可参考地址：

http://www.nowamagic.net/academy/detail/3008060

Java中是如何实现计算hash值的？

一、哈希表基本概念

如果在记录的存储位置和其关键字之间建立某种直接关系，那么在进行查找时，就无须作比较或只作很少次的比较就能直接由关键字找到相应的记录。哈希表（Hash table）正是基于这种思想建立的。

哈希表法又叫做杂凑法或散列法。它的基本思想是：设置一个长度为m的表，用一个函数H把数据集合中n个记录的关键字尽可能唯一地转换成0~(m-1)范围内的数值，即对于集合中任意记录的关键字K_i，有：

$$0 \leq H(K_i) \leq m-1 (0 \leq i < n)$$

这样，就可以利用函数H将数据集合中的记录映射到表中，$H(K_i)$便是记录R_i在表中的存储位置。我们称建立表与记录关键字之间映射关系的函数H为哈希函数。

显然，一旦哈希表建立，进行查找时可以用给定的关键字和建立哈希表时所采用的哈希函数直接在给定的表中进行查找。然而，由于集合中各记录关键字的取值可能在一个很大的范围内，所以即便当集合中记录的个数不是很多时，也很难选取一个合适的哈希函数H，保证对于任意不同的K_i和K_j，有$H(K_i) \neq H(K_j)$。我们把$K_i \neq K_j$，而$H(K_i) = H(K_j)$的现象称为冲突现象。

二、构造哈希函数的方法

1. 直接定址法：当关键字是整型数时，可以取关键字本身或它的线性函数作为它的哈希地址。即$H(K) = K$或$H(K) = aK + b(a, b为常数)$。但是该方法会造成空间的大量浪费。
2. 除留余数法：选取一个合适的不大于哈希表长的正整数m，用m去除关键字K，所得的余数作为其哈希地址，即：$H(K) = K mod m$，这种方法称为除留余数法，该方法的优劣取决于m值的选取。若m取某个偶数值，其结果是将奇数关键字的记录映射到奇数地址上，将偶数关键字的记录映射到偶数地址上，因此产生的哈希地址很可能不均匀分布。若m取关键字的基数的幂次值，那么产生的哈希地址是关键字的某几个低位值。这种方法是一种简单而且行之有效的构造哈希函数的方法。
3. 数字分析法：关键字有d位数，选取其中若干位的值构造哈希地址的方法称为数字分析法。这种方法要事先知道所有关键字或大多数关键字的值，对这些关键字的各位值做分析，丢掉不均匀的位值，留下分布较均匀的位值构造器哈希函数。
4. 平方取中法：取关键字平方后的中间若干位作为其哈希地址。即：$H(K) =$" K^2的中间几位 "。因为关键字平方后使得它的中间几位和组成关键字的各位值均有关，从而使哈希地址的分布较均匀，减少了发生冲突的可能性。所取的位数取决于哈希表的表长。
5. 折叠移位法：根据哈希表将关键字分成尽可能等长的若干段，然后将这几段的值相加，并将最高位的进位舍去，所得结果即为其哈希地址。相加时有两种方法：一种是顺折，即把每一段中的各位值对齐相加，称之为移位法；另一种是对折，像这纸条一样，把原来关键字中的数字按照划分的中界向中间段折叠，然后求和，诚挚为折叠法。

三、哈希地址冲突的方法

1. 链地址法（拉链法）：设基本哈希表为CT[m]，将所有具有相同哈希地址的记录放在同一单链表中，哈希表中的第i个元素CT[i]存放哈希表地址为i的记录组成的单链表的头指针。
2. 开放定址法

