**21计科2班数据结构复习资料**

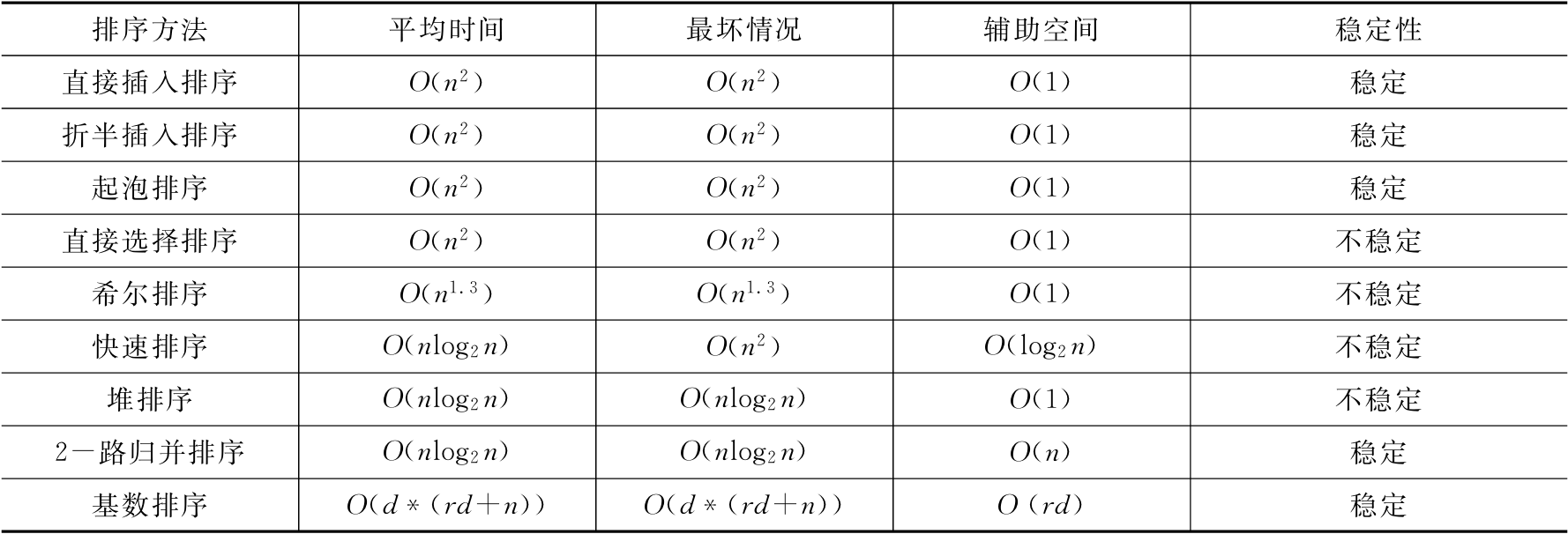
**一、算法效率分析**

时间复杂度的计算（Time Complexity）

空间复杂度（Space Complexity）

计算**语句频度**（Frequency Count）（6分）

一个语句的频度，是指该语句在算法中被重复执行的次数。算法中所有语句的频度之和是该算法所求解问题规模n的函数。一个特定算法“运行工作量”的大小，只依赖于问题的规模，或者说是问题规模的函数。



