**目录**

[[隐藏](javascript:toggleToc())]

* [1 基本概念](http://www.nocow.cn/index.php/%E6%95%A3%E5%88%97%E8%A1%A8#.E5.9F.BA.E6.9C.AC.E6.A6.82.E5.BF.B5)
* [2 常用的构造散列函数的方法](http://www.nocow.cn/index.php/%E6%95%A3%E5%88%97%E8%A1%A8#.E5.B8.B8.E7.94.A8.E7.9A.84.E6.9E.84.E9.80.A0.E6.95.A3.E5.88.97.E5.87.BD.E6.95.B0.E7.9A.84.E6.96.B9.E6.B3.95)
* [3 散列表的定义](http://www.nocow.cn/index.php/%E6%95%A3%E5%88%97%E8%A1%A8#.E6.95.A3.E5.88.97.E8.A1.A8.E7.9A.84.E5.AE.9A.E4.B9.89)
* [4 处理冲突的方法](http://www.nocow.cn/index.php/%E6%95%A3%E5%88%97%E8%A1%A8#.E5.A4.84.E7.90.86.E5.86.B2.E7.AA.81.E7.9A.84.E6.96.B9.E6.B3.95)
* [5 建立散列表](http://www.nocow.cn/index.php/%E6%95%A3%E5%88%97%E8%A1%A8#.E5.BB.BA.E7.AB.8B.E6.95.A3.E5.88.97.E8.A1.A8)
* [6 插入元素](http://www.nocow.cn/index.php/%E6%95%A3%E5%88%97%E8%A1%A8#.E6.8F.92.E5.85.A5.E5.85.83.E7.B4.A0)
* [7 散列函数](http://www.nocow.cn/index.php/%E6%95%A3%E5%88%97%E8%A1%A8#.E6.95.A3.E5.88.97.E5.87.BD.E6.95.B0)
* [8 查找元素](http://www.nocow.cn/index.php/%E6%95%A3%E5%88%97%E8%A1%A8#.E6.9F.A5.E6.89.BE.E5.85.83.E7.B4.A0)
* [9 删除元素](http://www.nocow.cn/index.php/%E6%95%A3%E5%88%97%E8%A1%A8#.E5.88.A0.E9.99.A4.E5.85.83.E7.B4.A0)
* [10 拉链法 （C++）](http://www.nocow.cn/index.php/%E6%95%A3%E5%88%97%E8%A1%A8#.E6.8B.89.E9.93.BE.E6.B3.95_.EF.BC.88C.2B.2B.EF.BC.89)
* [11 查找的性能分析](http://www.nocow.cn/index.php/%E6%95%A3%E5%88%97%E8%A1%A8#.E6.9F.A5.E6.89.BE.E7.9A.84.E6.80.A7.E8.83.BD.E5.88.86.E6.9E.90)
* [12 参见：](http://www.nocow.cn/index.php/%E6%95%A3%E5%88%97%E8%A1%A8#.E5.8F.82.E8.A7.81.EF.BC.9A)

基本概念

* 若结构中存在关键字和*K*相等的记录，则必定在*f*(*K*)的存储位置上。由此，不需比较便可直接取得所查记录。称这个对应关系*f*为[散列函数](http://www.nocow.cn/index.php?title=%E6%95%A3%E5%88%97%E5%87%BD%E6%95%B0&action=edit" \o "散列函数)(Hash function)，按这个思想建立的表为**散列表**。
* 对不同的关键字可能得到同一散列地址，即*key*1≠*key*2，而*f*(*key*1)=*f*(*key*2)，这种现象称冲突。具有相同函数值的关键字对该散列函数来说称做同义词。综上所述，根据散列函数*H*(key)和处理冲突的方法将一组关键字映象到一个有限的连续的地址集（区间）上，并以关键字在地址集中的“象”作为记录在表中的存储位置，这种表便称为散列表，这一映象过程称为[散列造表](http://www.nocow.cn/index.php?title=%E6%95%A3%E5%88%97%E9%80%A0%E8%A1%A8&action=edit" \o "散列造表)或[散列](http://www.nocow.cn/index.php?title=%E6%95%A3%E5%88%97&action=edit)，所得的存储位置称[散列地址](http://www.nocow.cn/index.php?title=%E6%95%A3%E5%88%97%E5%9C%B0%E5%9D%80&action=edit)。
* 若对于关键字集合中的任一个关键字，经散列函数映象到地址集合中任何一个地址的概率是相等的，则称此类散列函数为[均匀散列函数](http://www.nocow.cn/index.php?title=%E5%9D%87%E5%8C%80%E6%95%A3%E5%88%97%E5%87%BD%E6%95%B0&action=edit" \o "均匀散列函数)(Uniform Hash function)，这就是使关键字经过散列函数得到一个“随机的地址”，从而减少冲突。

