# 哈希

所谓哈希，就是对数据按照某种算法编码，变成一个整数，使其作为数组下标，来索引其他值，比如c++里的map,定义map<string,int>ind,令ind[“asd”]=112,意味着可以用字符串”asd”索引112。而哈希算法也叫哈希函数，既然是函数，就要求自变量只能对应一个因变量，“asd”是自变量，它只能对应一个因变量112.

哈希算法是研究对数据编码成哈希值的算法

## 字符串的哈希

一个很朴素的想法就是令哈希值是每一位阿斯克码之和，这样会导致”ab”和”ba”的哈希值相同，这叫做哈希冲突，要处理冲突，这样就浪费了时间。一个好的算法会尽可能减少哈希冲突，显然朴素的想法哈希冲突很多。

下面介绍几种经典的较好的哈希算法：

这里仅仅介绍哈希算法，算出来的数值太大，还不能作为数组下标来映射，顶多完成P3370这道题(给你n个字符串问你有多少个不同的)，筛去重复数据。

此外这里没有对哈希冲突(有的地方叫哈希碰撞)做处理，一旦有冲突，会直接导致答案错误，这个在map实现里说

### BKDR Hash：

它的主要思路是选取恰当的进制，可以把字符串中的字符看成一个大数字中的每一位数字，然后取一个进制Q，和一个模数P,利用同余定理算出一个小于P的数值，选取好的Q和P可以让冲突变少，Q的选取要求比字符集字符数多，p则是一个大数，p最少也要10^9级别，P越小冲突概率越高,P取素数也可以降低冲突概率，和生日悖论有关。此外对于每个字符对应的数字，a-z千万不能让a对应0，这样不能区分ab和b，所以应该a-z对应1-27.或者干脆直接用阿斯克码当中代表的数(阿斯科马的0代表空，不会对应有效字符)。P的选择也有个巧妙方法是直接选择2^64-1这样利用unsigned long long类型，可以自动取模。

### Md5 Hash：

计算机的存储单位为字节，一个字节对应8个二进制位，共可以表示2^8也就是256种状态。若表示数的话，最多只能表示256个数。

如一个字节可以表示非负整数的0~255，而表示更大的数，则需要占用多个字节，如表示256至少需要两个字节。

256的二进制形式为 00000001 00000000。这样在计算机存储上就存在一个问题：是先存储00000001这个字节，还是先存储00000000这个字节呢？实际上，采用这两种存储方式的都有，取决于CPU架构和编译器。这就引出了字节序的概念。

来看” hello”这个字符串，它是阿斯克字符构成，1个字符1位，那么长度是5，意味着由5个01字符串编码而成。每个字符是阿斯克码，展开2进制是这样：

**01101000** 01100101 **01101100** 01101100 **01101111**

**第一步:处理原文**

首先，我们计算出原文的比特长度，

对512求余的结果，如果不等于448，就需要填充原文使得原文对512求余的结果等于448。填充的方法是第一位填充1，其余位填充0。填充完后，信息的长度就是512\*N+448。

之后，用剩余的位置（512-448=64位）记录原文的真正长度，把长度的二进制值补在最后。这样处理后的信息长度就是512\*(N+1)。

## Map的实现

现在介绍map的实现，这是真正意义的哈希表