**二、平均查找长度（Average Search Length）**

在查找时，为确定元素在顺序表中的位置，需和给定值进行比较的数据元素个数的期望值称为查找算法在查找成功时的平均查找长度

**三、循环队列**

队空的条件：Q.front == Q.rear

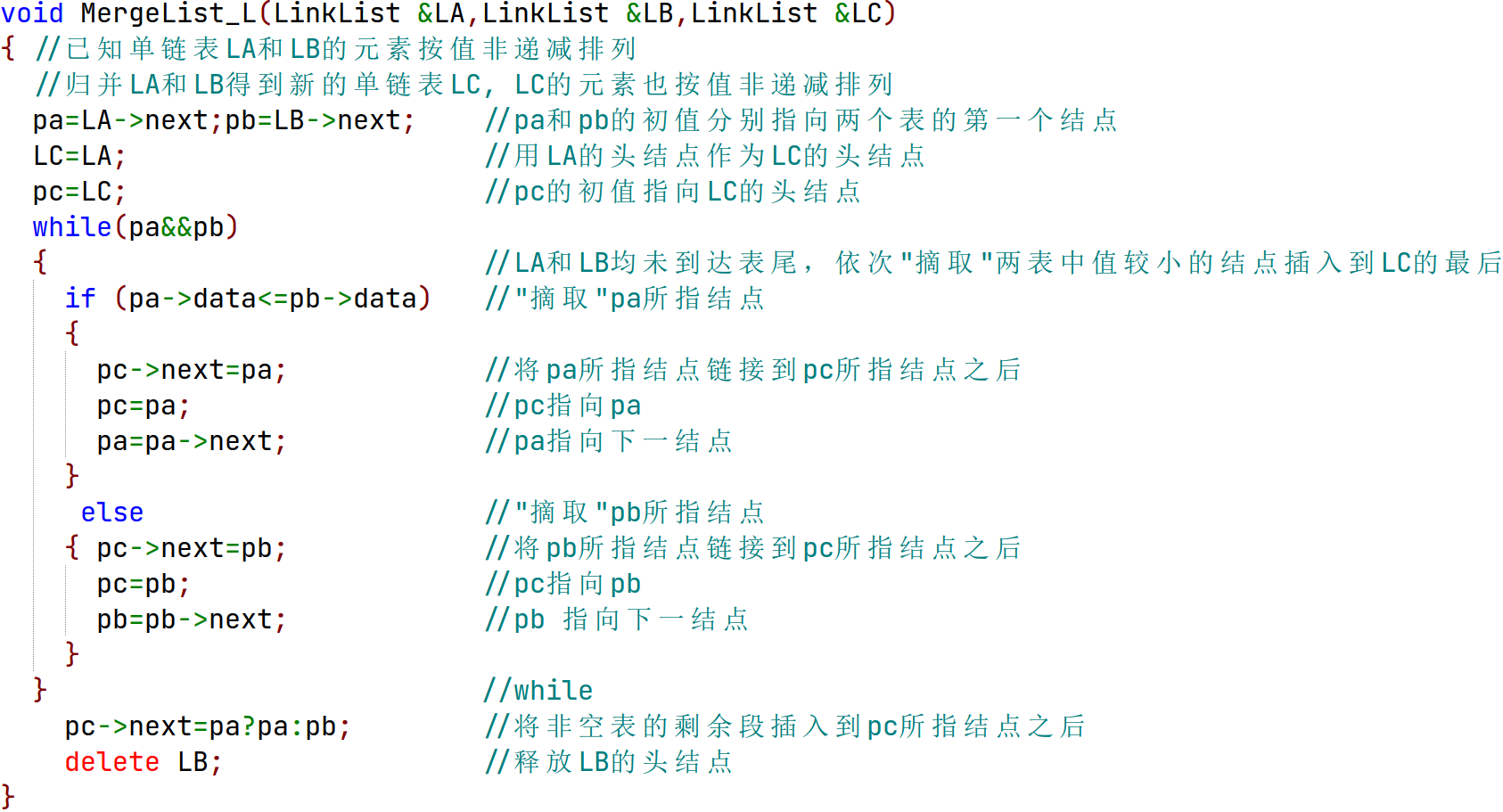
队满的条件：(Q.rear + 1)%MAXQSIZE == Q.front

队列的长度：(Q.rear-Q.front+MAXQSIZE)%MAXQSIZE

**四、有序表的归并**

用顺序表实现的有序表称为有序顺序表。有序顺序表的归并是将两个有序顺序表合并





**五、排序的稳定性**

当排序记录中的关键字()都不相同时，则任何一个记录的无序序列经排序 后得到的结果唯一；反之，当待排序的序列中存在两个或两个以上关键字相等的记录时，则排序所得的结果不唯一。

假设，也就是说其对应的关键字相同，且在排序前的序列中领先于（即 ）。若在排序后的序列中仍领先于，则称所用的排序方法是**稳定的**；反之，若可能使排序后的序列中领先于，则称所用的排序方法是**不稳定的**。注意，排序算法的稳定性是针对所有 记录而言的。也就是说，在所有的待排序记录中，只要有一组关键字的实例不满足稳定性要求，则该排序方法就是不稳定的。虽然稳定的排序方法和不稳定的排序方法排序结果不同，但不能说不稳定的排序方法就不好，各有各的适用场合。

例：排序算法的稳定性是指（A）。

A.经过排序后，能使关键字相同的元素保持原顺序中的相对位置不变

B.经过排序后，能使关键字相同的元素保持原顺序中的绝对位置不变

C.排序算法的性能与被排序元素个数关系不大

D.排序算法的性能与被排序元素的个数关系密切

**六、希尔排序（Shell’s Sort）**

【算法步骤】

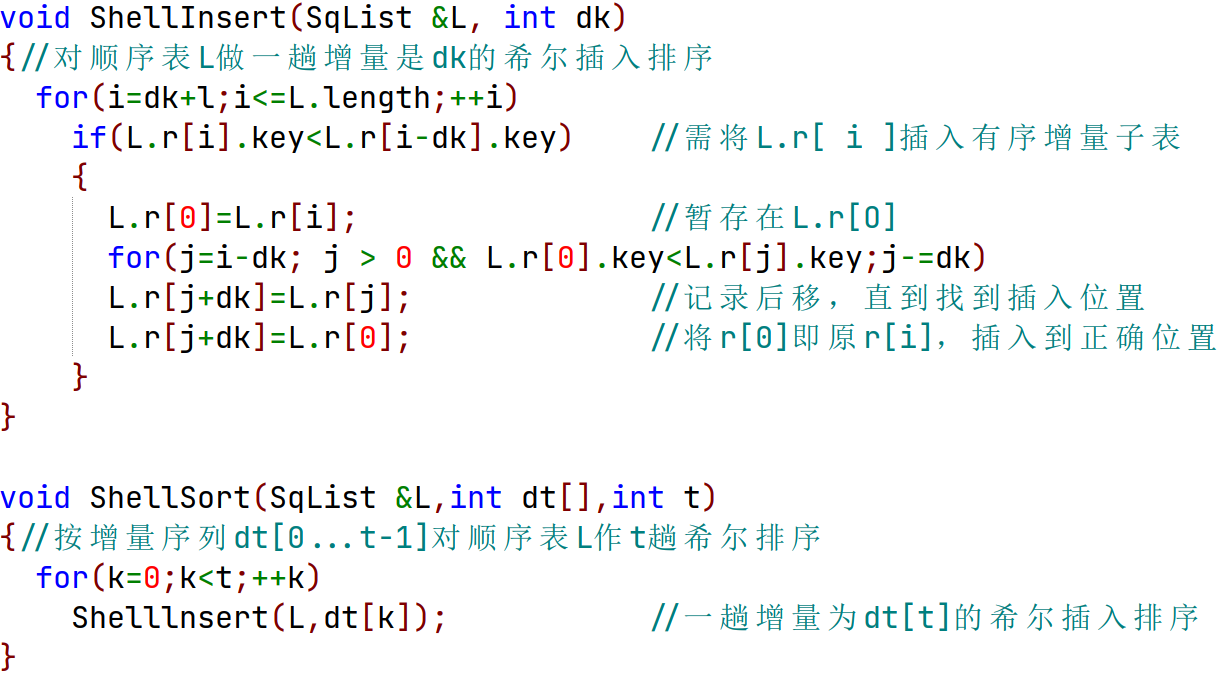
希尔排序实质上是采用分组插入的方法。先将整个待排序记录序列分割成几组，从而减少参与直接插入排序的数据量，对每组分别进行直接插入排序，然后增加每组的数据量，重新分组。这样当经过几次分组排序后，整个序列中的记录“基本有序”时，再对全体记录进行一次直接插入排序。

