### **问题：什么是哈希表？什么是哈希冲突?如何解决哈希冲突？**

#### **1.哈希表定义**

**官方：**哈希表(散列表)是根据关键码的值而直接进行访问的数据结构。它通过把关键码值映射到表中一个位置来访问记录，以加快查找的速度。这个映射函数叫做散列函数。

**简单点：**哈希表就是通过一个映射函数f(key)将一组数据散列存储在数组中的一种数据结构。

**例：**给定表M，存在函数f(key)，对任意给定的关键字值key，代入函数后若能得到包含该关键字的记录在表中的地址，则称表M为哈希(Hash）表，函数f(key)为哈希(Hash) 函数。

ps：数组就是一张哈希表，时间复杂度为O(1)

#### **2.哈希冲突**

不同关键字可能得到同一散列地址，这种现象称作哈希冲突

#### **3.如何减少哈希冲突**

（1）除留取余法：取关键字被某个不大于哈希表长m的数p除后所得余数为哈希地址。即：f(key)=key % p, p≤m;

（2）直接定址法：直接定址法是指取关键字或关键字的某个线性函数值为哈希地址。即：f(key)=key 或者 f(key)=a\*key+b

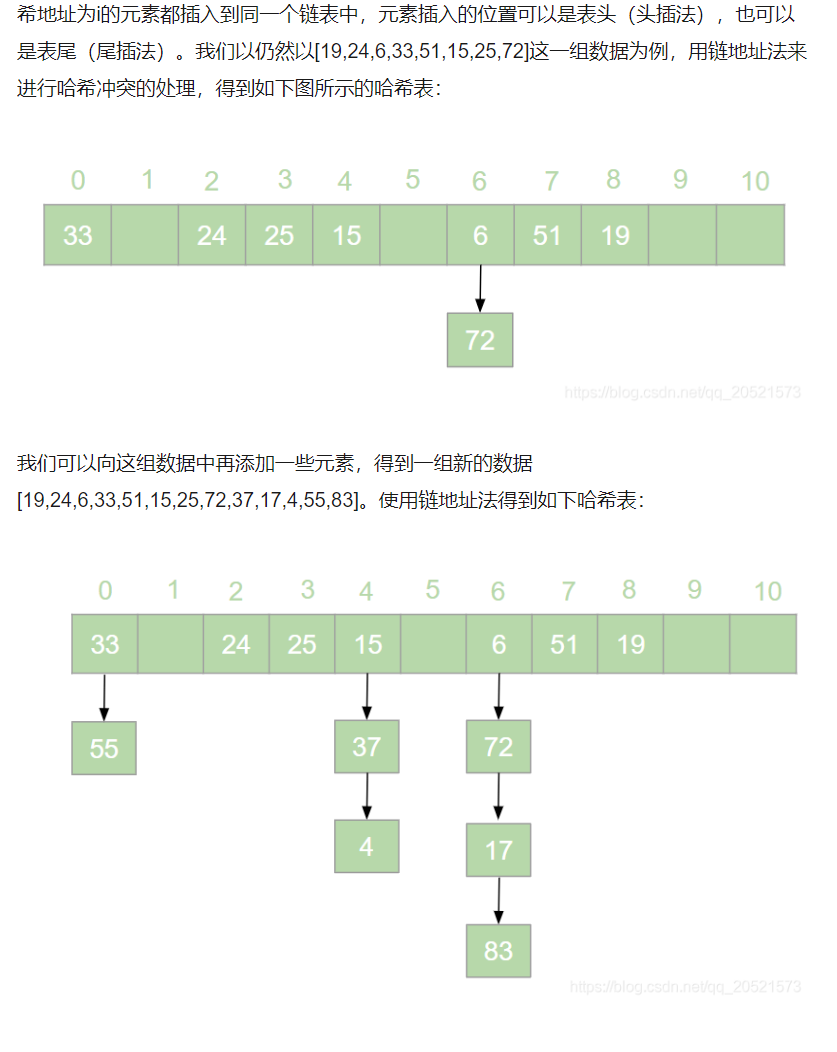
（3）数字分析法：假设关键字是以为基的数（如以10为基的十进制数），并且哈希表中可能出现的关键字都是事先知道的，则可以选取关键字的若干位数组成哈希表。

#### **4.如何解决哈希冲突**

（1）**开放定址法**：开放定址法是指当发生地址冲突时，按照某种方法继续探测哈希表中的其他存储单元，直到找到空位置为止。

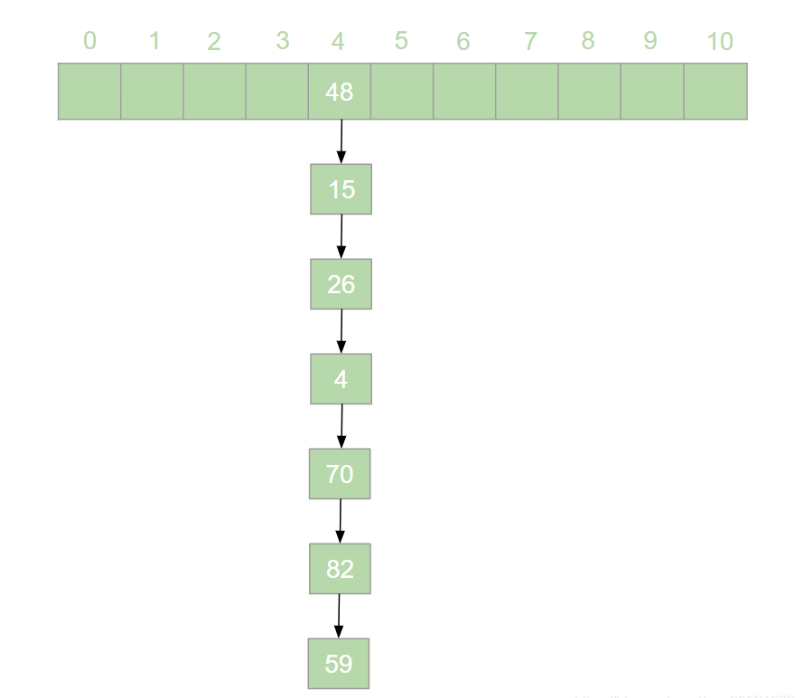


1. **链地址法**：链地址法是指在碰到哈希冲突的时候，将冲突的元素以链表的形式进行存储。



#### **Ps：链地址法优化**

虽然链地址法是一种很好的处理哈希冲突的方法，但是在一些极端情况下链地址法也会出现问题。举个例子，我们现在有这样一组数据：[48,15,26,4,70,82,59]。我们将这组数据仍然散列存储到长度为11的数组中，此时则得到了如下的结果：



可以发现，此时的哈希表俨然已经退化成了一个链表，当我们在这样的数据结构中去查找某个元素的话，时间复杂度又变回了o(n)。

这显然不符合我们的预期。因此，当哈希表中的链表过长时就需要我们对其进行优化。我们知道，二叉查找树的查询效率是远远高于链表的。

因此，当哈希表中的链表过长时我们就可以把这个链表变成一棵红黑树。上面的一组数据优化后可得到如下结果：

#### **@B6I9$XBKM9YL[3GL7_F@{X**

红黑树是一个可以自平衡的二叉查找树。

****它的查询的时间复杂度为o(lgn)。通过这样的优化可以提高哈希表的查询效率。****

#### **用途：**

一般哈希表都是用来快速判断一个元素是否出现在集合里