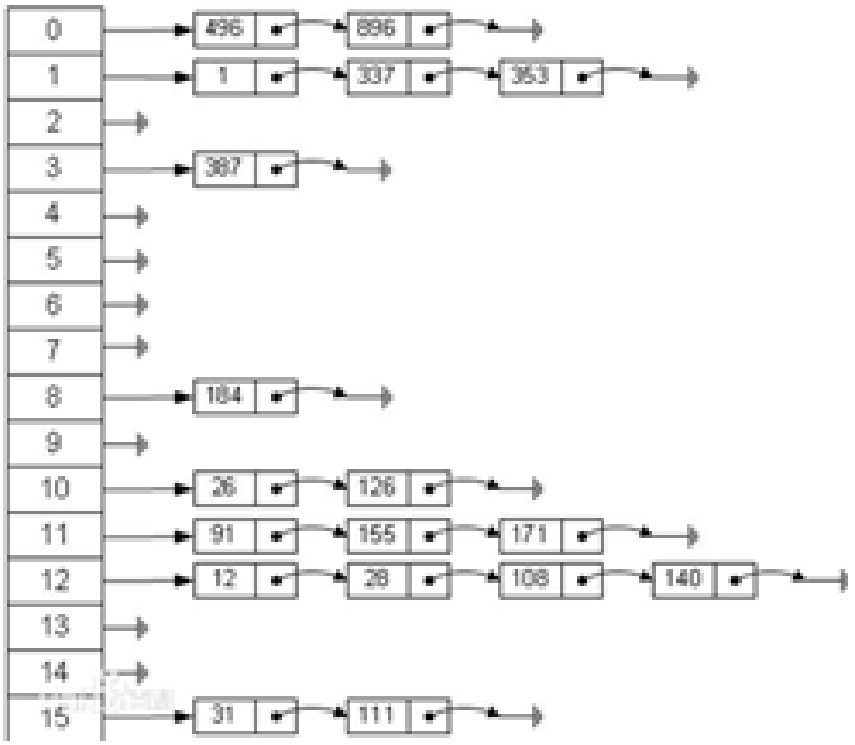
哈希表（Hash Table)

一、基本介绍

散列表（Hash table，也叫哈希表），是根据关键码值(Key value)而直接进行访问的数据结构。也就是说，它通过把关键码值映射到表中一个位置来访问记录，以加快查找的速度。这个映射函数叫做散列函数，存放记录的数组叫做散列表。

二、图解 （链表的数组）



三、比较常见的哈希函数（散列函数）：

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）。若H(key）中已经有值了，就往下一个找，直到H(key）中没有值了，就放进去。

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：当无法确定关键字中哪几位分布较均匀时，可以先求出关键字的平方值，然后按需要取平方值的中间几位作为哈希地址。这是因为：平方后中间几位和关键字中每一位都相关，故不同关键字会以较高的概率产生不同的哈希地址。

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。数位叠加可以有移位叠加和间界叠加两种方法。移位叠加是将分割后的每

一部分的最低位对齐，然后相加；间界叠加是从一端向另一端沿分割界来回折叠，然后对齐相加。

5. 随机数法：选择一随机函数，取关键字的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，容易产生同义词。

然而还有一个比较重要的东西，在放数据的时候，若是根据函数都放到了一个地方了怎么办？这就是处理冲突的方法，当然如果你自己定义的函数足够好，在数据量足够大的情况下还能保证每个都能分配到唯一的位置，那也是可以的，当然这在现实操作中几乎不可能办到。

以下是常见的处理冲突的方法:

1. 开放寻址法：Hi=(H(key) + di) MOD m,i=1,2，…，k(k<=m-1），其中H(key）为散列函数，m为散列表长，di为增量序列，可有下列三种取法：

- 线性探测：di=1,2,3，…，m-1，即在获得的散列地址后增加步长继续探测是否冲突。
- 平方探测：di=1^2,-1^2,2^2,-2^2,3^2，…，±（k)^2,(k<=m/2），即在获得的散列地址前后以一定步长继续探测是否冲突。
- 双散列探测：di=H2(key),2H2(key),3H2(key)，…，kH2(key),H2(key)!=0，其中H2(key)为另一个散列函数，我们可以使H2(key)=m - (key MOD m)，m为散列表长。

2. 再散列法：Hi=R Hi(key),i=1,2，…，k RHi均是不同的散列函数，即在同义词产生地址冲突时计算另一个散列函数地址，直到冲突不再发生，这种方法不易产生“聚集”，但增加了计算时间。

3. 链地址法（拉链法）

在哈希表维护的数组中，每个地址存放的不再是数据，则是一个链表，当冲突发生时，直接将数据放入冲突地址所在的链表中即可。

4. 建立一个公共溢出区

除了建立哈希表必须维护的数组外，在开辟一个数组，专门用来存放发生冲突的数据

四、案例实现

google公司的一个上机题:

有一个公司,当有新的员工来报道时,要求将该员工的信息加入(id,性别,年龄,名字,住址..),当输入该员工的 id时,要求查找到该员工的 所有信息.

要求:

1) 不使用数据库,,速度越快越好=>哈希表(散列)

2) 添加时，保证按照 id从低到高插入 [课后思考：如果 id不是从低到高插入，但要求各条链表仍是从低到高，怎么解决?]

3) 使用链表来实现哈希表, 该链表不带表头[即: 链表的第一个结点就存放雇员信息]

4) 思路分析并画出示意图