https://blog.csdn.net/u012371712/article/details/80040185

http://www.cnblogs.com/lucifer1982/archive/2008/06/18/1224319.html

http://www.cnblogs.com/xingzc/p/5765572.html

https://blog.csdn.net/du_senge/article/details/83305558

哈希表算法原理

哈希表（Hash Table）是一种根据关键字直接访问内存存储位置的数据结构。通过哈希表，数据元素的存放位置和数据元素的关键字之间建立起某种对应关系，建立这种对应关系的函数称为哈希函数。

哈希函数构造方法

哈希表的构造方法是：假设要存储的数据元素个数为n，设置一个长度为m（m≥n）的连续存储单元，分别以每个数据元素的关键字 *Ki(0<= i <=n-1)* 为自变量，通过哈希函数 *hash(Ki)* 把 *Ki* 映射为内存单元的某个地址 *hash(ki)*，并将该数据元素存储在该内存单元中。

一般有以下几种常见的方法：

1）直接定址法

该方法是取关键字的某个线性函数值为哈希地址。可以简单的表示为：

$$hash(K) = aK + C$$

优点是不会产生冲突，但缺点是空间复杂度可能会很高，适用于元素较少的情况下；

2）除留余数法

它是用数据元素关键字除以某个常数所得的余数作为哈希地址，该方法计算简单，适用范围广，是最经常使用的一种哈希函数，可以表示为：

$$hash(K) = K \bmod C$$

该方法的关键是常数的选取，一般要求是接近或等于哈希表本身的长度，理论研究表明，该常数取素数时效果最好。

3）数字分析法

该方法是取数据元素关键字中某些取值较均匀的数字位来作为哈希地址的方法，这样可以尽量避免冲突，但是该方法只适合于所有关键字已知的情况。对于想要设计出更加通用的哈希表并不适用。

1. 哈希冲突解决办法

在构造哈希表时，存在这样的问题，对于两个不同的关键字，通过我们的哈希函数计算哈希地址时却得到了相同的哈希地址，我们将这种现象称为哈希冲突

哈希冲突主要与两个因素相关：

第一，填装因子，所谓的填装因子是指哈希表中已存入的数据元素个数与哈希地址空间大小的比值，即α=n/m， α越小，冲突的可能性就越小，相反则冲突可能性越大；但是α越小，哈希表的存储空间利用率也就很低， α越大，存储空间的利用率也就越高，为了兼顾哈希冲突和存储空间利用率，通常将α控制在0.6-0.9之间（JDK中取0.75），而.NET中的Hashtable则直接将α的最大值定义为0.72（注：虽然微软官方MSDN中声明Hashtable默认填装因子为1.0，事实上所有的填装因子都为0.72的倍数）；

第二，与所用的哈希函数有关，如果哈希函数选择得当，就可以使哈希地址尽可能的均匀分布在哈希地址空间上，从而减少冲突的产生，但一个良好的哈希函数的得来很大程度上取决于大量的实践，不过幸好前人已经总结实践了很多高效的哈希函数，可以参考园子里大牛Lucifer的文章：数据结构 ： Hash Table [I]