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

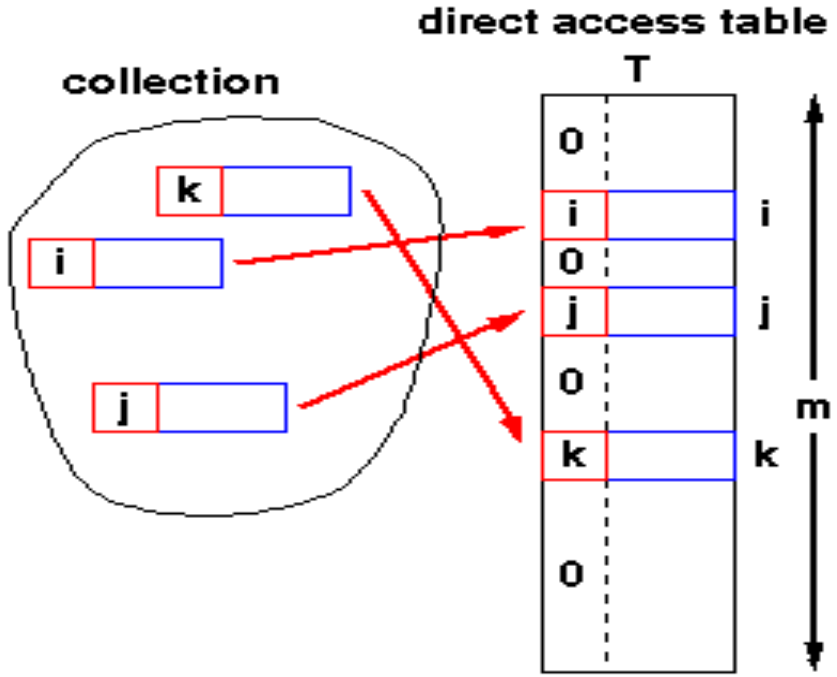
- 10.3.0 散列中的基本问题
- 10.3.1 散列函数碰撞的处理
- 10.3.2 开散列方法
- 10.3.3 闭散列方法
- 10.3.4 闭散列表的算法实现
- 10.3.5 散列方法的效率分析

散列中的基本问题

- 基于关键码比较的检索
  - 顺序检索，==, !=
  - 二分法、树型 >, == , <
- 检索是直接面向用户的操作
- 当问题规模 n 很大时，上述检索的时间效率可能使得用户无法忍受
- 最理想的情况
  - 根据关键码值，直接找到记录的存储地址
  - 不需要把待查关键码与候选记录集合的某些记录进行逐个比较

由数组的直接寻址想到散列

- 例如，读取指定下标的数组元素
  - 受此启发，计算机科学家发明了散列方法（Hash, 有人称“哈希”）
- 一个确定的函数关系 h
  - 以结点的关键码 K 为自变量
  - 函数值 h(K) 作为结点的存储地址
- 检索时也是根据这个函数计算其存储位置
  - 通常散列表的存储空间是一个一维数组
  - 散列地址是数组的下标



例子1

例10.1：已知线性表关键码集合为：

S = { and, array, begin, do, else, end, for, go, if,
repeat, then, until, while, with}

可设散列表为：

char HT2[26][8];

散列函数 H(key) 的值，取为关键码 key 中的第一个字母在字母表 {a, b, c, …, z} 中的序号，即：

H(key) = key[0] – ‘a’

10.3 散列检索

例子1（续）

散列地址	关键码
0	(and, array)
1	begin
2	
3	do
4	(end, else)
5	for
6	go
7	
8	if
9	
10	
11	
12	

散列地址	关键码
13	
14	
15	
16	
17	repeat
18	
19	then
20	until
21	
22	(while, with)
23	
24	
25	

例子2

```
// 散列函数的值为key中首尾字母在字母表中序号的平均值，即：
int H3(char key[])
{
    int i = 0;
    while ((i<8) && (key[i]!='\0'))  i++;
    return((key[0] + key(i-1) – 2*'a') /2 )
}
```

例子2（续）

散列地址	关键码
0	
1	and
2	
3	end
4	else
5	
6	if
7	begin
8	do
9	
10	go
11	for
12	array

散列地址	关键码
13	while
14	with
15	until
16	then
17	
18	repeat
19	
20	
21	
22	
23	
24	
25	

