散列-hash

散列：一种用于以常数平均时间（）执行插入、删除和查找的技术。

散列无法支持findMin，findMax的操作，也无法按一定的顺序（例如从小到大）来遍历数据表。

散列表：理想情况下，是一个包含一些项的具有固定大小的数组。项是由key-value构成。表的大小为TableSize。

散列函数：将每个键映射到0到TableSize-1的某个数（即数组的下标），并将其放到适当的单元中，这个映射称为散列函数。

在理想情况下，不同的键应该映射到不同的单元，但实际是不可能的，单元的数目TableSize是有限的，而键是用不完的。当不同的键散列到同一个单元时，称为冲突。

散列最重要的工作是寻找一个散列函数，首先要在单元之间均匀的分配键，发生冲突时做什么以及如何确定散列表的大小。

如果能够保证散列表的大小是一个素数（质数），是一个好的办法。

散列其实是一种压缩映射。

装填因子：散列表中元素的个数/散列表的大小

解决散列中的冲突：

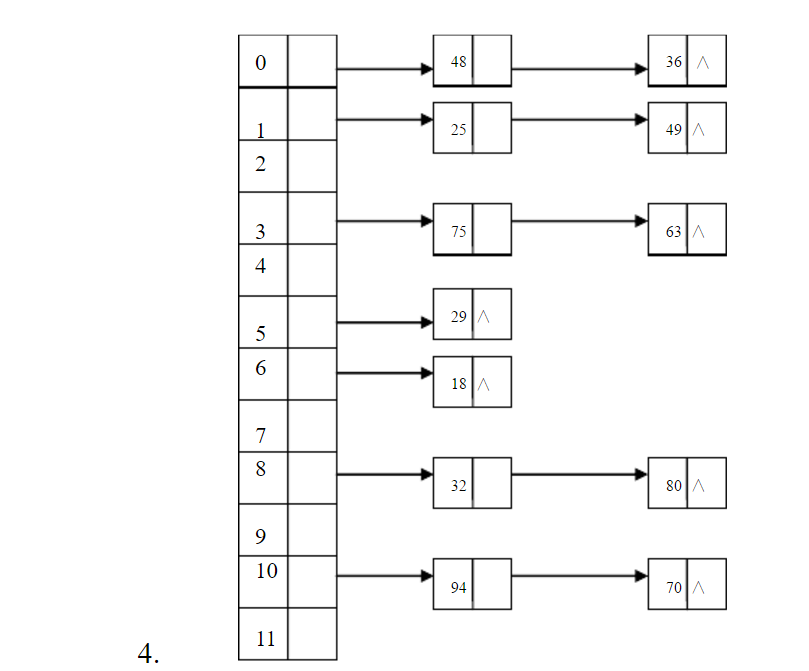
1. 分离链表法：

将散列到同一个值的所有元素保留到一个链表中。执行search时，使用散列函数来确定要遍历哪个链表。

哈希表的平均查找长度：不同的解决冲突的方法得到的成功平均查找长度和不成功平均查找长度的值不同。

等概率下成功平均查找长度：

假定一个待哈希存储的线性表为(32,75,29,63,48,94,25,36,18,70,49,80)，哈希地址空间为HT[12]，若采用除留余数法构造哈希函数（Hash(key) = key MOD 12）和拉链法处理冲突，画出最后得到的哈希表，并求出平均查找长度。



成功平均查找长度：待查找的数字肯定在散列表中。

数字A的查找长度=需要和散列表中的数字的比较次数。

成功平均查找长度

其中为散列表中第i个元素的概率，为找到第i个数据时已比较的次数，为散列表中元素的个数。在等概率下，

在上例的散列表中，找到数据18需要比较1次，查找长度为1，找到数据80需要比较2次，查找长度为2.

平均查找长度=(1\*6 + 2\*5)/12=17/12

不成功平均查找长度：待查找的数据肯定不在散列表中。



其中为散列表中第i个元素的概率，为哈希函数取值为i时，确定查找不成功时的比较次数，为哈希函数的取值个数，在本例中，哈希函数为key MOD 12，取值个数为12.

在上例的散列表中，哈希函数取值为0时，确定查找不成功的比较次数为3(在链表中比较2次后，还要在比较一次才能确定不成功).哈希函数取值为4时，确定查找不成功的比较次数为1.

不成功平均查找长度为=(5\*3 + 2\*2 + 5\*1)/12=2

分离链表法解决冲突的缺点是需要使用链表，如果不使用链表，可以使用探测散列表：当冲突发生时，尝试去选择另外一个单元，直到找的空的单元，将所有的数据都放置到表内。



i指的是冲突发生的次数，i=0为未发生冲突，所以要求，函数为冲突解决函数，并且。

线性探测流行的冲突解决函数为：

探测散列表需要的表的大小要比分离链接表要大，而且装填因子λ一般应小于0.5，便于在发生冲突时探测，如果λ过大，探测就比较困难。

线性探测会引发一次聚焦的问题：散列的单元会形成一些集中的区块，这样即使表相对很空，找到一个自由单元也会花费比较长的时间。

平方探测：冲突函数是二次函数，流行的选择为：

对于平方探测，如果表至少有一半为空，并且表的大小为素数（质数），则总能插入一个新的元素。使用平方探测时，装填比λ最好小于0.5.

再散列：使用分离链表法时，如果装填因子，使用线性探测和平方探测法时，如果装填因子，都需要考虑使用再散列。再散列是创建一个大约两倍大（原表两倍大的第一个素数）的表（而且使用一个相关的新散列函数），扫描整个原始散列表，计算每个（未删除）元素的新散列值并插入到新散列表中。

再散列的运行时间为，所以尽可能的不要经常使用。

编写程序解决<http://kb.cnblogs.com/page/189480/>中的百度面试问题，hash函数使用Blizzard，冲突使用线性探测，hash表为key-value模式，key为检索字符串，value为对应的次数。

例：程序hash\_test1

例：程序hash\_test2

例：程序hash\_test3

例：程序hash\_test4

github上找代码。