上一篇说到了，设计一个简单、均匀、存储利用率高的散列函数是散列技术中最关键的问题。那么我们今天开始就看看，如何去设计[散列函数](http://www.nowamagic.net/academy/tag/%E6%95%A3%E5%88%97%E5%87%BD%E6%95%B0)。

**散列函数的设计原则**

* 不管做什么事情，要做到最优都不容易，既要付出尽可能的少，又要得到最大化的多。那么什么才算是好的散列函数呢？这里我们有两个原则可以参考。

1. 计算简单

你说设计一个算法可以保证所有的关键字都不会产生冲突，但是这个算法需要很复杂的计算，会耗费很多时间，这对于需要频繁地査找来说，就会大大降低査找的效率了。因此散列函数的计算时间不应该超过其他査找技术与关键字比较的时间。

2. 散列地址分布均匀

我们前面也提到冲突带来的问题，最好的办法就是尽量让散列地址均匀地分布在存储空间中，这样可以保证存储空间的有效利用，并减少为处理冲突而耗费的时间。

* 简单科普一下，以PHP为例。PHP的Hash采用的是目前最为普遍的DJBX33A (Daniel J. Bernstein, Times 33 with Addition)，这个算法被广泛运用与多个软件项目，Apache、Perl和Berkeley DB等。对于字符串而言这是目前所知道的最好的哈希算法，原因在于该算法的速度非常快，而且分类非常好（冲突小，分布均匀）。

下面我们逐个介绍一些常用的散列函数构造方法。估计设计这些方法的前辈们当年可能是从事间谍工作，因为这些方法都是将原来数字按某种规律变成另一个数字而已。首先是直接定址法。

**直接定址法**

如果我们现在要对0-100岁的人口数字统计表，那么我们对年龄这个关键字就可以直接用年龄的数字作为地址。此时f(key) = key。



* 这个时候，我们可以得出这么个哈希函数：f(0) = 0，f(1) = 1，……，f(20) = 20。这个是根据我们自己设定的直接定址来的。人数我们可以不管，我们关心的是如何通过关键字找到地址。

如果我们现在要统计的是80后出生年份的人口数，那么我们对出生年份这个关键字可以用年份减去1980来作为地址。此时f (key) = key-1980。



* 假如今年是2000年，那么1980年出生的人就是20岁了，此时 f(2000) = 2000 - 1980，可以找得到地址20，地址20里保存了数据“人数500万”。

也就是说，我们可以取关键字的某个线性函数值为散列地址，即：

f(key) = a × key + b

这样的散列函数优点就是简单、均匀，也不会产生冲突，但问题是这需要事先知道关键字的分布情况，适合査找表较小且连续的情况。由于这样的限制，在现实应用中，[直接定址法](http://www.nowamagic.net/academy/tag/%E7%9B%B4%E6%8E%A5%E5%AE%9A%E5%9D%80%E6%B3%95)虽然简单，但却并不常用。