<https://blog.csdn.net/weiyuefei/article/details/52048036>

[数据结构HASH总结一：理论学习篇](http://blog.csdn.net/yankai0219/article/details/8185796)

[数据结构HASH总结二：程序学习篇](http://blog.csdn.net/yankai0219/article/details/8185816)

[数据结构HASH总结三：实践基础篇](http://blog.csdn.net/yankai0219/article/details/8185847)

[数据结构HASH总结四：程序高级篇](http://blog.csdn.net/yankai0219/article/details/8185872)

[数据结构HASH总结五：Nginx中的HASH（version 0.1）](http://blog.csdn.net/yankai0219/article/details/8186998)

转载请注明出处<http://blog.csdn.net/yankai0219/article/details/8185796>

零、学习方法

     简要学习理论篇，进入程序学习篇，再回头学习理论篇和实践篇

一、基本概念

**1.Hash定义**

     Hash定义：将**任意长度**的输入，通过**散列算法**，变成**固定长度**的输出，该输出就是散列值。

     hash函数：就是一种将任意长度的消息压缩到某一固定长度的消息摘要的函数。

     Hash表：一种数据结构，既满足了数据的查找方便，同时不占用太多的内容空间。

**2.Hash用途**

     1）哈希主要用于信息安全领域中的加密算法，把一些不同长度的信息转化成杂乱的128位的编码。

     2）在海量数据处理中有广泛应用。

**3.常见的Hash函数（散列算法）**

     1)除法散列法

     2)平方散列法

     3）斐波那契散列法

**4.Hash表介绍**

     Hash表本质：归类方法。

     Hash表：一种新的数据结构，兼有数组的易寻址和链表的易插入和删除的特点。

     Hash表形象描述：如果有一堆书，就如同链表及线性表一样，杂乱无序，查找起来十分麻烦；

                                   如果进行编号，采用二分法，很快就可以查到

                                   如果可以按照工科、理科、文科进行分类，就可以更快的查到。

     Hash表的实现：有多种实现，最常用的是拉链法。

**5.举例说明Hash函数**

在下面这一个字符串hash函数中，通过遍历字符串，经过表达式hash = 31\*hash +\*p，循环计算得到Hash值。

很多hash函数都是如此操作，只是其表达式（散列算法）有所不同。

|  |
| --- |
| unsigned int yk\_simple\_hash(char \*str,int str\_len) {         register unsigned int hash;         register unsigned char \*p;         int i;          for(hash = 0, i = 0, p = (unsigned char \*)str; \*p && i < str\_len; p++,i++)                 hash = 31 \* hash + \*p;          return (hash & 0x7FFFFFFF); } |

**二、冲突相关**

1.冲突定义：假设哈希表的地址集为0~（n-1），冲突是指由关键字得到的哈希地址为j（0<=j<=n-1）的位置上已经有记录。**在关键字得到的哈希地址上已经有记录，那么就称之为冲突**

2.处理冲突：就是为该关键字的记录扎到另一个“空”的哈希地址。即在处理哈希地址的冲突时，若得到的另一个哈希地址H1仍然发生冲突，则再求下一个地址H2，若H2仍然冲突，再求的H3，直至Hk不发生冲突为止，则Hk为记录在表中的地址。

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)为哈希函数；m为哈希表表长；di为增量序列。有3中增量序列：

               1）线性探测再散列：di=1,2,3,...,m-1

               2）二次探测再散列：di=1^2,-1^2,2^2,-2^2,....+-k^2(k<=m/2)

               3）伪随机探测再散列：di=伪随机数序列

**缺点：**

          我们可以看到一个现象：当表中i,i+1,i+2位置上已经填有记录时，下一个哈希地址为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哈希地址不同的记录争夺同一个后继哈希地址的现象称为“**二次聚集**”，即在处理同义词的冲突过程中又添加了非同义词的冲突。但另一方面，用线性探测再散列处理冲突可以保证做到：只要哈希表未填满，总能找到一个不发生冲突的地址Hk。而二次探测再散列只有在哈希表长m为形如4j+3（j为整数）的素数时才可能。

**即开放定址法会造成二次聚集的现象，对查找不利**。

**2)再哈希法**

               Hi = RHi（key），i=1,2,...k

               RHi均是不同的哈希函数，即在同义词产生地址冲突时计算另一个哈希函数地址，直到不发生冲突为止。这种方法不易产生聚集，但是增加了计算时间。

               缺点：增加了计算时间。

**3)链地址法（拉链法）**

               将所有关键字为同义词的记录存储在同一线性链表中。

**4)建立一个公共溢出区**

               假设哈希函数的值域为[0,m-1]，则设向量HashTable[0...m-1]为基本表，每个分量存放一个记录，另设立向量OverTable[0....v]为溢出表。所有关键字和基本表中关键字为同义词的记录，不管他们由哈希函数得到的哈希地址是什么，一旦发生冲突，都填入溢出表。

          拉链法的优点:

①拉链法处理冲突简单，且无堆积现象，即非同义词决不会发生冲突，因此平均查找长度较短；  
②由于拉链法中各链表上的结点空间是动态申请的，故它更适合于造表前无法确定表长的情况；  
③开放定址法为减少冲突，要求装填因子α较小，故当结点规模较大时会浪费很多空间。而拉链法中可取α≥1，且结点较大时，拉链法中增加的指针域可忽略不计，因此节省空间；  
④在用拉链法构造的散列表中，删除结点的操作易于实现。只要简单地删去链表上相应的结点即可。而对开放地址法构造的散列表，删除结点不能简单地将被删结 点的空间置为空，否则将截断在它之后填人散列表的同义词结点的查找路径。这是因为各种开放地址法中，空地址单元(即开放地址)都是查找失败的条件。因此在 用开放地址法处理冲突的散列表上执行删除操作，只能在被删结点上做删除标记，而不能真正删除结点

               拉链法的缺点：

                    拉链法的缺点是：指针需要额外的空间，故当结点规模较小时，开放定址法较为节省空间，而若将节省的指针空间用来扩大散列表的规模，可使装填因子变小，这又减少了开放定址法中的冲突，从而提高平均查找速度

**三.查找：**

     从哈希表的查找过程可见：

          1）虽然哈希表在关键字与记录的存储位置直接建立了直接映像，但是由于“冲突”的产生，使得哈希表的查找过程仍然是一个给定值和关键字进行比较的过程。因此仍需以**平均查找长度**作为衡量哈希表的查找效率的量度。

          2)查找过程中需和给定值进行比较的关键字的个数取决于下列三个因素：哈希函数，处理冲突的方法和哈希表的装填因子。

               在一般情况下，处理冲突方法相同的哈希表，其平均查找长度依赖于哈希表的装填因子。

**装填因子=（表中填入的记录数）/(哈希表的长度)**. 装填因子越小，发生冲突的可能性就越小；反之，装填因子越大，表中已经填入的记录越多，再填记录时，发生冲突的可能性就越大，则查找时，给定值需与之进行比较的关键字的个数也就越多。

**四 关于Hash使用与学习**

     首先一定要明白Hash表的拉链法。因为很多应用都是采用Hash表的拉链法。对于拉链法，我们可以理解为是一个链表的数组。1）数组的每一个元素都是一个链表。2）一个链表中的所有结点都具有相同的Hash值，其Hash值就是这个数组元素的下标。

     其次，要学会Hash表初始化、插入元素、查找元素三大操作。

转载请注明出处<http://blog.csdn.net/yankai0219/article/details/8185796>