希尔对记录的分组，不是简单地“逐段分割”，而是将相隔某个“增量”的记录分成一组。

① 第一趟取增量() 把全部记录分成个组，所有间隔为的记录分在同一组，在各个组中进行直接插入排序。

② 第二趟取增量 ()，重复上述的分组和排序。

③ 依次类推，直到所取的增量=1 ()，所有记录在同一组中进行直接插入排序为止。



例题：给出关键字序列｛50,26,38,80,70,90,8,30,40,20｝的希尔排序过程，取增量序列为d={5,3,1}，排序结果为从小到大排列。

原始序列： 50,26,38,80,70,90,8,30,40,20

第一趟（增量 5）： 50,8,30,40,20,90,26,38,80,70

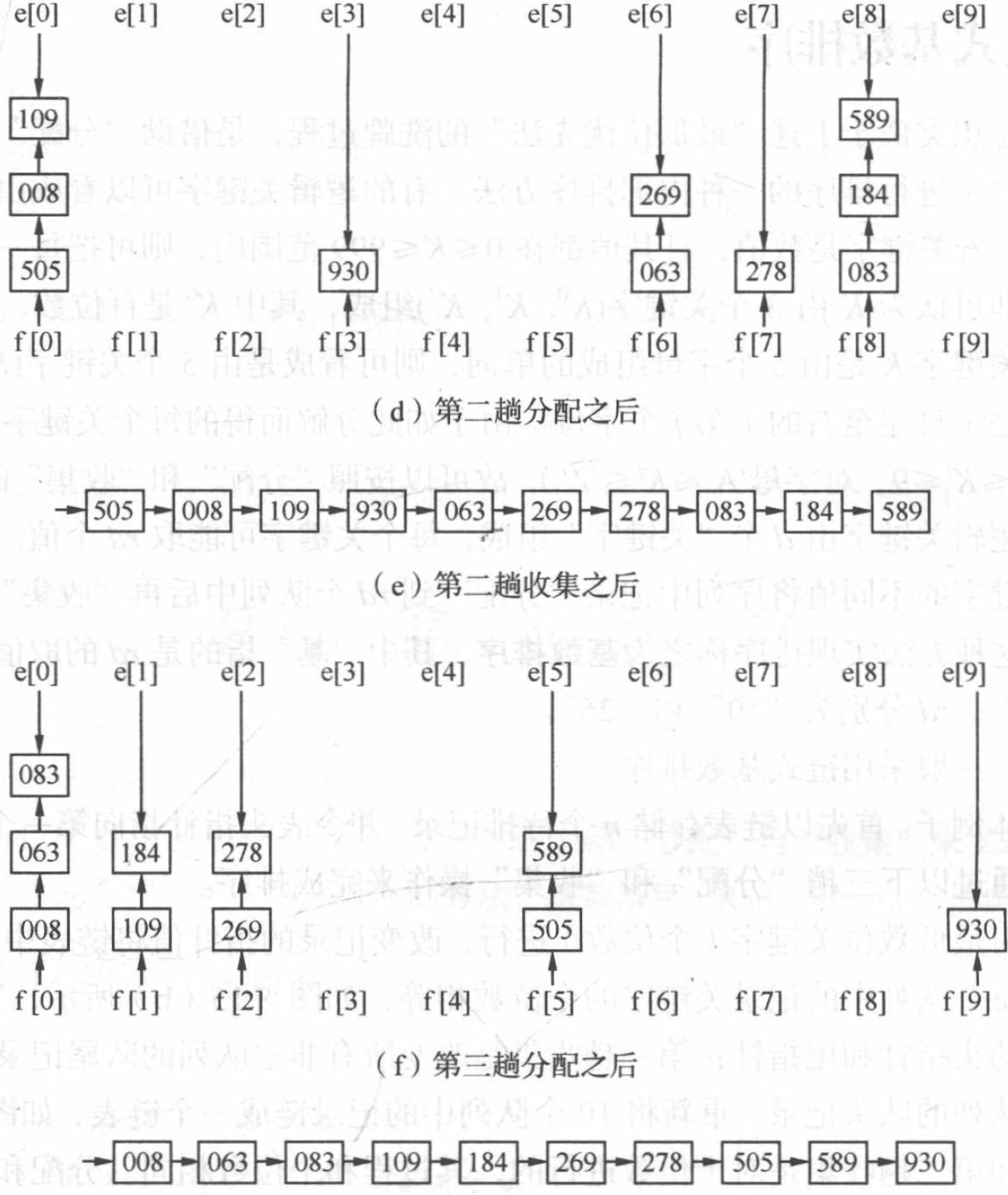
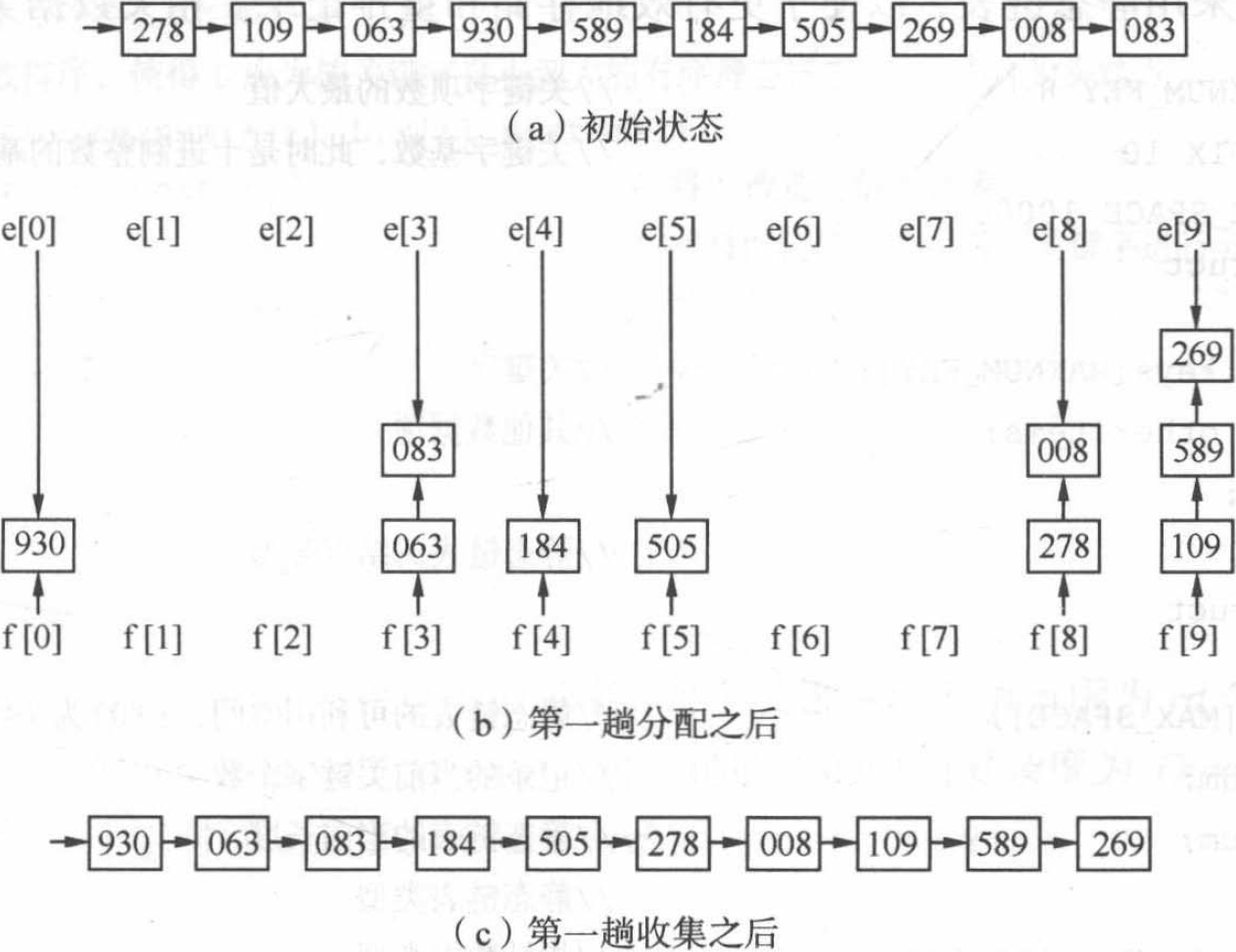
第二趟（增量 3）： 26,8,30,40,20,80,50,38,90,70

