数据结构与算法（十）

张铭 主讲

采用教材：张铭，王腾蛟，赵海燕 编写
高等教育出版社，2008. 6（“十一五”国家级规划教材）

http://www.jpk.pku.edu.cn/pkujpk/course/sjjg

第十章　检索

- 10.1 线性表的检索
- 10.2 集合的检索
- 10.3 散列表的检索
- 总结

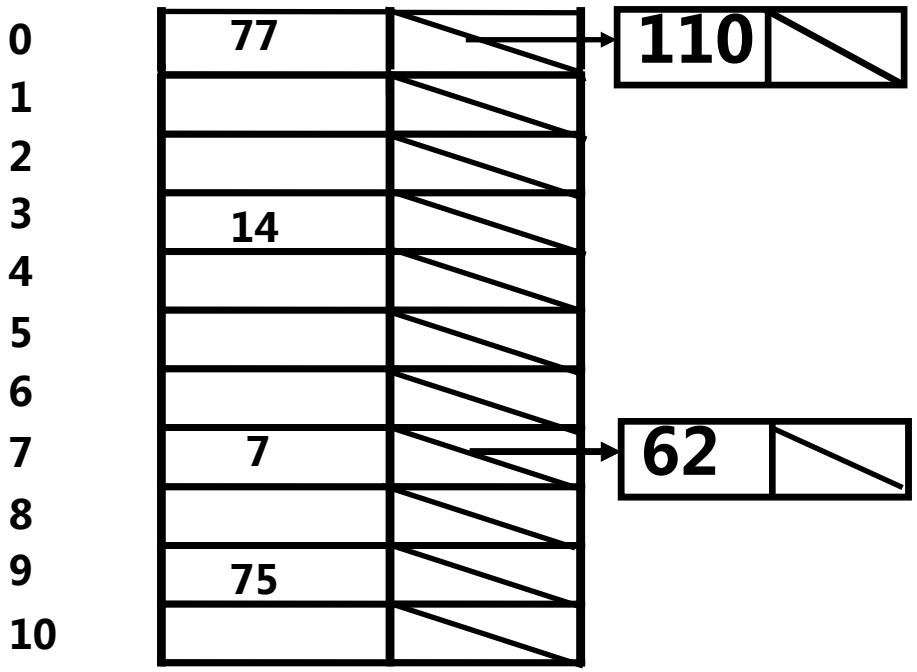
散列检索

- 10.3.0 散列中的基本问题
- 10.3.1 散列函数碰撞的处理
- 10.3.2 开散列方法
- 10.3.3 闭散列方法
- 10.3.4 闭散列表的算法实现
- 10.3.5 散列方法的效率分析

开散列法

{77、14、75、 7、110、62 、95 }

■h(Key) = Key % 11



- 表中的空单元其实应该有特殊值标记出来
  - 例如 -1 或 INFINITY
  - 或者使得散列表中的内容就是指针，空单元则内容为空指针

拉链法性能分析

- 给定一个大小为 M 存储 n 个记录的表
  - 散列函数(在理想情况下)将把记录在表中 M 个位置平均放置，使得平均每一个链表中有 n/M 个记录
  - M>n 时，散列方法的平均代价就是 $\Theta(1)$

10.3.3 闭散列方法

- d_0=h（K）称为 K 的基地址
- 当冲突发生时，使用某种方法为关键码K生成一个散列地址序列

  d_1，d_2，... d_i ，... ， d_{m-1}
  - 所有 d_i (0<i<m) 是后继散列地址
- 形成探查的方法不同，所得到的解决冲突的方法也不同
- 插入和检索函数都假定每个关键码的探查序列中至少有一个存储位置是空的
  - 否则可能会进入一个无限循环中
- 也可以限制探查序列长度

可能产生的问题——聚集

- “聚集”（clustering，或称为“堆积”）
  - 散列地址不同的结点，争夺同一后继散列地址
  - 小的聚集可能汇合成大的聚集
  - 导致很长的探查序列

几种常见的闭散列方法

- 1. 线性探查
- 2. 二次探查
- 3. 伪随机数序列探查
- 4. 双散列探查法

1. 线性探查

- 基本思想：
  - 如果记录的基位置存储位置被占用，那么就在表中下移，直到找到一个空存储位置
    - 依次探查下述地址单元：d+1，d+2，……，M-1，0，1，……，d-1
  - 用于简单线性探查的探查函数是：

$$p(K, i) = i$$

- 线性探查的优点
  - 表中所有的存储位置都可以作为插入新记录的候选位置

散列表示例

- M = 15，h(key) = key%13
- 在理想情况下，表中的每个空槽都应该有相同的机会接收下一个要插入的记录。
  - 下一条记录放在第11个槽中的概率是2/15
  - 放到第7个槽中的概率是11/15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
26	25	41	15	68	44	6				36		38	12	51