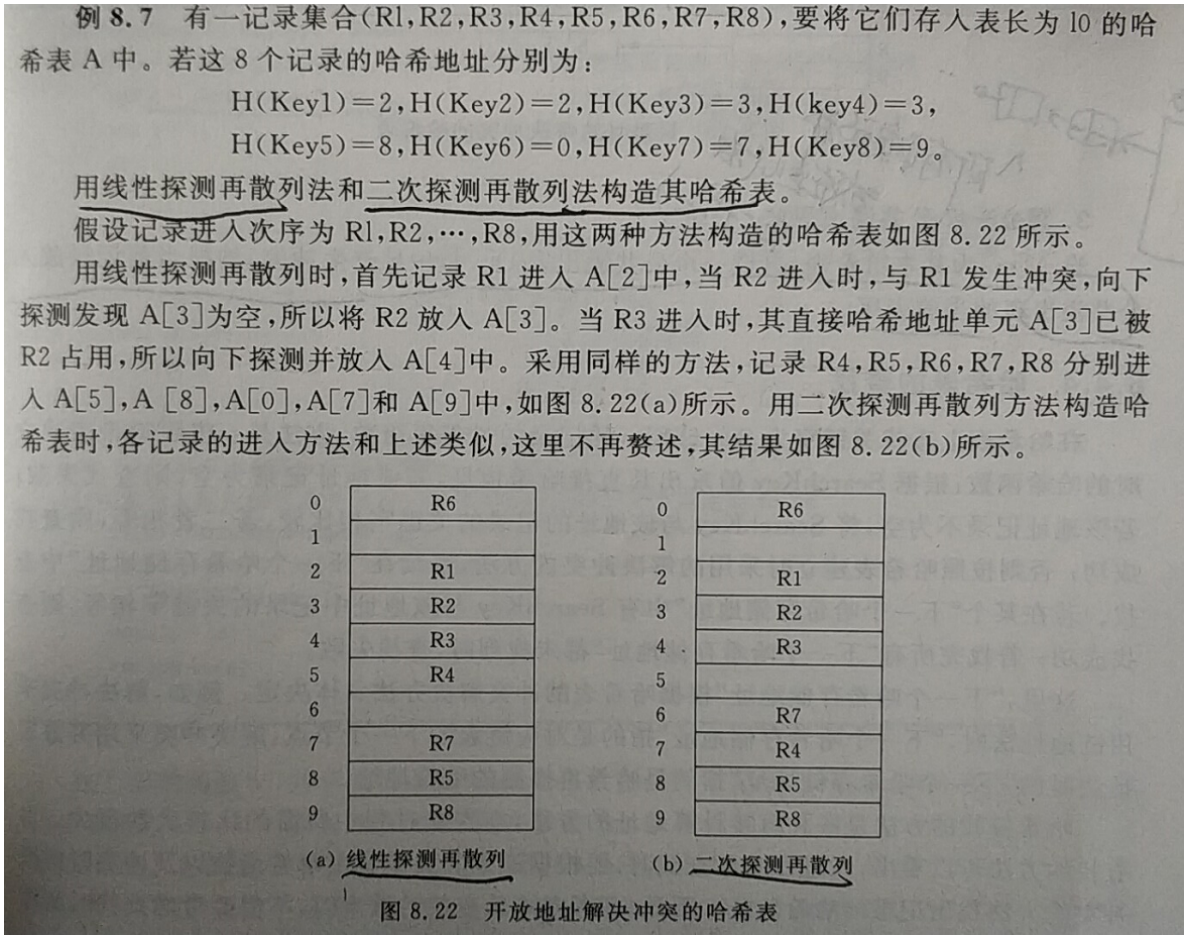
   开放定址法的基本思想是在发生冲突时，按照某种方法继续探测基本表中的其他存储单元，直到找到空位置位置。我们可以用下式描述：

   $$H_i(Key) = (H(Key) + d_i) \quad mod \quad m \quad (i = 1, 2, \ldots, k \quad (k \leq m - 1))$$

   其中，H(Key)为关键字Key的直接哈希地址，m为哈希表长，d_i为每次再探测时的地址增量。用这种方法时，首先计算出它的直接哈希地址H(Key)，若该单元已被其他记录占用，继续查看地址$H(Key) + d_1$的单元，若也被占用，再继续查看地址为$H(Key) + d_2$的单元，如此下去，当发现某个单元为空时，将关键字为Key的记录存放到该单元中。

   增量d_i可以有不同的取法：

   - 线性探测再散列：$d_i = 1, 2, 3, \ldots, m - 1$
   - 二次探测再散列：$d_i = 1^2, -1^2, 2^2, -2^2, \ldots, k^2, -k^2 \quad (k \leq m/2)$
   - 伪随机再散列：$d_i =$ 伪随机序列



例 8.7　有一记录集合(R1,R2,R3,R4,R5,R6,R7,R8)，要将它们存入表长为 10 的哈希表 A 中。若这 8 个记录的哈希地址分别为：

H(Key1)=2,H(Key2)=2,H(Key3)=3,H(key4)=3,
H(Key5)=8,H(Key6)=0,H(Key7)=7,H(Key8)=9。

用线性探测再散列法和二次探测再散列法构造其哈希表。

假设记录进入次序为 R1,R2,…,R8,用这两种方法构造的哈希表如图 8.22 所示。

用线性探测再散列时，首先记录 R1 进入 A[2]中，当 R2 进入时，与 R1 发生冲突，向下探测发现 A[3]为空，所以将 R2 放入 A[3]。当 R3 进入时，其直接哈希地址单元 A[3]已被 R2 占用，所以向下探测并放入 A[4]中。采用同样的方法，记录 R4,R5,R6,R7,R8 分别进入 A[5],A [8],A[0],A[7]和 A[9]中，如图 8.22(a)所示。用二次探测再散列方法构造哈希表时，各记录的进入方法和上述类似，这里不再赘述，其结果如图 8.22(b)所示。

0	R6
1	
2	R1
3	R2
4	R3
5	R4
6	
7	R7
8	R5
9	R8

(a) 线性探测再散列

0	R6
1	
2	R1
3	R2
4	R3
5	
6	R7
7	R4
8	R5
9	R8

(b) 二次探测再散列

图 8.22　开放地址解决冲突的哈希表