第三趟（增量 1）： 8,20,26,30,38,40,50,70,80,90

**七、基数排序**

假设记录的逻辑关键字由个“关键字”组成，每个关键字可能取个值。只要从最低数位关键字起，按关键字的不同值将序列中记录“分配”到个队列中后再“收集”之，如此重复次完成排序。按这种方法实现排序称之为**基数排序**，其中“基”指的是的取值范围。

具体实现时，一般采用**链式基数排序**。



**八、哈希表**

（1）同义词：哈希函数值相同的关键字称为同义词。

（2）冲突：对不同的关键字可能得到同一哈希地址，即，而

（3）哈希函数：在记录的存储位置和其关键字之间建立一个确定的对应关系，使，称这个对应关系为哈希函数，为哈希地址。

**除留余数法**

假设哈希表表长为，选择一个不大于的数，用去除关键字，除后所得余数为哈希地址，即

这个方法的关键是选取适当的，一般情况下，**可以选为小于表长的最大质数**。例如，表长,可取。

除留余数法计算简单，适用范围非常广，是最常用的构造散列函数的方法。它不仅可以对关键字直接取模，也可在折叠、平方取中等运算之后取模，这样能够保证散列地址一定落在哈希表的地址空间中。

例题：

1.对于线性表(7,34,55,25,64,46,20,10)进行散列存储时，若选用作为哈希函数，则哈希地址为1的元素有( D )个。

A. 1 B. 2 C. 3 D. 4

提示：哈希地址为1，即找同义词，可以分别计算哈希函数的值，值为1的元素个数即是所求的哈希地址为1的元素个数。

2.设哈希表中有个存储单元，散列函数，则最好选择( B )。

A.小于等于的最大奇数 B.小于等于的最大素数

C.小于等于的最大偶数 D.小于等于的最大合数

**处理冲突的方法（考一种）**

（1）链地址法

链地址法的基本思想是：把具有相同哈希地址的记录放在同一个单链表中，称为同义词链表。有个哈希地址就有个单链表，同时用数组存放各个链表的头指针，凡是哈希地址为的记录都以结点方式插入到以为头结点的单链表中。

（2）开放地址法

开放地址法的基本思想是：把记录都存储在散列表数组中，当某一记录关键字的初始哈希地址为发生冲突时，以为基础，采取合适方法计算得到另一个地址，如果仍然发生冲突，以为基础再求下一个地址，若仍然冲突，再求得。依次类推，直至不发生冲突为止，则以为该记录在表中的哈希地址。

这种方法在寻找“下一个”空的哈希地址时，原来的数组空间对所有的元素都是开放的，所以称为开放地址法。通常把寻找“下一个”空位的过程称为**探测**，上述方法可用如下公式表示：

其中，为哈希函数，为哈希表表长，为增量序列。根据取值的不同，可以分为以下2 种探测方法。

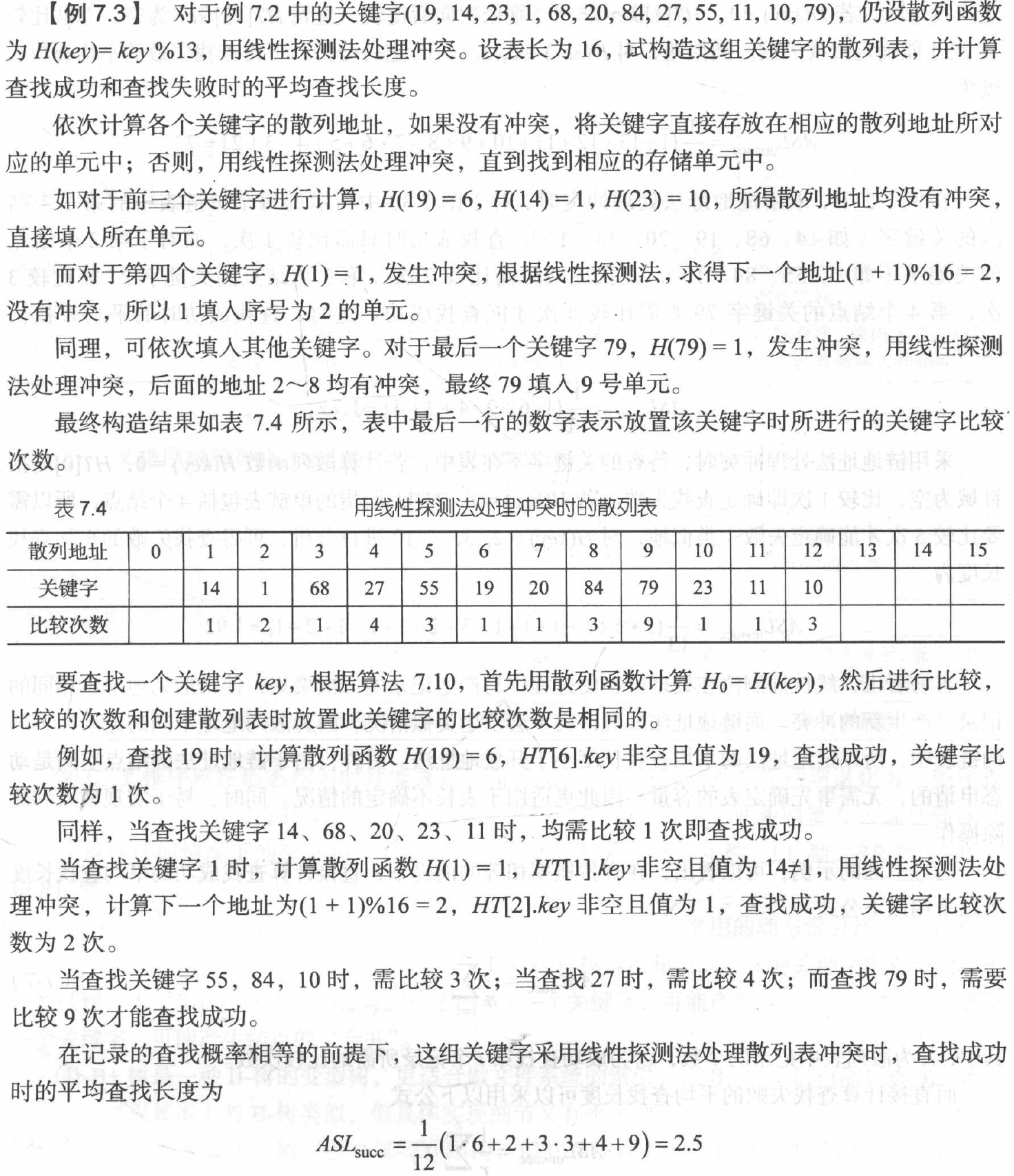
1）线性探测法。当时，称为**线性探测法**。这种方法的特点是：冲突发生时，顺序查看表中下一个单元（探测到表尾地址时，下一个探测地址是表首地址0）, 直到找出一个空闲单元（当表未填满时一定能找到一个空闲单元）或查遍全表。

