散列冲突处理：开放定址法

前面我们讲了一些设计散列函数的方法，从前面的[除留余数法](http://www.nowamagic.net/academy/detail/3008040)的例子也可以看出，我们设计得再好的散列函数也不可能完全避免冲突，这就像我们再健康也只能尽量预防疾病，但却无法保证永远不得病一样，既然冲突不能避免，就要考虑如何处理它。

那么当我们在使用散列函数后发现两个关键字key1≠key2，但是却有f(key1) = f(key2)，即有冲突时，怎么办呢？我们可以从生活中找寻思路。

* 试想一下，当你观望很久很久，终于看上一套房打算要买了，正准备下订金，人家告诉你，这房子已经被人买走了，你怎么办？对呀，再找别的房子呗！这其实就是一种处理冲突的方法开放定址法。

**开放定址法**

所谓的[开放定址法](http://www.nowamagic.net/academy/tag/%E5%BC%80%E6%94%BE%E5%AE%9A%E5%9D%80%E6%B3%95)就是一旦发生了冲突，就去寻找下一个空的散列地址，只要散列表足够大，空的散列地址总能找到，并将记录存入。

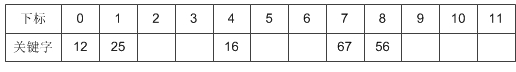
公式为：

fi(key) = (f(key)+di) MOD m (di=1,2,3,......,m-1)

* 用开放定址法解决冲突的做法是：当冲突发生时，使用某种探测技术在散列表中形成一个探测序列。沿此序列逐个单元地查找，直到找到给定的关键字，或者碰到一个开放的地址（即该地址单元为空）为止（若要插入，在探查到开放的地址，则可将待插入的新结点存人该地址单元）。查找时探测到开放的地址则表明表中无待查的关键字，即查找失败。

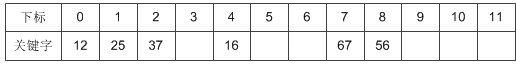
比如说，我们的关键字集合为{12,67,56,16,25,37,22,29,15,47,48,34},表长为12。 我们用散列函数f(key) = key mod l2。

当计算前S个数{12,67,56,16,25}时，都是没有冲突的散列地址，直接存入：

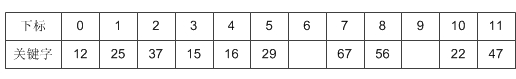


计算key = 37时，发现f(37) = 1，此时就与25所在的位置冲突。

于是我们应用上面的公式f(37) = (f(37)+1) mod 12 = 2。于是将37存入下标为2的位置。这其实就是房子被人买了于是买下一间的作法：。



接下来22,29,15,47都没有冲突，正常的存入：



到了 key=48，我们计算得到f(48) = 0，与12所在的0位置冲突了，不要紧，我们f(48) = (f(48)+1) mod 12 = 1，此时又与25所在的位置冲突。于是f(48) = (f(48)+2) mod 12=2，还是冲突……一直到 f(48) = (f(48)+6) mod 12 = 6时，才有空位，机不可失，赶快存入：

http://www.nowamagic.net/librarys/images/201303/2013_03_13_07.png

我们把这种解决冲突的开放定址法称为[线性探测](http://www.nowamagic.net/academy/tag/%E7%BA%BF%E6%80%A7%E6%8E%A2%E6%B5%8B)法。

从这个例子我们也看到，我们在解决冲突的时候，还会碰到如48和37这种本来都不是同义词却需要争夺一个地址的情况，我们称这种现象为堆积。很显然，堆积的出现，使得我们需要不断处理冲突，无论是存入还是査找效率都会大大降低。

**二次探测法**

考虑深一步，如果发生这样的情况，当最后一个key=34，f(key)=10,与22所在的位置冲突，可是22后面没有空位置了，反而它的前面有一个空位置，尽管可以 不断地求余数后得到结果，但效率很差。

因此我们可以改进di = 12, -12, 22, -22,……, q2, -q2 (q <= m/2),这样就等于是可以双向寻找到可能的空位置。

对于34来说，我 们取di即可找到空位置了。另外增加平方运算的目的是为了不让关键字都聚集在 某一块区域。我们称这种方法为二次探测法。

fi(key) = (f(key)+di) MOD m (di = 12, -12, 22, -22,……, q2, -q2, q <= m/2)

**随机探测法**

还有一种方法是，在冲突时，对于位移量 di 采用随机函数计算得到，我们称之为随机探测法。

此时一定会有人问，既然是随机，那么查找的时候不也随机生成办吗？如何可以获得相同的地址呢？这是个问题。这里的随机其实是伪随机数。

伪随机数是说，如果我们设置随机种子相同，则不断调用随机函数可以生成不会重复的数列，我们在査找时，用同样的随机种子，它每次得到的数列是相同的，相同的 di 当然可以得到相同的散列地址。

fi(key) = (f(key)+di) MOD m (di是一个随机数列)

总之，开放定址法只要在散列表未填满时，总是能找到不发生冲突的地址，是我们常用的解决冲突的办法。