[[编辑](http://www.nocow.cn/index.php?title=%E6%95%A3%E5%88%97%E8%A1%A8&action=edit&section=2)] 常用的构造散列函数的方法

散列函数能使对一个数据序列的访问过程更加迅速有效，通过散列函数，数据元素将被更快地定位ǐ

1. 直接定址法：取关键字或关键字的某个线性函数值为散列地址。即H(key)=key或H(key) = a•key + b，其中a和b为常数（这种散列函数叫做自身函数）
2. [数字分析法](http://www.nocow.cn/index.php/%E6%95%B0%E5%AD%97%E5%88%86%E6%9E%90%E6%B3%95)
3. 平方取中法:计算关键值再取中间r位形成一个2^r位的表
4. 折叠法:把所有字符的ASCII码加起来 (对于字符串)
5. [随机数法](http://www.nocow.cn/index.php/%E9%9A%8F%E6%9C%BA%E6%95%B0%E6%B3%95)
6. 除留余数法：取关键字被某个不大于散列表表长m的数p除后所得的余数为散列地址。即 H(key) = key MOD p, p<=m。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。对p的选择很重要，一般取素数或m，若p选的不好，容易产生同义词。
7. 针对字符串的一些常用方法，比如[ELFHash](http://www.nocow.cn/index.php/ELFHash)和[BKDRHash](http://www.nocow.cn/index.php/BKDRHash" \o "BKDRHash)（更易于编写，效率不错）

散列表的定义

C/C++:

#define NIL -1 //空结点标记

#define M 997 //表长度，这个值依赖于实际需求，一般应取大素数

typedef struct{ *//散列表结点类型*

int key; *//存储的内容，类型不一定要int*

InfoType otherinfo; *//此类依赖于应用*

}NodeType;

typedef NodeType HashTable[m]; *//散列表类型*

处理冲突的方法

1. 开放定址法；Hi=(H(key) + di) MOD m, i=1,2,…, k(k<=m-1)，其中H(key)为散列函数，m为散列表长，di为增量序列，可有下列三种取法：
   1. di=1,2,3,…, m-1，称线性探测再散列；
   2. di=1^2, -1^2, 2^2,-2^2, 3^2, …, ±k^2,(k<=m/2)称二次探测再散列；
   3. di=伪随机数序列，称伪随机探测再散列。
2. 再散列法：Hi=RHi(key), i=1,2,…,k RHi均是不同的散列函数，即在同义词产生地址冲突时计算另一个散列函数地址，直到冲突不再发生，这种方法不易产生“聚集”，但增加了计算时间。
3. 链地址法(拉链法)
4. 建立一个公共溢出区

[[编辑](http://www.nocow.cn/index.php?title=%E6%95%A3%E5%88%97%E8%A1%A8&action=edit&section=5)] 建立散列表

建表时首先要将表中各结点的关键字清空，使其地址为开放的；然后调用插入算法将给定的关键字序列依次插入表中。 C/C++：

void CreateHashTable(HashTable T,NodeType A[],int n) *//根据A[0..n-1]中结点建立散列表T[0..m-1]*

{

int i;

if (n>m) *//用开放定址法处理冲突时，装填因子α须不大于1*

Error("Load factor>1!");

for(i=0;i<m;i++)