线性探测法可能使第个哈希地址的同义词存入第个散列地址，这样本应存入第个散列地址的元素就争夺第个哈希地址的元素的地址，**也就是说同义词冲突的探查序列和非同义词之间不同的探查序列交织在一起**，从而造成大量元素在相邻的哈希地址上“聚集”（或堆积）起来，大大降低了查找效率。

2）平方探测法。当时，称为**平方探测法**，其中，哈希表长度必须是一个可以表示成的素数，又称**二次探测法**。

平方探测法是一种处理冲突的较好方法，可以避免出现“堆积”问题，它的缺点是不能探测到哈希表上的所有单元，但至少能探测到一半单元。

**哈希表查找性能计算——线性探测法**



**装填因子**

例题：

1.已知哈希表地址空间为,哈希函数为,采用线性探测再散列处理冲突。若依次将数据序列：76,45,88,21,94,77,17存入该散列表中，则元素17存储的下标为（ C ）；在等概率情况下查找成功的平均查找长度为（ H ）。

A. 0 B.1 C. 2 D. 3

E. 4 F. 5 G. 6 H. 7

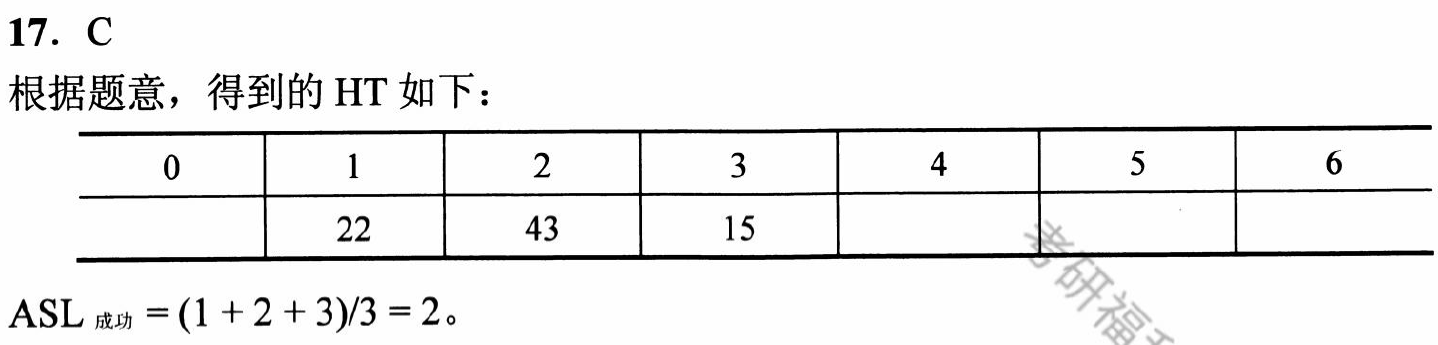
提示：注意处理冲突时是模去表长而不是模去哈希函数的值，所以本题是

2. 假定有个关键字互为同义词，若用线性探测法把这个关键字存入散列表中，至少要进行( D )次检测。

A. B. C. D.

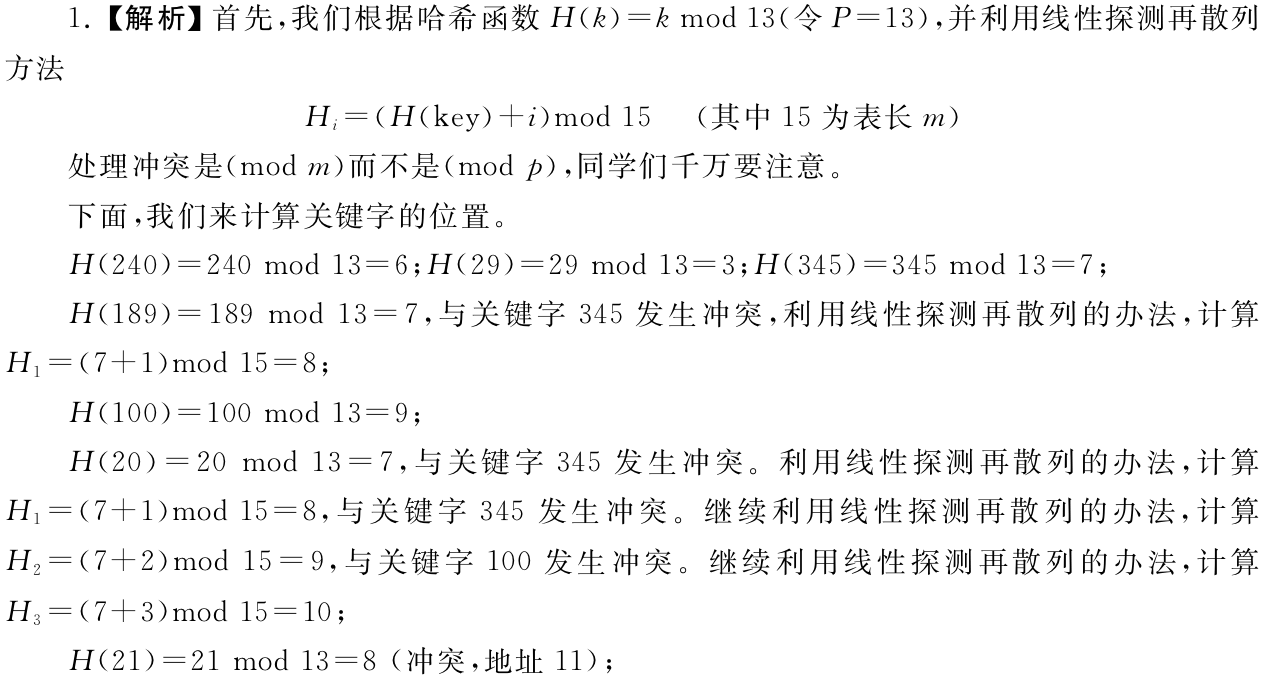
3. 现有长度为、初始为空的散列表，散列函数，用线性探测再散列法解决冲突。将关键字依次插入后，查找成功的平均查找长度是( C )。

