问题

哈希冲突及其解决方法是面试中的常见问题之一，需要掌握一下。这节内容的概念偏多，为了不造成阅读疲劳，可能会适当地简写一些，如果想深入理解的话，可以看看本文最后的参考博客。

哈希表

在处理哈希冲突之前，首先得清楚什么是哈希表，以及哈希函数如何产生的。

哈希表（hash table）是一种通过关键字（key）和值（value）进行访问的数据结构。

既然要访问，那肯定需要知道访问的内容存放在哪里，所以这就需要通过一种映射关系将访问元素的关键字和元素的存储地址关联起来，而这种映射关系就是通过哈希函数来实现的。

哈希函数

在元素的关键字 k 和元素的存储位置 p 之间建立一个对应关系 f，使得 $p = f(k)$，这个对应关系 f 就称为哈希函数。

按照这种对应关系，在创建哈希表时，将关键字为 k 的元素存放在地址为 $f(k)$ 的单元上；之后若需要查找关键字 k 时，再利用哈希函数计算出该元素的存储位置 $p = f(k)$，就可以直接获取到该元素了。（这应该也是为什么哈希查找时间复杂度为$O(1)$的原因吧）

那我们怎么知道这个映射关系 f 是什么呢？那就需要自己构造一个这样的哈希函数了。

哈希函数的构造原则：

①函数本身便于计算；

②计算出来的地址分布均匀，即对任一关键字 k，$f(k)$对应不同地址的概率相等，目的是尽可能减少冲突。

接下来介绍5种构造哈希函数的方法：（具体原理不展开讲述了，有需要了解可以看看参考博客，担心面试问这么详细的话可以记住其中一两种方法应该就够了，例如**除留余数法和伪随机数法**，其他的了解即可）

1.数字分析法

如果事先知道关键字集合，并且每个关键字的位数比哈希表的地址码位数多时，可以从关键字中选出分布较均匀的若干位，构成哈希地址。

2.平方取中法

当无法确定关键字中哪几位分布较均匀时，可以先求出关键字的平方值，然后按需要取平方值的中间几位作为哈希地址。

3.分段叠加法

这种方法是按哈希表地址位数将关键字分成位数相等的几部分（最后一部分可以较短），然后将这几部分相加，舍弃最高进位后的结果就是该关键字的哈希地址。

4.除留余数法

假设哈希表长为 m，p 为小于等于 m的最大素数，则哈希函数为：

$$H(k) = k\%p, \text{ 其中}\%\text{为取余运算} \tag{1}$$

例如，已知待散列元素为（18，75，60，43，54，90，46），表长m=10，p=7，则有

h(18)=18 % 7=4　h(75)=75 % 7=5　h(60)=60 % 7=4

h(43)=43 % 7=1　h(54)=54 % 7=5　h(90)=90 % 7=6

h(46)=46 % 7=4

此时冲突较多。为减少冲突，可取较大的m值和p值，如m=p=13，结果如下：

h(18)=18 % 13=5　h(75)=75 % 13=10　h(60)=60 % 13=8

h(43)=43 % 13=4　h(54)=54 % 13=2　h(90)=90 % 13=12

h(46)=46 % 13=7

此时没有冲突。

5.伪随机数法

采用一个伪随机函数做哈希函数，即 $H(key) = random(key)$。

哈希冲突

哈希冲突产生的原因

因为通过哈希函数产生的值是有限的，而数据可能比较多，导致通过哈希函数映射后仍有很多不同的数据对应了相同的哈希值，这时候就产生了哈希冲突。

例如，用除留余数法构造一个$H(k) = k\%p$ 的哈希函数，若散列表为 $\{18，75，60，43，54，90，46\}$，$p = 7$，那么经过哈希函数计算后的结果为 $\{4，5，4，1，5，6，4\}$，可以发现，有些原本不同的数据却对应到了相同的哈希值上了，因此发生了冲突。

解决哈希冲突的四种方法

通过构造性能良好的哈希函数，可以减少冲突，但一般不可能完全避免冲突，因此解决冲突是哈希法的另一个关键问题。创建哈希表和查找哈希表都会遇到冲突，两种情况下解决冲突的方法应该一致。下面以创建哈希表为例，说明解决冲突的方法。常用的解决冲突方法有以下四种：

