* **什么是Hash**

Hash算法，简称**散列算法**，也成哈希算法（英译），是**将一个大文件映射成一个小串字符**。与指纹一样，就是**以较短的信息来保证文件的唯一性**的标志，这种标志与文件的每一个字节都相关，而且难以找到逆向规律。

举个列子：

服务器存了10个文本文件，你现在想判断一个新的文本文件和那10个文件有没有一个是一样的。你不可能去比对每个文本里面的每个字节，很有可能，两个文本文件都是5000个字节，但是只有最后一位有所不同，但这样的，你前面4999位的比较就是毫无意义。那一个解决办法，就是在存储那10个文本文件的时候，都将每个文件映射成一个hash字符串。服务器只需要存储10个hash字符串，在判断的时候，只需要判断新的这个文本文件的hash值是否和那10个文件的hash值一致，那就可以解决这个问题了。

* 简单点说，hash就是将任意长度的消息压缩成某一固定长度的消息摘要的函数。
* 由于文件是无限的，而映射后的字符串能表示的位数是有限的。因此可能会存在不同的key对应相同的Hash值。这就是存在碰撞的可能。
* Hash算法是不可逆的，即不同通过Hash值逆向推出key的值。

**应用**

* **Hash在数据结构中的应用**
* java的数据结构中，很多类都有用到hash算法，比如String，HashMap。
* String中的hashCode()方法

public int hashCode() {  
 int h = hash;  
 if (h == 0 && value.length > 0) {  
 char val[] = value;  
  
 for (int i = 0; i < value.length; i++) {  
 h = 31 \* h + val[i];  
 }  
 hash = h;  
 }  
 return h;  
}

hash的默认值为0，对String当做一个数组，这个**字符数组中的每个字符都转为int**，**加上h乘以31，即是hash值**。从中可以看出，hash算法返回的是int型的数值。

private int hash; // Default to 0

那这里为什么选择31来作为乘级因子？

1. 选择数字31是因为它是一个奇质数，如果选择一个偶数，**因为乘二相当于左移一位，可能会产生溢出，导致数值信息溢出**。
2. 这一点的优势并不明显，但这是一个传统（选择质数）。
3. 同时，数字31有一个很好的特性，乘法运算可以被移位和减法运算来取代，来获取更好的性能，而且这一点可以由jvm来自动完成。
4. 31\*i=(i<<5)-i

* HashMap中hash值存在的目的是加速键值对的查找，key的作用是为了将元素适当的放在各个桶里，对于抗碰撞的要求没那么高。
* 对key的hash计算，就是计算出key的hash值，并移动到低位，完成高低位的融合。
* Hash在密码学中的应用
* hash算法在密码学中主要用于消息摘要和签名。换句话说，主要是对整个消息的完整性进行校验。
* **解决碰撞的方法**

通过构造性能良好的hash算法，可以减少冲突，但不能完全避免冲突。而且在创建hash表和查找hash表都有可能会遇到冲突。因为作为一种可用的hash算法，其位数一定是有限的，但是它所能记录的文件又是无限的。所以碰到碰撞的概率永远不会是零。

常见的解决方法主要有**开放寻址法**，**再哈希法**，**链地址法**，**建立公共溢出区**

* 开放寻址法
  + 主要是通过key转出hash值h1，如果遇到冲突，那在h1的基础上，再次进行hash操作得到h2，直到不产生冲突为止。
* 再哈希法
  + 主要是同时构造多个hash函数，当第一个hash函数计算出来的遇到冲突时，调用第二个hash函数计算出来的值。以此类推，直到不产生冲突为止。
* 链地址法
  + 主要将所有hash值（比如都为i）相同的元素构成一个所谓同义词链的单链表，将单链表的头指针存放在hash表的第i个单元中。
* 建立公共溢出区
  + 主要将hash表分为基本表和溢出表，凡是和基本表相冲突的，都放在溢出表中。
* **常见的Hash算法**

主要的Hash实现主要有一下几类，其中MD5和SHA-1是应用最为广泛的Hash算法。

* MD4
  + MD4是MIT的Rivest在1990年设计，MD是信息摘要Message Digest的缩写。它是基于32位操作数的位操作来实现的。
* MD5
  + MD5是Rivest在1991年对MD4的改进，MD5比MD4来得复杂，因此速度慢一些，但安全性更好。
* SHA-1
  + SHA-1是由NIST NSA设计的，它对长度小于264位的输入，产生长度位160位的散列值。因此抗穷举性更好。SHA-1模仿了MD4的算法。
* 总结
* 在java中array和list是两个极端，array查询快，但是添加删除操作慢；list添加删除快，但查询慢。
* HashMap折中了上面两点，Key的hash用数组，value用列表，如果value有碰撞，就是用链表依次存储。