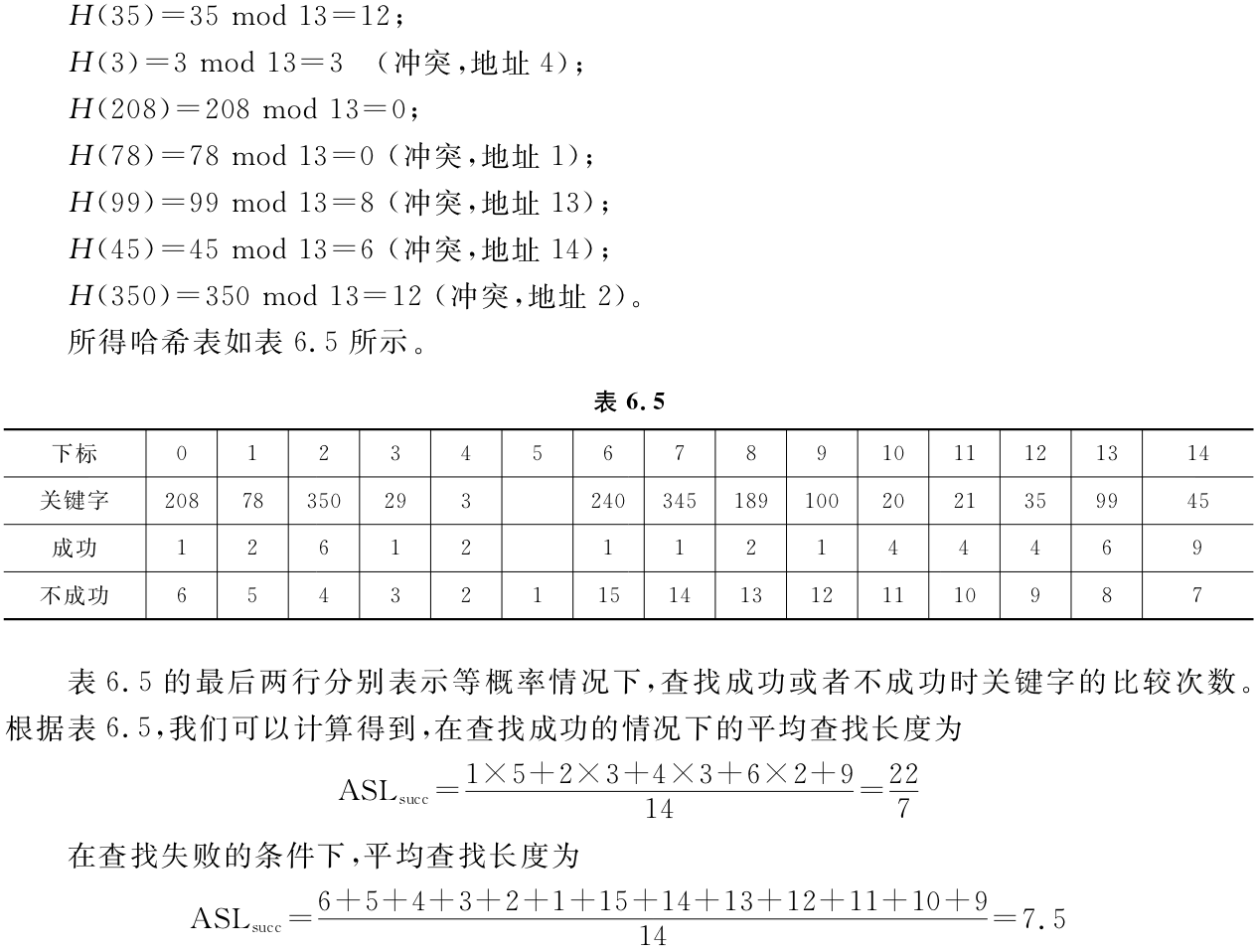
A. 1.5 B. 1.6 C. 2 D. 3



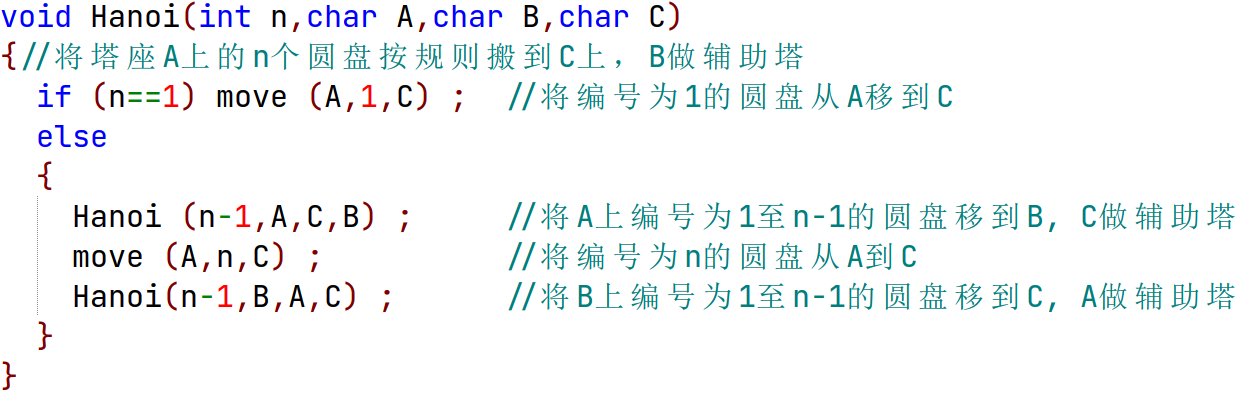
4. 已知哈希表地址空间为,哈希函数为，采用线性探测法处理冲突。将下面各数依次存入该哈希表中，并求出在等概率下，查找成功和查找失败的平均查找长度。

240, 29, 345, 189, 100, 20, 21, 35, 3, 208, 78, 99, 45, 350





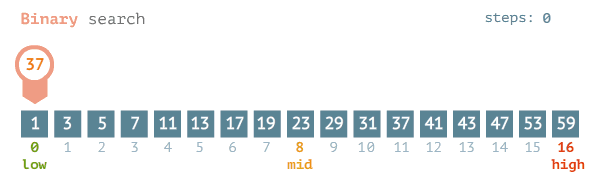
**九、汉诺塔函数（可能考）**



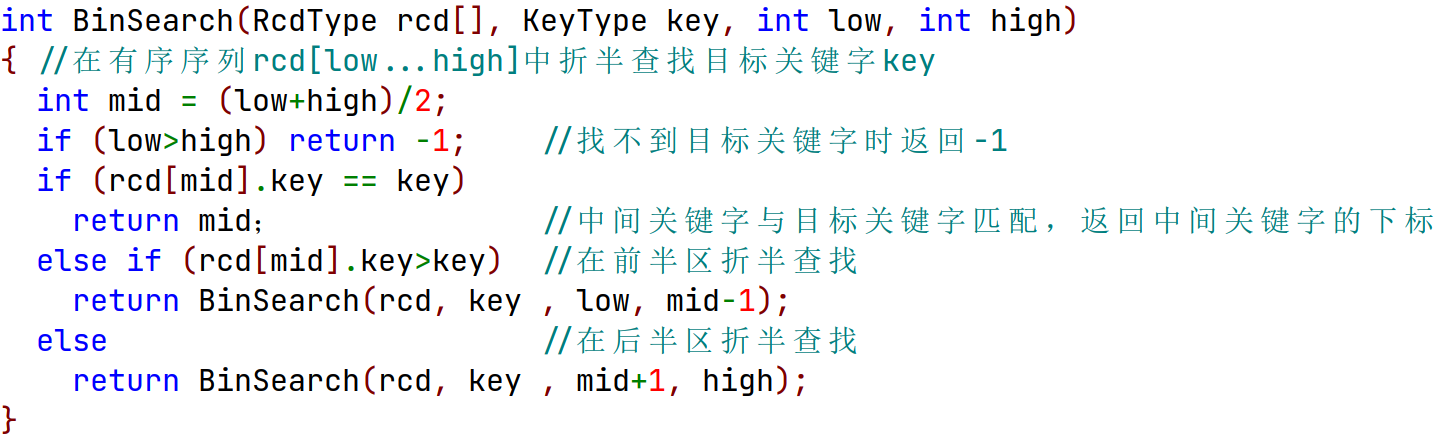
**十、折半查找( Binary Search )**

折半查找也称二分查找，它是一种效率较高的查找方法。**但是，折半查找要求线性表必须采用顺序存储结构，而且表中元素按关键字有序排列。**在下面及后续的讨论中，均假设有序表是递增有序的。

折半查找的查找过程为：从表的中间记录开始，如果给定值和中间记录的关键字相等，则查找成功；如果给定值大于或者小于中间记录的关键字，则在表中大于或小于中间记录的那一半中查找，这样重复操作，直到查找成功，或者在某一步中查找区间为空，则代表查找失败。



**递归实现：**



例题：

1.采用折半查找法查找长度为的线性表时，每个元素的平均查找长度为( D )。

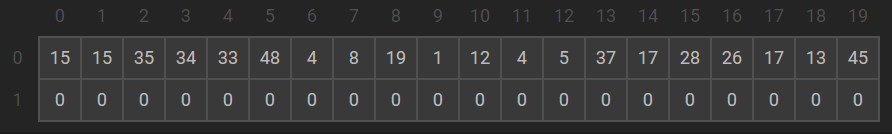
A. B. C. D.