1.开放地址法（也称再散列法）

基本思想是：当关键字 key 的哈希地址 $p = H(key)$ 出现冲突时，以 p为基础，产生另一个哈希地址 $p1$，如果 $p1$ 仍然冲突，再以 p 为基础，产生另一个哈希地址 $p2$，...，**直到找出一个不冲突的哈希地址 p_i，将相应元素存入其中**。

这种方法有一个通用的再散列函数形式：

$$H_i = (H(key) + d_i)\%m，\ i = 1, 2, \ldots, n. \tag{2}$$

其中，$H(key)$为哈希函数，m为表长，d_i为增量序列。

增量序列 d_i 的取值方式不同，相应的再散列方式也不同，主要有以下三种：

(1)线性探测

增量序列为：$d_i = 1, 2, \ldots, m-1$

其特点是：在发生冲突时，顺序查看表中下一单元，直到找出一个空单元或查遍全表，将冲突的元素存放进去。

(2)再平方探测

增量序列为： $d_i = 1^2, -1^2, 2^2, -2^2, \ldots, k^2, -k^2. \ (k \leq m/2)$

其特点是：在发生冲突时，在表的左右进行跳跃试探，比较灵活。

(3)伪随机探测

增量序列： $d_i = 伪随机序列$

具体实现时，建立一个伪随机数发生器生成伪随机序列（如2，5，9，.....），在发生冲突时，先令 $d_i = 2$，算出 H_i，如果还有冲突，则再令 $d_i = 5$，算出 H_i，若还有冲突，则继续下去.....

2.链式地址法

基本思想：将所有哈希地址为 i 的元素构成一个称为同义词链的单链表，并将单链表的头指针存在哈希表的第 i 个单元中，因而查找、插入和删除都比较方便。

例如，已知一组关键字（32，40，36，53，16，46，71，27，42，24，49，64），哈希表长度为13，哈希函数为：H（key）＝key % 13，则用链地址法处理冲突的结果如图8.27所示：

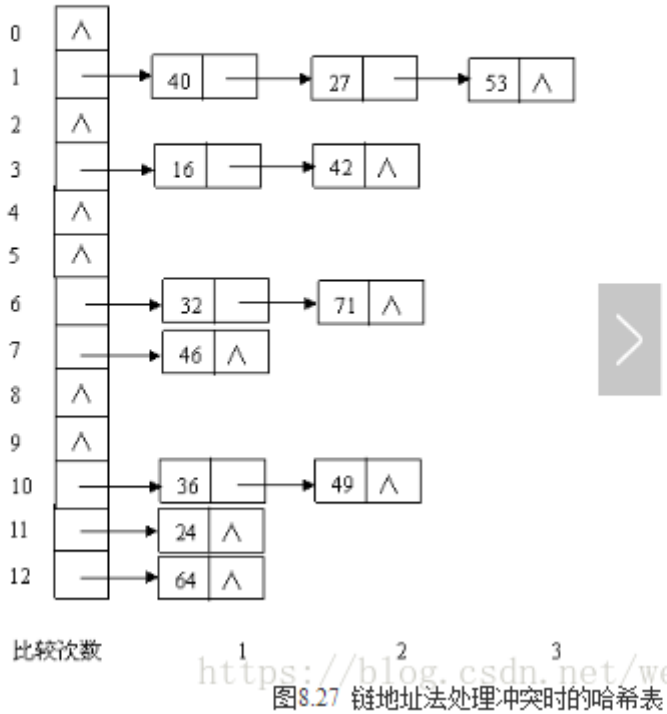

图8.27 链地址法处理冲突时的哈希表

3.建立公共溢出区

基本思想是：将哈希表分为**基本表**和**溢出表**两部分，凡是和基本表发生冲突的元素，一律填入溢出表中。

4.再哈希表

基本思想是：对于冲突的哈希值，再构造另一个哈希函数进行处理，直至没有哈希冲突，也就是同时使用多个哈希函数来处理。

参考资料

解决哈希（HASH）冲突的主要方法

哈希表和哈希冲突