几个重要概念

- 负载因子 $\alpha=n/m$
  - 散列表的空间大小为 m
  - 填入表中的结点数为 n
- 冲突
  - 某个散列函数对于不相等的关键码计算出了相同的散列地址
  - 在实际应用中，不产生冲突的散列函数极少存在
- 同义词
  - 发生冲突的两个关键码

散列函数

- 散列函数：把关键码值映射到存储位置的函数，通常用 h 来表示
- $Address = Hash(key)$
- 散列函数的选取原则
  - 运算尽可能简单
  - 函数的值域必须在表长的范围内
  - 尽可能使得关键码不同时，其散列函数值亦不相同

需要考虑各种因素

- 关键码长度
- 散列表大小
- 关键码分布情况
- 记录的检索频率
- …

常用散列函数选取方法

- 1. 除余法
- 2. 乘余取整法
- 3. 平方取中法
- 4. 数字分析法
- 5. 基数转换法
- 6. 折叠法
- 7. ELFhash 字符串散列函数

1. 除余法

- 除余法：用关键码 x 除以 M (往往取散列表长度)，并取余数作为散列地址。散列函数为：

$$h(x) = x \bmod M$$

- 通常选择一个质数作为 M 值
  - 函数值依赖于自变量 x 的所有位，而不仅仅是最右边 k 个低位
  - 增大了均匀分布的可能性
  - 例如，4093

M 为什么不用偶数

- 若把 M 设置为偶数
  - x 是偶数，h(x) 也是偶数
  - x 是奇数，h(x) 也是奇数
- 缺点：分布不均匀
  - 如果偶数关键码比奇数关键码出现的概率大，那么函数值就不能均匀分布
  - 反之亦然

M 不用幂

$x \bmod 2^8$ 选择最右边 8 位

01100101110**00011010**

- 若把 M 设置为 2 的幂
  - 那么，$h(x) = x \bmod 2^k$ 仅仅是 x (用二进制表示)最右边的k个位(bit)
- 若把 M 设置为 10 的幂
  - 那么，$h(x) = x \bmod 10^k$ 仅仅是 x (用十进制表示)最右边的 k 个十进制位(digital)
- 缺点：散列值不依赖于 x 的全部比特位

除余法面临的问题

- 除余法的潜在缺点
  - 连续的关键码映射成连续的散列值
- 虽然能保证连续的关键码不发生冲突
- 但也意味着要占据连续的数组单元
- 可能导致散列性能的降低

2. 乘余取整法

- 先让关键码 key 乘上一个常数 A $(0<A<1)$，提取乘积的小数部分
- 然后，再用整数 n 乘以这个值，对结果向下取整，把它作为散列地址
- 散列函数为：
  - $hash(key) = \lfloor n * (A * key \% 1) \rfloor$
  - *“$A * key \% 1$”表示取 $A * key$ 小数部分:*
  - $A * key \% 1 = A * key - \lfloor A * key \rfloor$

乘余取整法示例

- 设关键码 key = 123456, n = 10000且取
  $A = (\sqrt{5}-1)/2 = 0.6180339$,
- 因此有

$$hash(123456) =$$
$$= \lfloor 10000*(0.6180339*123456 \ \% \ 1) \rfloor =$$
$$= \lfloor 10000 * (76300.0041151... \ \% \ 1) \rfloor =$$
$$= \lfloor 10000 * 0.0041151... \rfloor = 41$$

乘余取整法参数取值的考虑

- 若地址空间为 p 位，就取 $n = 2^p$
  - 所求出的散列地址正好是计算出来的
  - A * key % 1 = A * key - ⌊ A * key ⌋值的小数点后最左 p 位 (bit) 值
  - 此方法的优点:对 n 的选择无关紧要
- Knuth认为：A 可以取任何值，与待排序的数据特征有关。一般情况下取黄金分割最理想

3. 平方取中法

- 此时可采用平方取中法：先通过求关键码的平方来扩大差别，再取其中的几位或其组合作为散列地址
- 例如，
  - 一组二进制关键码：(00000100，00000110，000001010，000001001，000000111)
  - 平方结果为：(00010000，00100100，01100010，01010001，00110001)
  - 若表长为 4 个二进制位，则可取中间四位作为散列地址：(0100，1001，1000，0100，1100)

4. 数字分析法

- 设有 n 个 d 位数，每一位可能有 r 种不同的符号
- 这 r 种不同的符号在各位上出现的频率不一定相同
  - 可能在某些位上分布均匀些，每种符号出现的几率均等
  - 在某些位上分布不均匀，只有某几种符号经常出现
- 可根据散列表的大小，选取其中各种符号分布均匀的若干位作为散列地址

