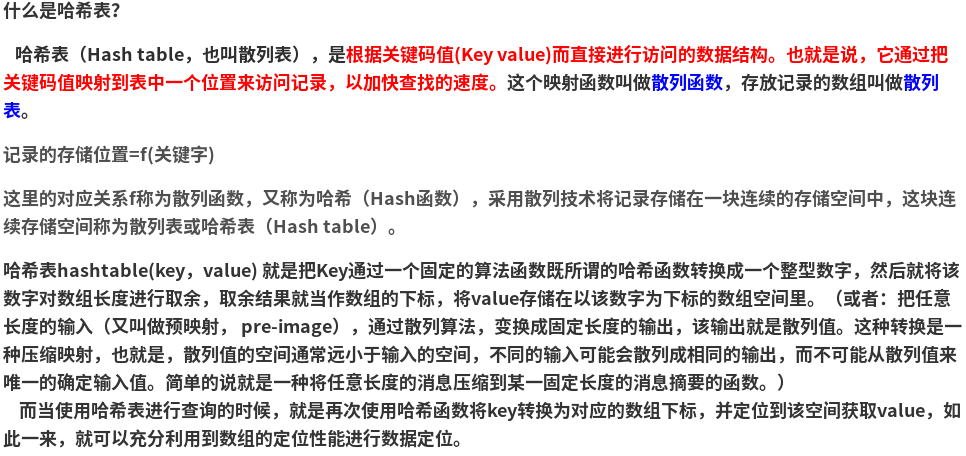
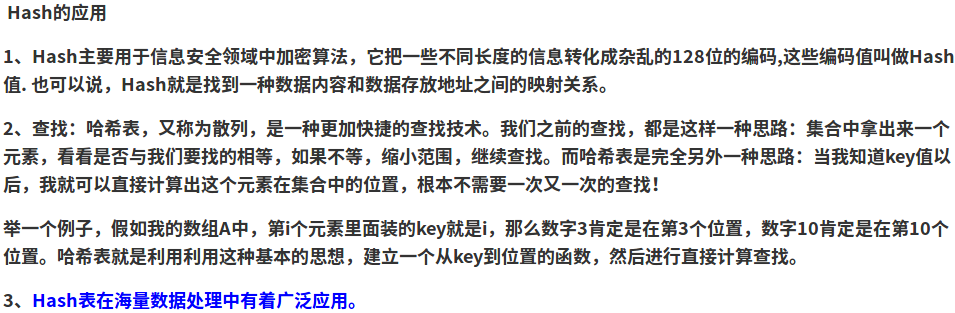
**哈希表总结**

哈希表的本质是一个数组，数组中每一个元素称为一个箱子(bin)，箱子中存放的是键值对。





**哈希函数的构造方法：**

1. **直接定址法**

取关键字或关键字的某个线性函数值为哈希地址，即：

H(key) = key 或 H(key) = a·key + b

1. **数字分析法**

假设关键字是以r为基的数，并且哈希表中可能出现的关键字都是事先知道的，则可取关键字的若干数位组成哈希表。

1. **平方取中法**

取关键字平方后的中间几位为哈希地址。一个数平方后的中间几位数和数的每一位都相关，由此使随机分布的关键字得到的哈希地址也是随机的。

1. **折叠法**

将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)作为哈希地址。

1. **除留余数法（最简单、最常用）**

取关键字被某个不大于哈希表表长m的数p除后所得余数为哈希地址。即：

H(key) = key MOD p ，p <= m

它不仅可以对关键字直接取模，也可以在折叠、平方取中等运算之后取模。

注：对p的选择很重要。一般可以选p为质数或不包含小于20的质因数的合数。

1. **随机数法**

选择一个随机函数，取关键字的随机函数值为它的哈希地址，即：H(key) = random(key)

**处理冲突的方法**

“处理冲突”就是为该关键字的记录找到另一个“空”的哈希地址。在处理冲突的过程中可能得到一个地址序列Hi , i = 1,2,...,k ，(Hi∈[0, n-1])。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，再求得H3。以此类推，直至Hk不发生冲突为止，则Hk为记录在表中的地址。

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

di的三种取法：

1. **线性探测再散列**(最常用，只要哈希表未填满，总能找到一个不冲突地址Hk)

di = 1, 2, 3, ... , m-1

1. **二次探测再散列**

di = 12, -12, 22, -22, 32, ..., ±k2 (k <= m/2)

1. **伪随机探测再散列**

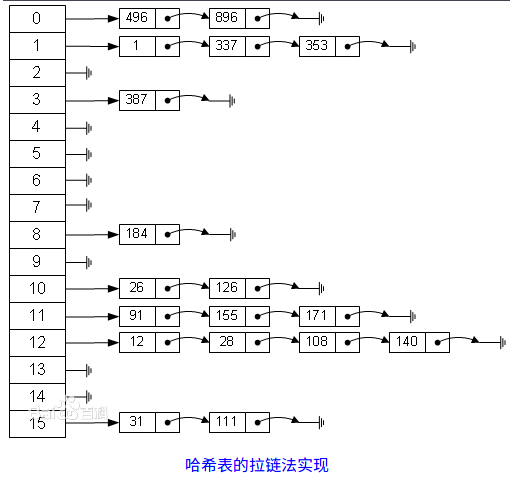
di = 伪随机数序列

1. **再哈希法**

Hi = RHi(key) i = 1, 2, ... ,k RHi均是不同的哈希函数

1. **链地址法**

链地址法的原理时如果遇到冲突，他就会在原地址新建一个空间，然后以链表结点的形式插入到该空间。我感觉业界上用的最多的就是链地址法。



1. **建立一个公共溢出区**

**哈希表的存储过程如下:**

1. 根据 key 计算出它的哈希值 h。
2. 假设箱子的个数为 n，那么这个键值对应该放在第 **(h % n)** 个箱子中。
3. 如果该箱子中已经有了键值对，就使用开放寻址法或者拉链法解决冲突。

**使用哈希查找有两个步骤:**

1. 使用哈希函数将被查找的键转换为数组的索引。在理想的情况下，不同的键会被转换为不同的索引值，但是在有些情况下我们需要处理多个键被哈希到同一个索引值的情况。所以哈希查找的第二个步骤就是处理冲突
2. 处理哈希碰撞冲突。有很多处理哈希碰撞冲突的方法，本文后面会介绍拉链法和线性探测法。

哈希表还有一个重要的属性: 负载因子(load factor)，它用来衡量哈希表的 **空/满** 程度，一定程度上也可以体现查询的效率，计算公式为:

负载因子 = 总键值对数 / 箱子个数

负载因子越大，意味着哈希表越满，越容易导致冲突，性能也就越低。因此，一般来说，当负载因子大于某个常数(可能是 1，或者 0.75 等)时，哈希表将自动扩容。

哈希表在自动扩容时，一般会创建两倍于原来个数的箱子，因此即使 key 的哈希值不变，对箱子个数取余的结果也会发生改变，因此所有键值对的存放位置都有可能发生改变，这个过程也称为重哈希(rehash)。

哈希表的扩容并不总是能够有效解决负载因子过大的问题。假设所有 key 的哈希值都一样，那么即使扩容以后他们的位置也不会变化。虽然负载因子会降低，但实际存储在每个箱子中的链表长度并不发生改变，因此也就不能提高哈希表的查询性能。

基于以上总结，细心的读者可能会发现哈希表的两个问题:

1. 如果哈希表中本来箱子就比较多，扩容时需要重新哈希并移动数据，性能影响较大。
2. 如果哈希函数设计不合理，哈希表在极端情况下会变成线性表，性能极低。