《数据结构》

散 列 表

主讲人：陈卫卫

$$\begin{cases} h_0 = hash(x) \\ h_i = (h_0 + d_i) \mod m \quad (i = 1,2,\ldots) \end{cases}$$

明白了散列函数的设计方法，那么，在散列函数的支持下，就可以从无到有地构建散列表，同时也可以在散列表中完成查找运算。

散列表的处理算法

如何在散列表中插入和查找？

- 构造：从空表开始，逐一插入
- 插入：通过计算元素x的散列函数值和解决冲突方法，为x找到一个空单元，存储x
- 查找：通过计算元素x的散列函数值和解决冲突方法，沿着x的插入路径就能找到x

	姓名	其他信息
0		
1	王五	略
2	吴九	略
3	张三	略
4	赵七	略
5	李四	略
6	牛二	略
7	郑甲？	

散列表的插入算法

算法　散列表的插入算法

步骤1）计算散列函数值h=hash(x);

步骤2）if(a[h]==0)

　　{a[h]=x; return 插入成功信息；}

步骤3）if(a[h]==x) return 出错信息; //x已在表中

步骤4）按预定的解决冲突方法，将x插在表中

散列表的查找算法

算法5-1-2 散列表的查找算法

步骤1）计算散列函数值h=hash(x);

步骤2）if(a[h]==x)return h; //查找成功

步骤3）if(a[h]==0)return 返回查找失败信息;

