Name	Best	Average	Worst	Memory	Stable	Method	Other notes
In-place merge sort	—	—	$nlog^2n$	1	Yes	Merging	Can be implemented as a stable sort based on stable in-place merging.
Heapsort	$nlogn$	$nlogn$	$nlogn$	1	No	Selection	
Merge sort	$nlogn$	$nlogn$	$nlogn$	n	Yes	Merging	Highly parallelizable (up to $O(\log n)$ using the Three Hungarian's Algorithm)
Timsort	n	$nlogn$	$nlogn$	n	Yes	Insertion & Merging	Makes $n-1$ comparisons when the data is already sorted or reverse sorted.
Quicksort	$nlogn$	$nlogn$	n^2	$logn$	No	Partitioning	Quicksort is usually done in-place with $O(\log n)$ stack space.
Shellsort	$nlogn$	$n^{4/3}$	$n^{3/2}$	1	No	Insertion	Small code size.
Insertion sort	n	n^2	n^2	1	Yes	Insertion	$O(n + d)$, in the worst case over sequences that have d inversions.
Bubble sort	n	n^2	n^2	1	Yes	Exchanging	Tiny code size.
Selection sort	n^2	n^2	n^2	1	No	Selection	Stable with O(n) extra space, when using linked lists, or when made as a variant of Insertion Sort instead of swapping the two items.

1.1 散列表的基本概念

参考：数据结构（C语言版 第2版）(严蔚敏)，第7章 查找

前6周讨论了基于线性结构、树表结构的查找方法，这类查找方法都是以关键字的比较为基础的。

线性表是一种具有相同数据类型的有限序列，其特点是每个元素都有唯一的直接前驱和直接后继。换句话

说，线性表中的元素之间存在明确的线性关系，每个元素都与其前后相邻的元素相关联。

线性结构是数据结构中的一种基本结构，它的特点是数据元素之间存在一对一的关系，即除了第一个元素和

最后一个元素以外，其他每个元素都有且仅有一个直接前驱和一个直接后继。线性结构包括线性表、栈、队

列和串等。

因此，线性表是线性结构的一种具体实现，它是一种最简单和最常见的线性结构。

在查找过程中只考虑各元素关键字之间的相对大小，记录在存储结构中的位置和其关键字无直接关系，其查找时间

与表的长度有关，特别是当结点个数很多时，查找时要大量地与无效结点的关键字进行比较，致使查找速度很慢。

如果能在元素的存储位置和其关键字之间建立某种直接关系，那么在进行查找时，就无需做比较或做很少次的比

较，按照这种关系直接由关键字找到相应的记录。这就是散列查找法（Hash Search）的思想，它通过对元素的关

键字值进行某种运算，直接求出元素的地址，即使用关键字到地址的直接转换方法，而不需要反复比较。因此，散

列查找法又叫杂凑法或散列法。

下面给出散列法中常用的几个术语。

(1) 散列函数和散列地址：在记录的存储位置p和其关键字 key 之间建立一个确定的对应关系 H，使 p=H(key)，称这

个对应关系H为散列函数，p为散列地址。

(2) 散列表：一个有限连续的地址空间，用以存储按散列函数计算得到相应散列地址的数据记录。通常散列表的存

储空间是一个一维数组，散列地址是数组的下标。

(3) 冲突和同义词：对不同的关键字可能得到同一散列地址,即 key1≠key2,而 H(key1) = H(key2) 这种现象称为冲

突。具有相同函数值的关键字对该散列函数来说称作同义词，key1与 key2 互称为同义词。

例如，在Python语言中，可以针对给定的关键字集合建立一个散列表。假设有一个关键字集合为 S1，其中包括关

键字 main , int , float , while , return , break , switch , case , do 。为了构建散列表，可以定义一个长度为
26的散列表 HT ， 其中每个元素是一个长度为8的字符数组。假设我们采用散列函数 H(key) ，该函数将关键字 key
中的第一个字母转换为字母表 {a,b,...,z} 中的序号（序号范围为0~25），即 H(key) = ord(key[0]) - ord('a') 。根据此散列函数构造的散列表 HT 如下所示：
其中，假设关键字 key 的类型为长度为8的字符数组。根据给定的关键字集合和散列函数，可以将关键字插入到相
应的散列表位置。
表1
HT = [['' for _ in range(8)] for _ in range(26)]1
2