数字分析法（续1）

- 各位数字中符号分布的均匀度 λ_k

$$\lambda_k = \sum_{i=1}^{r} (\alpha_i^k - n/r)^2$$

  - α_i^k 表示第 i 个符号在第 k 位上出现的次数
  - n/r 表示各种符号在 n 个数中均匀出现的期望值
- λ_k 值越小，表明在该位 (第 k 位)各种符号分布得越均匀

数字分析法（续2）

- 若散列表地址范围有 3 位数字, 取各关键码的 ④ ⑤ ⑥ 位做为记录的散列地址
- 也可以把第 ①，②，③ 和第 ⑤ 位相加，舍去进位，变成一位数，与第 ④，⑥ 位合起来作为散列地址。还可以用其它方法

9	9	2	1	4	8
9	9	1	2	6	9
9	9	0	5	2	7
9	9	1	6	3	0
9	9	1	8	0	5
9	9	1	5	5	8
9	9	2	0	4	7
9	9	0	0	0	1
①	②	③	④	⑤	⑥

①位，$\lambda_1 = 57.60$
②位，$\lambda_2 = 57.60$
③位，$\lambda_3 = 17.60$
④位，$\lambda_4 = 5.60$
⑤位，$\lambda_5 = 5.60$
⑥位，$\lambda_6 = 5.60$

数字分析法（续3）

- 数字分析法仅适用于事先明确知道表中所有关键码每一位数值的分布情况
  - 它完全依赖于关键码集合
- 如果换一个关键码集合，选择哪几位数据要重新决定

5. 基数转换法

- 把关键码看成是另一进制上的数后
- 再把它转换成原来进制上的数
- 取其中若干位作为散列地址
- 一般取大于原来基数的数作为转换的基数，并且两个基数要互素

例：基数转换法

- 例如，给定一个十进制数的关键码是$(210485)_{10}$，把它看成以 13 为基数的十三进制数 $(210485)_{13}$ ，再把它转换为十进制
- $(210485)_{13}$

  $= 2\times13^5 + 1\times13^4 + 4\times13^2 + 8\times13 + 5$

  $= (771932)_{10}$
- 假设散列表长度是 10000，则可取低 4 位1932 作为散列地址

6. 折叠法

- 关键码所含的位数很多，采用平方取中法计算太复杂
- 折叠法
  - 将关键码分割成位数相同的几部分（最后一部分的位数可以不同）
  - 然后取这几部分的叠加和（舍去进位）作为散列地址
- 两种叠加方法：
  - 移位叠加 — 把各部分的最后一位对齐相加
  - 分界叠加 — 各部分不折断，沿各部分的分界来回折叠，然后对齐相加，将相加的结果当做散列地址

例：折叠法

- [例10.6] 如果一本书的编号为 04-42-20586-4

5 8 6 4	0 4 4 2 2 0 5 8 6 4
4 2 2 0	0 2 2 4 4 0
+ 0 4	+ 0 4
[1] 0 0 8 8	6 0 9 2
h(key)=0088	h(key)=6092
(a) 移位叠加	(b) 分界叠加

7. ELFhash字符串散列函数

- 用于 UNIX 系统 V4.0 “可执行链接格式”
  (Executable and Linking Format，即ELF)

```
int ELFhash(char* key) {
    unsigned long h = 0;
    while(*key) {
        h = (h << 4) + *key++;
        unsigned long g = h & 0xF0000000L;
        if (g) h ^= g >> 24;
        h &= ~g;
    }
    return h % M;
}
```

ELFhash函数特征

- 长字符串和短字符串都很有效
- 字符串中每个字符都有同样的作用
- 对于散列表中的位置不可能产生不平均的分布

散列函数的应用

- 在实际应用中应根据关键码的特点，选用适当的散列函数
- 有人曾用“轮盘赌”的统计分析方法对它们进行了模拟分析，结论是平方取中法最接近于“随机化”
  - 若关键码不是整数而是字符串时，可以把每个字符串转换成整数，再应用平方取中法

思考

- 采用散列技术时考虑：
  - （1）如何构造(选择)使结点“分布均匀”的散列函数？
  - （2）一旦发生冲突，用什么方法来解决？

- 散列表本身的组织方法

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