Hash算法

概述

Hash，一般翻译做散列、杂凑，或音译为哈希，是把任意长度的输入（又叫做预映射pre-image）通过散列算法变换成固定长度的输出，该输出就是散列值。这种转换是一种压缩映射，也就是，散列值的空间通常远小于输入的空间，不同的输入可能会散列成相同的输出，所以不可能从散列值来确定唯一的输入值。

哈希函数简单的说就是一种将任意长度的消息压缩到某一固定长度的消息摘要的函数。

哈希可以被认为是一种高级思想。

简单示例

哈希一般用来快速查找，通过hash函数将输入的键值（key）映射到某一个地址，然后就可以获得该地址的内容。

例如我们想通过一个对7取余的哈希算法，来存储1-7这10个数字，那么可能的结果就会如下显示：

索引	0	1	2	3	4	5	6
数值	7	1	2	3	4	5	6

不过这其中会出现一些问题，最常见的是出现冲突。就是输入不同的key，经过hash之后得到同样的值，也就是在同一个地址要储存不同的data，例如使用上图的hash，输入的key为1和8得到的地址都是1，则这种就是出现了冲突。

解决这种冲突的方法有多种，比如线性探测法，折叠法，链表法等等。

举例，我们可以将存储数值部分的数据结构修改成vector，这样，即使二者计算出来的哈希结果相同，那么也可以存储在不同的位置，如下图所示：

索引	0	1	2	3	4	5	6
数值	7	1	2	3	4	5	6
		8	9				

8和9计算完毕后，得到了与1,2相同的hash值，但是由于vector的存在，使得他们能够被正常的存储。

特性

第一、它具有单向性。

我们只能够把长的数据计算成一个短的哈希，但是我们不可能把哈希再推导出原来的数据，这就是哈希锁定。

第二、哈希具有唯一性。

哈希的唯一性就是我们把一个长的数据变成一个哈希数据，每一个长的数据它变成的哈希数据都是唯一与之对应的，但是这句话是有问题的，因为把长数据变成短数据，或者是把长的文件变成一个短的哈希，那在科学上，它是有一定的概率会形成相同的哈希的，只是概率极低极低，哈希有唯一性就是这么来。

第三点、哈希具有离散性。

离散性就是我们两个非常相近的文件，或者是我们两个只相差一位数的一个长数据，我们计算出来的哈希它的数值是天壤之别，是没有任何相似地方的，哈希的离散性主要是为了规避一些有特征的攻击。

如果们的两个文件相差只有那么一丢丢，如果计算出来的哈希也很相近，它是更容易遭受到攻击的，所以正常的哈希它有这三个特点，哈希它的长度是32个字节，每个字节是八位数，现在通用的哈希都是256位数字，哈希的数值就是0-2的256次方，那2的256次方大概是多大，它比全宇宙的原子数量还要多。

常用HASH函数

1．直接寻址法。取关键字或关键字的某个线性函数值为散列地址。即H(key)=key或H(key) = a • key + b，其中a和b为常数（这种散列函数叫做自身函数）

2．数字分析法。分析一组数据，比如一组员工的出生年月日，这时我们发现出生年月日的前几位数字大体相同，这样的话，出现冲突的几率就会很大，但是我们发现年月日的后几位表示月份和具体日期的数字差别很大，如果用后面的数字来构成散列地址，则冲突的几率会明显降低。因此数字分析法就是找出数字的规律，尽可能利用这些数据来构造冲突几率较低的散列地址。

3．平方取中法。取关键字平方后的中间几位作为散列地址。

4．折叠法。将关键字分割成位数相同的几部分，最后一部分位数可以不同，然后取这几部分的叠加和（去除进位）作为散列地址。

5．随机数法。选择一随机函数，取关键字作为随机函数的种子生成随机值作为散列地址，通常用于关键字长度不同的场合。

6．除留余数法。取关键字被某个不大于散列表表长m的数p除后所得的余数为散列地址。即H(key) = key MOD p,p<=m。不仅可以对关键字直接取模，也可在折叠、平方取中等运算之后取模。对p的选择很重要，一般取素数或m，若p选的不好，容易产生碰撞。

处理冲突方法

开放定址法：当关键字的哈希地址 p=H（key）出现冲突时，以 p 为基础，产生另一个哈希地址 p1，如果 p1 仍然冲突，再以 p 为基础，产生另一个哈希地址 p2，循环此过程直到找出一个不冲突的哈希地址，将相应元素存入其中；

再哈希法：这种方法是同时构造多个不同的哈希函数，当哈希地址 Hi=RH1（key）发生冲突时，再计算 Hi=RH2（key），循环此过程直到找到一个不冲突的哈希地址，这种方法唯一的缺点就是增加了计算时间；

链地址法：这种方法的基本思想是将所有哈希地址为 i 的元素构成一个称为同义词链的单链表，并将单链表的头指针存在哈希表的第 i 个单元中，因而查找、插入和删除主要在同义词链中进行。链地址法适用于经常进行插入和删除的情况；

建立公共溢出区：将哈希表分为基本表和溢出表两部分，凡是和基本表发生冲突的元素，一律填入溢出表。

常用hash算法

（1）MD4

MD4(RFC 1320)是 MIT 的Ronald L. Rivest在 1990 年设计的，MD 是 Message Digest（消息摘要） 的缩写。它适用在32位字长的处理器上用高速软件实现——它是基于 32位操作数的位操作来实现的。

（2）MD5

MD5(RFC 1321)是 Rivest 于1991年对MD4的改进版本。它对输入仍以512位分组，其输出是4个32位字的级联，与 MD4 相同。MD5比MD4来得复杂，并且速度较之要慢一点，但更安全，在抗分析和抗差分方面表现更好。

（3）SHA-1及其他

SHA1是由NIST NSA设计为同DSA一起使用的，它对长度小于264的输入，产生长度为160bit的散列值，因此抗穷举（brute-force）性更好。SHA-1 设计时基于和MD4相同原理,并且模仿了该算法。