0 1 2 3 4 5 ... 8 ... 12 ... 17 18 ... 22 ... 25
break case do float int main return switch while
假设关键字集合扩充为:
S2 = S1 + {short, default, double, static, for, struct}
如果散列函数不变，新加人的七个关键字经过计算得到：H(short)=H(static)=H(struct)=18，H(default)=H(double)=3，H(for)=5，而 18、3 和5这几个位置均已存放相应的关键字，这就发生了冲突现象，其
中,switch、short、static 和 struct 称为同义词；float 和 for 称为同义词；do、default 和 double 称为同义词。
集合S2中的关键字仅有 15 个，仔细分析这 15个关键字的特性，应该不难构造一个散列函数避免冲突。但在实际应
用中，理想化的、不产生冲突的散列函数极少存在，这是因为通常散列表中关键字的取值集合远远大于表空间的地
址集。例如，高级语言的编译程序要对源程序中的标识符建立一张符号表进行管理，多数都采取散列表。在设定散
列函数时，考虑的查找关键字集合应包含所有可能产生的关键字，不同的源程序中使用的标识符一般也不相同，如
果此语言规定标识符为长度不超过8的、字母开头的字母数字串，字母区分大小写，则标识符取值集合的大小为:
而一个源程序中出现的标识符是有限的,所以编译程序将散列表的长度设为 1000 足矣。于是要将多达 10” 个可能的
标识符映射到有限的地址上，难免产生冲突。通常，散列函数是一个多对一的映射，所以冲突是不可避免的，只能
通过选择一个“好”的散列函数使得在一定程度上减少冲突。而一旦发生冲突，就必须采取相应措施及时予以解决。
综上所述，散列查找法主要研究以下两方面的问题:
(1) 如何构造散列函数;
(2) 如何处理冲突。
1.2 散列函数的构造方法
构造散列函数的方法很多，一般来说，应根据具体问题选用不同的散列函数，通常要考虑以下因素:
(1) 散列表的长度;
(2) 关键字的长度;
(3) 关键字的分布情况;
(4) 计算散列函数所需的时间;
(5) 记录的查找频率。
构造一个“好”的散列函数应遵循以下两条原则:(1)函数计算要简单，每一关键字只能有一个散列地址与之对应;(2)函

数的值域需在表长的范围内，计算出的散列地址的分布应均匀，尽可能减少冲突。下面介绍构造散列函数的几种常
用方法。
1.数字分析法
如果事先知道关键字集合，且每个关键字的位数比散列表的地址码位数多，每个关键字由n位数组成，如k1k2,...
kn，则可以从关键字中提取数字分布比较均匀的若干位作为散列地址。
例如，有 80个记录，其关键字为8位十进制数。假设散列表的表长为100，则可取两位十进制数组成散列地址，选
取的原则是分析这80个关键字，使得到的散列地址尽量避免产生冲突。假设这 80个关键字中的一部分如下所列:
3

