个键集(key set)中，查找指定键的位置：

顺序查找

平均情况时间复杂度是O(n/2)

二分查找：

必须先强调二分查找的前提：1、顺序；2、连续存储（可随即访问）；物理线性存储；

0 1 2 3 4 5 6 7 8 9

3 5 9 1 0 8 4 2 7 6

h t

m = (h + t) / 2

时间复杂度 O(logN)

**散列桶：**

对于key，进行一个运算，得到index，这个index就是存放key的空间的下标。

关键问题是选取合理的散列函数，使得不同的key，尽可能地得到不同的index；

但是，要完全完成上述要求，则，index空间开销可能会大到无法承受！因此，总是希望能尽可能的减少index空间，并使得不同key得到不同的index。

事实上，上述思想肯定会存在不可避免的“碰撞”问题（冲突问题）；解决散列桶的冲突，成为散列桶性能和开销之间平衡的关键。

通常的散列函数是模质数，MOD 质数（%质数）

1、再散列；

2、开链法；

3、顺序探测。

安全单项散列函数：不可逆加密算法

非对称加密技术简介：

两个密钥

公钥

私钥

**二叉查找树**

根结点数值大于左孩子数值，且小于右孩子数值。

根结点数值大于左子树最大数值，且小于右子树最小数值。

1、建立二叉查找树；

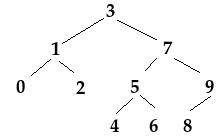
以第一个元素为根结点，比根结点大的放在右孩子处，比根结点小的放在左孩子处，以此类推。

2、查找。

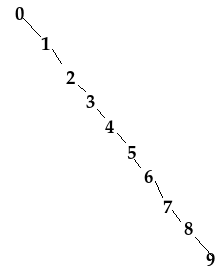
若有如下数据：

3 7 1 0 9 2 8 5 6 4

构建成二叉查找树如下：

注意：如果二叉查找树是完全顺序，则退化成为了一个链表。

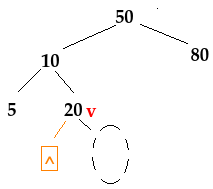
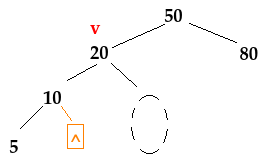
0 1 2 3 4 5 6 7 8 9

****

**AVL平衡二叉查找树**

AVL树的节点旋转

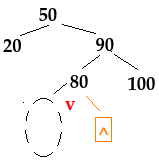
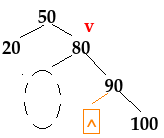
条件：~~v.right == NULL && v.left.right ==NULL~~ ~~&&~~ LL

v.left.left -> v.left

v -> v.left.right

v.left.right -> v.left

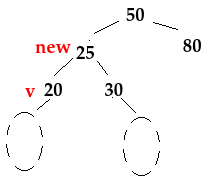
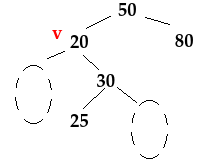
条件：~~v.left == NULL && v.right.left == NULL &&~~ RR

v.right -> v

v -> v.right.left

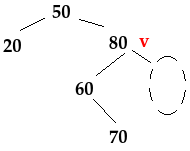
v.right.left -> v.right

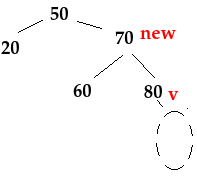
条件：~~v.left == NULL && v.right.right == NULL &&~~ RL

new -> v

v -> new.left

v.right -> new.right

条件：失衡 && LR

new -> v

v -> new.right

v.left -> new.left