步骤4）按插入时采用的冲突处理方法进行查找。

现在，请思考如何解决散列冲突问题？


PLA UNIVERSITY OF SCIENCE AND TECHNOLOGY
中国人民解放军理工大学

散列表处理冲突的基本思路

第一种思路：把冲突的元素用链表连起来，称为链接法。

第二种思路：当发生冲突时，反复用探测的方法在散列表中寻找下一个结点，称为开放地址法。

这种探测的方法一定要满足某种规律，而且对插入和查找是一致的！

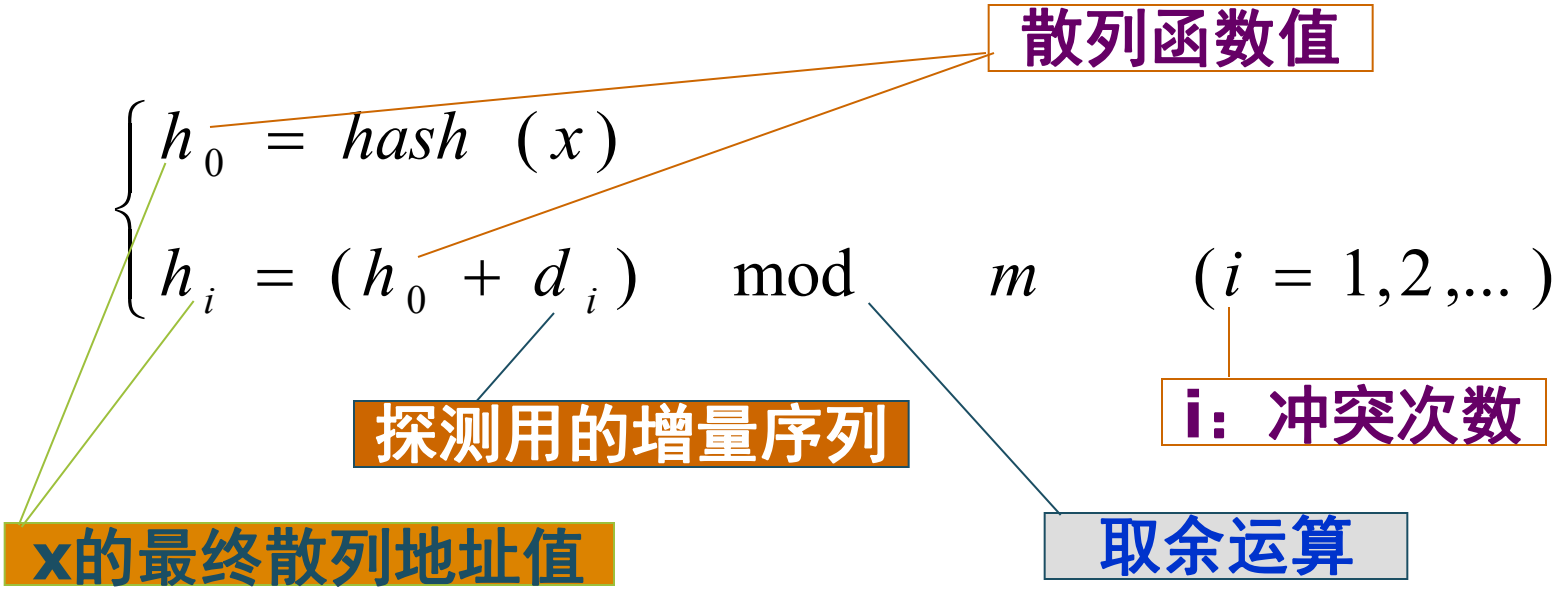
散列表处理冲突的基本思路

第一种思路：把冲突的元素用链表连起来，称为链接法。

第二种思路：当发生冲突时，反复用探测的方法在散列表中寻找下一个结点，称为开放地址法。

这种探测的方法一定要满足某种规律，而且对插入和查找是一致的！

开放地址法的散列地址公式



二次散列性能的好坏，主要取决于增量序列的选取形式

增量序列

- 如何设置探测用的增量序列d_i？

- 3种增量序列：
  - 线性序列——线性探测
  - 平方序列——平方探测
  - 随机序列——随机探测
- 性能各有千秋

线性探测

线性探测：**增量序列是冲突次数i的线性函数**

例如：

散列表a[18]（表长m=18）

散列函数为：hash(x)=x mod 17 （即取余法）

方法一 $d_i=i$ $h_i=(h_0+d_i)\bmod m$

输入序列：73，57，107，92，141，110（开始为空表）

线性探测

线性探测：**增量序列是冲突次数i的线性函数**

$d_i=i$，m=18，p=17

h0=hash(k)=k mod 17

hi=(h0+i) mod 18 （i=1,2,...）

将元素73 、57、107、92、141、110 依次插入散列表。

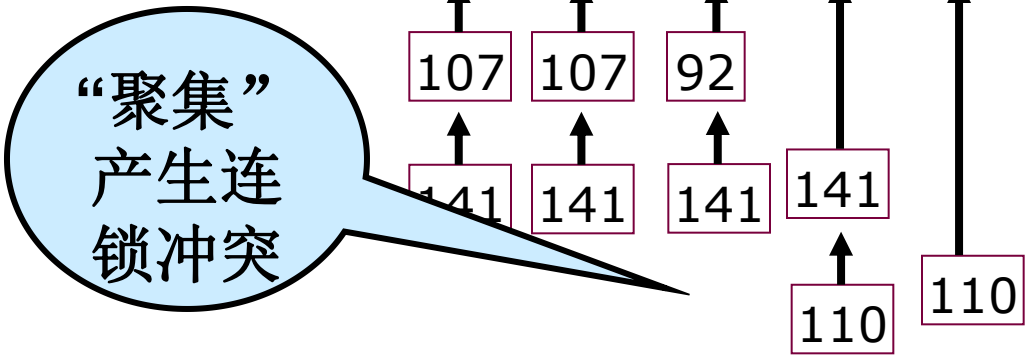
x	73	57	107	92	141	110
h0	5	6	5	7	5	8

线性探测

$d_i=i$，m=18，p=17；h0=hash(k)=k mod 17；hi=(h0+i) mod 18 （i=1,2,...）

x	73	57	107	92	141	110
h0	5	6	5	7	5	8

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
					73	57	107	92	141	110							



线性探测

输入序列：73，57，107，92，141，110

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
					73	57	107	92	141	110							

线性探测的特点

优点：简单，可环视一周寻找空单元

缺点：易产生聚集现象，增加“冲突链”长度

二次散列效果不佳

原因：增量的“步长”为1

改进措施：加大步长

线性探测

练一练：

方法二　$d_i=5i$　　$h_i=(h_0+d_i)\bmod m$

散列表，开始为空 [m=18，　hash(x)=x mod 17]

输入序列：　54，88，42，76，122

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
76			54		122			88					42				

线性探测

练一练：

方法二　$d_i=5i$　$h_i=(h_0+d_i)\bmod m$

散列表，开始为空 [m=18，　hash(x)=x mod 17]

输入序列：　54，88，42，76，122

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
76			54		122			88					42				

特点：只要步长c与m互质，便可环视一周寻找空单元

缺点：表面上不聚集，实际上仍易产生间隔为c的聚集串，
3到8，8到13，13到0，0到5，步长都等于5，
“冲突链”长度会不断增加，二次散列效果也不佳

原因：等步长探测

改进措施：改用变步长探测法

平方探测

方法一　$d_i=i^2$　$h_i=(h_0+d_i)\bmod 18$

散列表，开始为空 [m=18，　hash(x)=x mod 17]

输入序列：57，75，91，108

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
						57	75			91					108		

最终散列存储结果如上所示

下面演示具体构造过程！

平方探测

方法一　$d_i=i^2$　$h_i=(h_0+d_i)\bmod 18$

散列表，开始为空　[m=18，　hash(x)=x mod 17]