对关键字全体的分析中可以发现:第①、②位都是“81”，第③位只可能取3或 4，第⑧位可能取 2、5或7，因此这4位
都不可取。由于中间的4位可看成是近乎随机的，因此可取其中任意两位，或取其中两位与另外两位的叠加求和后
舍去进位作为散列地址。
数字分析法的适用情况：事先必须明确知道所有的关键字每一位上各种数字的分布情况。
在实际应用中，例如，同一出版社出版的所有图书，其ISBN号的前几位都是相同的，因此，若数据表只包含同一出
版社的图书，构造散列函数时可以利用这种数字分析排除ISBN 号的前几位数字。
2.平方取中法
通常在选定散列函数时不一定能知道关键字的全部情况，取其中哪几位也不一定合适，而一个数平方后的中间几位
数和数的每一位都相关，如果取关键字平方后的中间几位或其组合作为散列地址，则使随机分布的关键字得到的散
列地址也是随机的，具体所取的位数由表长决定。平方取中法是一种较常用的构造散列函数的方法。
例如，为源程序中的标识符建立一个散列表，假设标识符为字母开头的字母数字串。假设人为约定每个标识的内部
编码规则如下：把字母在字母表中的位置序号作为该字母的内部编码，如 I 的内部编码为 09，D 的内部编码为
04，A 的内部编码为 01。数字直接用其自身作为内部编码,如 1的内部编码为 01，2 的内部编码为 02。根据以上编
码规则，可知“IDA1”的内部编码为09040101，同理可以得到“IDB2”、“XID3”和“YID4”的内部编码。之后分别对内部
编码进行平方运算，再取出第7位到第9位作为其相应标识符的散列地址，如表 2所示。
表2 标识符及其散列地址
3.折叠法将
4

关键字分割成位数相同的几部分（最后一部分的位数可以不同），然后取这几部分的叠加和（舍去进位）作为散列
地址，这种方法称为折叠法。根据数位叠加的方式，可以把折叠法分为移位叠加和边界叠加两种。移位叠加是将分
割后每一部分的最低位对齐，然后相加;边界叠加是将两个相邻的部分沿边界来回折看，然后对齐相加。
例如，当散列表长为 1000 时，关键字key=45387765213，从左到右按3 位数一段分割，可以得到 4个部分:453、
877、652、13。分别采用移位叠加和边界叠加，求得散列地址为 995 和914，如图 1 所示。
图 1由折叠法求得散列地址
折叠法的适用情况：适合于散列地址的位数较少，而关键字的位数较多，且难于直接从关键字中找到取

值较分散的
几位。
4.除留余数法
假设散列表表长为 m，选择一个不大于m 的数p，用p去除关键字，除后所得余数为散列地址，即
H(key) = key%p
这个方法的关键是选取适当的p，一般情况下，可以选p为小于表长的最大质数。例如，表长m=100，可取p=97。
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散列表的地址空间中。
1.3 处理冲突的方法
选择一个"好"的散列函数可以在一定程度上减少冲突，但在实际应用中，很难完全避免发生冲突，所以选择一个有
效的处理冲突的方法是散列法的另一个关键问题。创建散列表和查找散列表都会遇到冲突，两种情况下处理冲突的
方法应该一致。下面以创建散列表为例，来说明处理冲突的方法。
处理冲突的方法与散列表本身的组织形式有关。按组织形式的不同，通常分两大类:开放地址法和链地址法。
1.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 key 的初始散列地址 H0 = H(key)发
生冲突时，以 H0 为基础，采取合适方法计算得到另一个地址 H1，如果 H1 仍然发生冲突，以 为基础再求下一个
地址 H2，若 H2 仍然冲突，再求得 H3。依次类推，直至 Hk 不发生冲突为止，则 Hk 为该记录在表中的散列地
址。
5

这种方法在寻找 "下一个" 空的散列地址时，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：
Hi=(H(key) +di)%m i=1,2,...,k(k≤m-l)
其中，H(key)为散列函数，m 为散列表表长，d为增量序列。根据d取值的不同，可以分为以下3种探测方法。
(1) 线性探测法
di = 1, 2, 3, ..., m-1
这种探测方法可以将散列表假想成一个循环表，发生冲突时，从冲突地址的下一单元顺序寻找空单元，如果到最后
一个位置也没找到空单元，则回到表头开始继续查找，直到找到一个空位，就把此元素放入此空位中。如果找不到
空位，则说明散列表已满，需要进行溢出处理。
(2) 二次探测法
(3)伪随机探测法
di = 伪随机数序列
例如，散列表的长度为 11，散列函数 H(key)=key%11，假设表中已填有关键字分别为 17、60、29 的记录，如图
7.29(a)所示。现有第四个记录，其关键字为38，由散列函数得到散列地址为 5，产生冲突。
若用线性探测法处理时，得到下一个地址6，仍冲突；再求下一个地址7，仍冲突；直到散列地址为8的位置为"空"
时为止，处理冲突的过程结束，38填入散列表中序号为8的位置，如图2(b)所示。