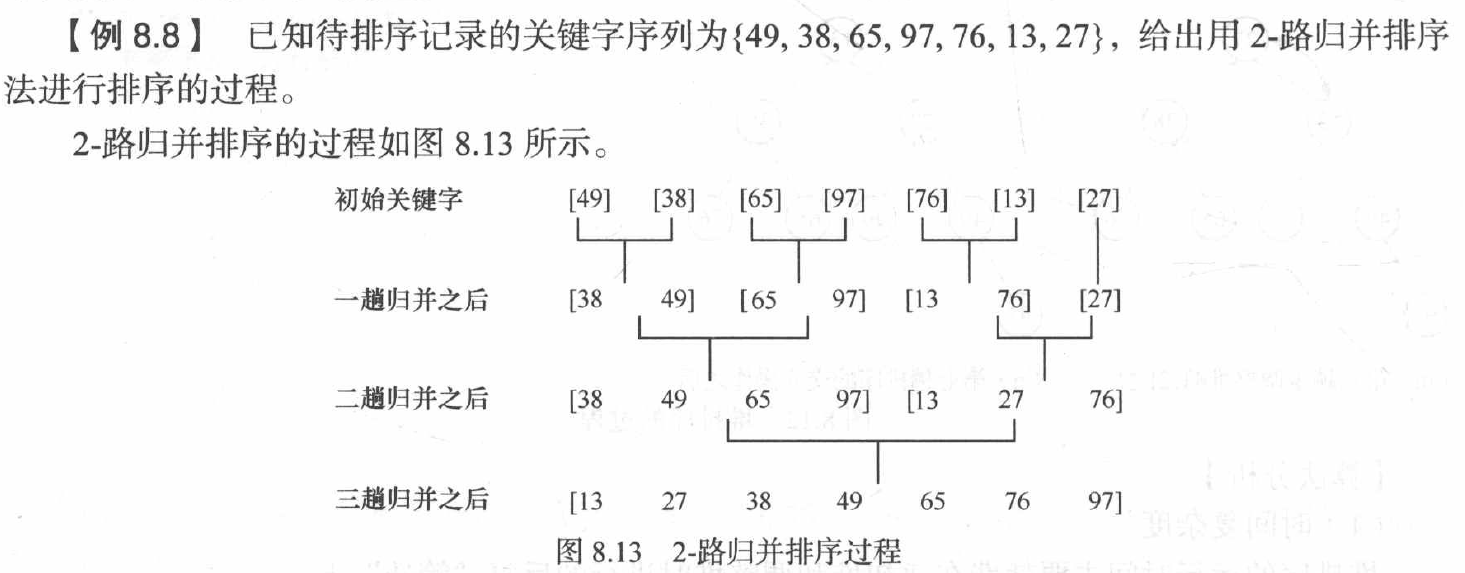
**十一、2-路归并**

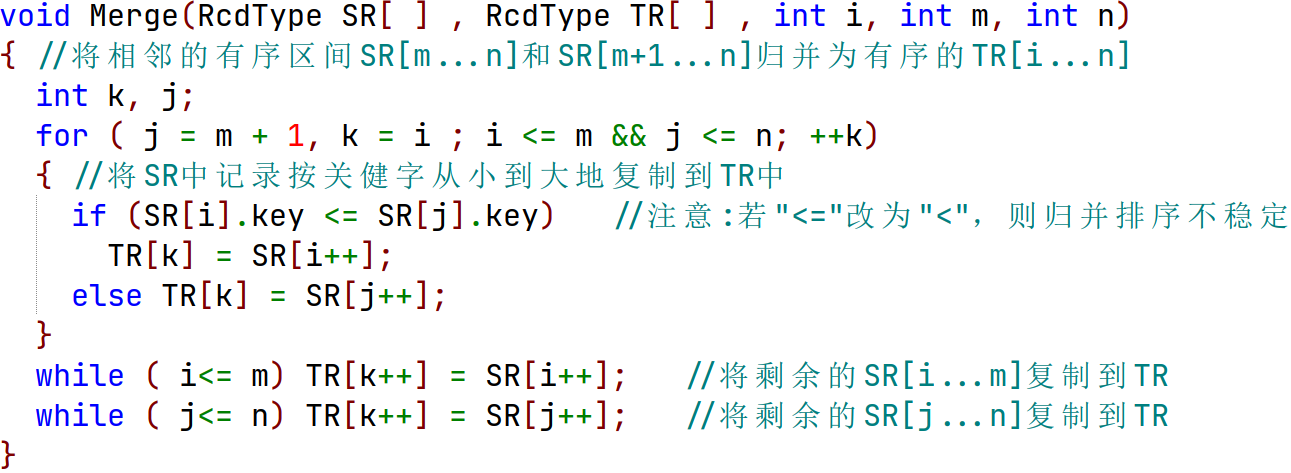
归并排序(Merging Sort)就是将两个或两个以上的有序表合并成一个有序表的过程。将两个有序表合并成一个有序表的过程称为**2-路归并**，2-路归并最为简单和常用。下面以2-路归并为例，介绍归并排序算法。

归并排序算法的思想是：

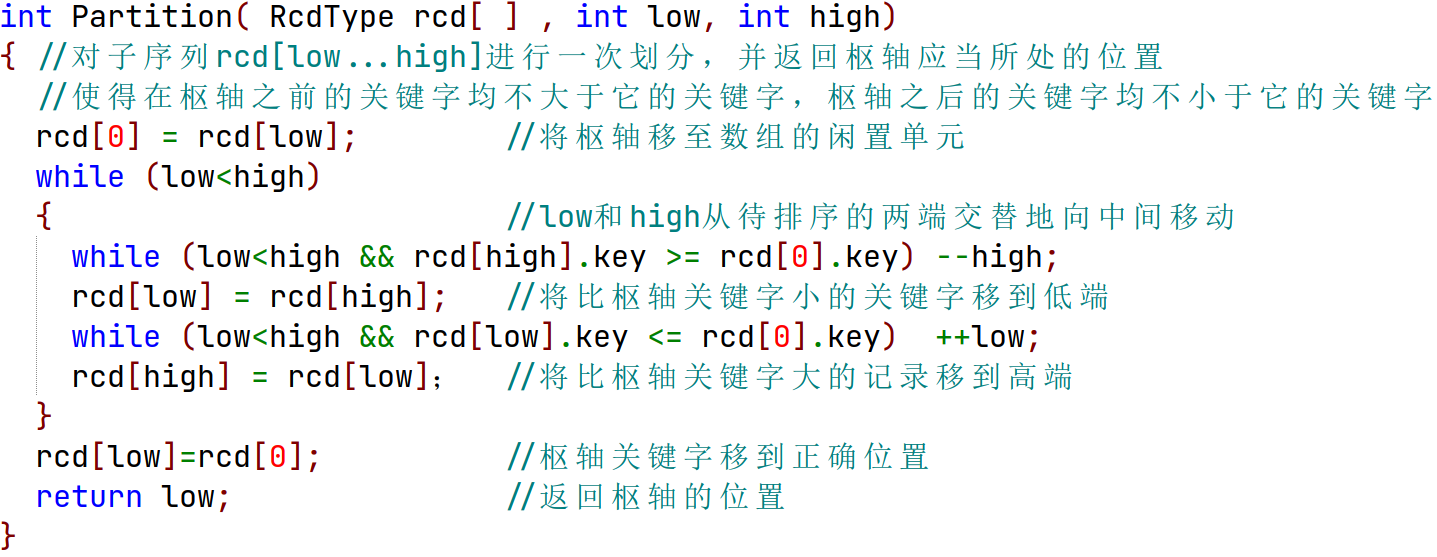
假设初始序列含有个记录，则可看成是个有序的子序列，每个子序列的长度为1，然后两两归并，得到个长度为2或1的有序子序列；再两两归并，如此重复，直至得到一个长度为的有序序列为止。







十二、划分算法和快速排序的枢轴



十三、迭代实现折半查找