改进线性探查

- 每次跳过常数 c 个而不是 1 个槽
  - 探查序列中的第 i 个槽是
    $(\mathrm{h}(K) + ic) \bmod M$
  - 基位置相邻的记录就不会进入同一个探查序列了
- 探查函数是 $\mathrm{p}(K, i) = i*c$
  - *必须使常数 c 与 M 互素*

例：改进线性探查

- 例，$c = 2$，要插入关键码 k_1 和 k_2，$\mathrm{h}(k_1) = 3$，$\mathrm{h}(k_2) = 5$
- 探查序列
  - k_1 的探查序列是3、5、7、9、...
  - k_2 的探查序列就是5、7、9、...
- k_1 和 k_2 的探查序列还是纠缠在一起，从而导致了聚集

2. 二次探查

- 探查增量序列依次为：1^2，-1^2，2^2，-2^2，…，即地址公式是

$$d_{2i-1} = (d + i^2) \% M$$

$$d_{2i} = (d - i^2) \% M$$

- 用于简单线性探查的探查函数是

$$p(K, 2i-1) = i*i$$

$$p(K, 2i) = -i*i$$

例：二次探查

- 例：使用一个大小 M = 13的表
  - 假定对于关键码 k_1 和 k_2，$h(k_1)=3$，$h(k_2)=2$
- 探查序列
  - k_1的探查序列是 3、4、2、7 、...
  - k_2的探查序列是 2、3、1、6 、...
- 尽管 k_2 会把 k_1 的基位置作为第 2 个选择来探查，但是这两个关键码的探查序列此后就立即分开了

3. 伪随机数序列探查

- 探查函数 p(K，i) = perm[i - 1]
  - 这里 perm 是一个长度为 M－1 的数组
  - 包含值从 1 到 M－1 的随机序列

```
// 产生n个数的伪随机排列
void permute(int *array, int n) {
    for (int i = 1;  i <= n;  i ++)
      swap(array[i-1], array[Random(i)]);
}
```

例：伪随机数序列探查

- 例：考虑一个大小为 M = 13的表，perm[0] = 2，perm[1] = 3，perm[2] = 7。
  - 假定两个关键码 k_1 和 k_2，$h(k_1)=4$，$h(k_2)=2$
- 探查序列
  - k_1 的探查序列是 4、6、7、11、...
  - k_2 的探查序列是 2、4、5、9、...
- 尽管 k_2 会把 k_1 的基位置作为第 2 个选择来探查，但是这两个关键码的探查序列此后就立即分开了

二级聚集

- 消除基本聚集
  - 基地址不同的关键码，其探查序列有所重叠
  - 伪随机探查和二次探查可以消除
- 二级聚集 (secondary clustering)
  - 两个关键码散列到同一个基地址，还是得到同样的探查序列，所产生的聚集
  - 原因探查序列只是基地址的函数，而不是原来关键码值的函数
  - 例子：伪随机探查和二次探查

4. 双散列探查法

- 避免二级聚集
  - 探查序列是原来关键码值的函数
  - 而不仅仅是基位置的函数
- 双散列探查法
  - 利用第二个散列函数作为常数
    - $p(K, i) = i * h_2(key)$
  - 探查序列函数
    - $d = h_1(key)$
    - $d_i = (d + i\ h_2(key)) \% M$

双散列探查法的基本思想

- 双散列探查法使用两个散列函数 h_1 和 h_2
- 若在地址 $h_1(key) = d$ 发生冲突，则再计算 $h_2(key)$，得到的探查序列为：

  $(d+h_2(key)) \% M$，$(d+2h_2(key)) \% M$，$(d+3h_2(key)) \% M$，...
- $h_2(key)$ 尽量与 M 互素
  - 使发生冲突的同义词地址均匀地分布在整个表中
  - 否则可能造成同义词地址的循环计算
- 优点：不易产生“聚集”
- 缺点：计算量增大

M 和 h2(k) 选择方法

- 方法1：选择 M 为一个素数，h_2 返回的值在 $1 \le h2(K) \le M-1$ 范围之间
- 方法2：设置 $M = 2^m$，让 h_2 返回一个 1 到 2^m 之间的奇数值
- 方法3：若 M 是素数，$h_1(K) = K \bmod M$
  - $h_2(K) = K \bmod (M-2) + 1$
  - 或者 $h_2(K) = [K / M] \bmod (M-2) + 1$
- 方法4： 若 M 是任意数，$h_1(K) = K \bmod p$，（p 是小于 M 的最大素数）
  - $h_2(K) = K \bmod q + 1$（q 是小于 p 的最大素数）

思考

- 插入同义词时，如何对同义词链进行组织？
- 双散列函数 $h_2(key)$ 与 $h_1(key)$ 有什么关系？

数据结构与算法

谢谢聆听

国家精品课“数据结构与算法”
http://www.jpk.pku.edu.cn/pkujpk/course/sjjg/

张铭，王腾蛟，赵海燕
高等教育出版社，2008. 6。“十一五”国家级规划教材