若用二次探测法，散列地址5冲突后，得到下一个地址6，仍冲突；再求得下一个地址 4，无冲突，38填入序号为4
的位置，如图 2(c)所示。
若用伪随机探测法，假设产生的伪随机数为9，则计算下一个散列地址为(5+9)%11=3，所以38 填入序号为3 的位
置，如图 2(d)所示。

这种方法在寻找 ”下一个” 空的散列地址时，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：
Hi=(H(key) +di)%m i=1,2,…,k(k≤m-l)
其中，H(key)为散列函数，m 为散列表表长，d为增量序列。根据d取值的不同，可以分为以下3种探测方法。
(1) 线性探测法
di = 1, 2, 3, …, m-1
这种探测方法可以将散列表假想成一个循环表，发生冲突时，从冲突地址的下一单元顺序寻找空单元，如果到最后
一个位置也没找到空单元，则回到表头开始继续查找，直到找到一个空位，就把此元素放入此空位中。如果找不到
空位，则说明散列表已满，需要进行溢出处理。
(2) 二次探测法
(3)伪随机探测法
di = 伪随机数序列
例如，散列表的长度为 11，散列函数 H(key)=key%11，假设表中已填有关键字分别为 17、60、29 的记录，如图
7.29(a)所示。现有第四个记录，其关键字为38，由散列函数得到散列地址为 5，产生冲突。
若用线性探测法处理时，得到下一个地址6，仍冲突；再求下一个地址7，仍冲突；直到散列地址为8的位置为“空”
时为止，处理冲突的过程结束，38填入散列表中序号为8的位置，如图2(b)所示。
若用二次探测法，散列地址5冲突后，得到下一个地址6，仍冲突；再求得下一个地址 4，无冲突，38填入序号为4
的位置，如图 2(c)所示。
若用伪随机探测法，假设产生的伪随机数为9，则计算下一个散列地址为(5+9)%11=3，所以38 填入序号为3 的位
置，如图 2(d)所示。
6

图 2 用开放地址法处理冲突时，关键字为38的记录插入前后的散列表
从上述线性探测法处理的过程中可以看到一个现象:当表中i, i+1, i+2位置上已填有记录时，下一个散列地址为i、
i+1、i+2和i+3的记录都将填入i+3的位置，这种在处理冲突过程中发生的两个第一个散列地址不同的记录争夺同一
个后继散列地址的现象称作“二次聚集”(或称作“堆积”)，即在处理同义词的冲突过程中又添加了非同义词的冲突。
可以看出，上述三种处理方法各有优缺点。线性探测法的优点是：只要散列表未填满，总能找到一个不发生冲突的
地址。缺点是：会产生“二次聚集”现象。而二次探测法和伪随机探测法的优点是：可以避免“二次聚集”现象。缺点
也很显然：不能保证一定找到不发生冲突的地址。
2.链地址法
链地址法的基本思想是：把具有相同散列地址的记录放在同一个单链表中，称为同义词链表。有 m个散

列地址就有
m 个单链表，同时用数组 HT[0...m-1]存放各个链表的头指针，凡是散列地址为i的记录都以结点方式插入到以 HT[]
为头结点的单链表中。
【例】 已知一组关键字为 (19,14,23，1，68，20，84,27，55，11，10，79)，设散列函数 H(key)=key%13，用链
地址法处理冲突，试构造这组关键字的散列表。
由散列函数 H(key)=key %13 得知散列地址的值域为 0~12，故整个散列表有 13 个单链表组成，用数组 HT[0..12]
存放各个链表的头指针。如散列地址均为1的同义词 14、1、27、79 构成一个单链表，链表的头指针保存在 HT[1]
中，同理，可以构造其他几个单链表，整个散列表的结构如图 3 所示。
图 3 用链地址法处理冲突时的散列表
这种构造方法在具体实现时，依次计算各个关键字的散列地址，然后根据散列地址将关键字插入到相应的链表中。