T[i].key=NIL； *//将各关键字清空，使地址i为开放地址*

for(i=0;i<n;i++) *//依次将A[0..n-1]插入到散列表T[0..m-1]中*

Hashlnsert(T，A[i])；

}

插入元素

插入算法首先调用查找算法，若在表中找到待插入的关键字或表已满，则插入失败；若在表中找到一个开放地址，则将待插入的结点插入其中，即插入成功。 C/C++：

void HashInsert(HashTable T,NodeType new) *//将新结点new插入散列表T[0..m-1]中*

{

int pos,sign;

sign=HashSearch(T,new.key,&pos); *//在表T中查找new的插入位置*

if(!sign) *//找到一个开放的地址pos*

T[pos]=new; *//插入新结点new，插入成功*

else *//插人失败*

if(sign>0)

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("duplicate key!"); *//重复的关键字*

else *//sign<0*

Error("hashtableoverflow!"); *//表满错误，终止程序执行*

}

散列函数

C/C++：

int Hash(int k,int i) *//求在散列表T[0..m-1]中第i次探查的散列地址hi，0≤i≤m-1*

{

*//下面的h是散列函数。Increment是求增量序列的函数，它依赖于解决冲突的方法*

return (h(k)+Increment(i))%m; *//Increment(i)相当于是d i*

}

若散列函数用除余法构造，并假设使用线性探查的开放定址法处理冲突，则上述函数中的h(K)和Increment(i)可定义为：

int h(int K) *//用除余法求K的散列地址*

{

return K%m;

}

int Increment(int i) *//用线性探查法求第i个增量d i*

{

return i; *//若用二次探查法，则返回i\*i*

}

查找元素

C/C++：

int HashSearch(HashTable T，KeyType K，int \*pos) *//在散列表T[0..m-1]中查找K，成功时返回1。失败有两种情况：找到*

*//一个开放地址时返回0，表满未找到时返回-1。 \*pos记录找到K或*

*//找到空结点时表中的位置*

{

int i=0； *//记录探查次数*

do

{

\*pos=Hash(K，i); *//求探查地址hi*

if(T[\*pos].key==K) return -1; *//查找成功返回*

if(T[\*pos].key==NIL) return 0;*//查找到空结点返回*

}

while(++i<m) *//最多做m次探查*

return -1; *//表满且未找到时，查找失败*

}

[[编辑](http://www.nocow.cn/index.php?title=%E6%95%A3%E5%88%97%E8%A1%A8&action=edit&section=9)] 删除元素

无

[[编辑](http://www.nocow.cn/index.php?title=%E6%95%A3%E5%88%97%E8%A1%A8&action=edit&section=10)] 拉链法 （C++）

int find(int k) {

Int h = hash(k);

Int p = first[h];

While (p)

{

if(key[p] == k) return p;

p = next [p];

}

return 0;

}

void insert(int x) {

Int h = hash(key[x]);

next[x] = first[h];

first[h] = x;

}

[[编辑](http://www.nocow.cn/index.php?title=%E6%95%A3%E5%88%97%E8%A1%A8&action=edit&section=11)] 查找的性能分析

散列表的查找过程基本上和造表过程相同。一些关键码可通过散列函数转换的地址直接找到，另一些关键码在散列函数得到的地址上产生了冲突，需要按处理冲突的方法进行查找。在介绍的三种处理冲突的方法中，产生冲突后的查找仍然是给定值与关键码进行比较的过程。所以，对散列表查找效率的量度，依然用平均查找长度来衡量。

查找过程中，关键码的比较次数，取决于产生冲突的多少，产生的冲突少，查找效率就高，产生的冲突多，查找效率就低。因此，影响产生冲突多少的因素，也就是影响查找效率的因素。影响产生冲突多少有以下三个因素：

1. 散列函数是否均匀；
2. 处理冲突的方法；
3. 散列表的装填因子。

散列表的装填因子定义为：α= 填入表中的元素个数 / 散列表的长度

α是散列表装满程度的标志因子。由于表长是定值，α与“填入表中的元素个数”成正比，所以，α越大，填入表中的元素较多，产生冲突的可能性就越大；α越小，填入表中的元素较少，产生冲突的可能性就越小。

实际上，散列表的平均查找长度是装填因子α的函数，只是不同处理冲突的方法有不同的函数。