输入序列：57，75，91，108

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
						57											

元素x=57

散列函数值hash(x)=6　（57=3×17+6）

插入第一个元素，不冲突

57的散列地址=6，存放在a[6]位置

平方探测

方法一　$d_i=i^2$　$h_i=(h_0+d_i)\bmod 18$

散列表，开始为空　[m=18，　hash(x)=x mod 17]

输入序列：57，75，91，108

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
						57	75										

元素x=75

散列函数值hash(x)=7　（75=4×17+7）

a[7]为空，　75的散列地址=7，存放在a[7]位置

平方探测

方法一　$d_i=i^2$　$h_i=(h_0+d_i)\bmod 18$

散列表，开始为空　[m=18，　hash(x)=x mod 17]

输入序列：57，75，91，108

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
						57	75			91							

元素x=91

散列函数值hash(x)=6　（91=5×17+6）

a[6]不空，冲突1次，加增量$d_1=1$，$h_1=7$

a[7]不空，冲突2次，加增量$d_2=4$，$h_2=10$

a[10]为空，　91的散列地址=10，存放在a[10]位置

平方探测

方法一　$d_i=i^2$　$h_i=(h_0+d_i)\bmod 18$

散列表，开始为空　[m=18，　hash(x)=x mod 17]

输入序列：57，75，91，108

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
						57	75			91					108		

元素x=108

散列函数值hash(x)=6　（108=6×17+6）

a[6]不空，冲突1次，加增量$d_1=1$，$h_1=7$

a[7]不空，冲突2次，加增量$d_2=4$，$h_2=10$

a[10]不空，冲突3次，加增量$d_3=9$，$h_3=15$

a[15]为空，　108的散列地址=15，存放在a[15]位置

平方探测

方法一　$d_i=i^2$　$h_i=(h_0+d_i)\bmod 18$

散列表，开始为空　[m=18，　hash(x)=x mod 17]

输入序列：57，75，91，108

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
						57	75			91					108		

特点：变步长减少了聚集现象

缺点：不易对表环视一周

原因：步长变化太大

改进措施：将步长改为正反平方数，形如：

1^2，-1^2，2^2，-2^2，3^2，-3^2，……

从冲突起点（h_0）向左右两边探测，减少探测死角

平方探测

练一练：

方法二 $d_i=1^2$，-1^2，2^2，-2^2，3^2，-3^2，……

散列表，开始为空 [m=18， hash(x)=x mod 17]

同样的输入序列：57，75，91，108

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
					91	57	75			108							

增量序列选为正负相间的平方数，可以减少探测死角。

随机探测

散列效果很大程度上取决于伪随机函数的性能

伪随机函数的最简单设计方法，采用如下公式：

b：选用一个奇数

p：选用一个较大的素数

$$d_{i+1}=(bd_i+p) \bmod r$$

r：通常取某个2^k

b，p，r是常数

取余运算

用初值d_0代入产生d_1，用d_1代入产生d_2，用d_2代入产生d_3，……

随机探测

伪随机函数的最简单设计方法，采用如下公式：

b：选用一个奇数

p：选用一个较大的素数

$$d_{i+1}=(bd_i+p) \bmod r$$

r：通常取某个2^k

b，p，r是常数

取余运算

用初值d_0代入产生d_1，用d_1代入产生d_2，用d_2代入产生d_3，……

例如，$d_{i+1}=(5d_i+11) \bmod 16$

用$d_0=0$代入，产生伪随机序列：

0，11，2，5，4，15，6，9，8，3，10，13，12，7，14，1，

0（循环）

随机探测

例如，$d_{i+1}=(5d_i+11) \mod 16$

用$d_0=0$代入，产生伪随机序列：0，11，2，5，4，15，6，9，8，3，10，13，12，7，14，1，0（循环）

$$\begin{cases} h_0 = hash\ (x) \\ h_i = (h_0 + d_i) \mod m \qquad (i = 1, 2, \ldots) \end{cases}$$

$$\begin{cases} d_0 = hash\ (x) \\ d_{i+1} = (bd_i + p) \bmod r \qquad (i = 1, 2, \ldots) \end{cases}$$

随机探测

伪随机序列：0，11，2，5，4，15，6，9，8，3，10，13，12，7，14，1，0　　[m=18，hash(x)=x mod 17]

输入序列：60，46，77，128

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	128								60			46					77

散列存储结果如上所示

下面演示具体构造过程！

随机探测

伪随机序列：0，11，2，5，4，15，6，9，8，3，10，13，12，7，14，1，0　　[m=18，hash(x)=x mod 17]

输入序列：60，46，77，128

插入前2个数60和46时，不冲突

hash(60)=9（60=3×17+9 ）

hash(46)=12（46=2×17+12 ）

60和46分别存储在a[9]和a[12]

随机探测

伪随机序列：0，11，2，5，4，15，6，9，8，3，10，13，12，7，14，1，0　　[m=18，hash(x)=x mod 17]

输入序列：60，46，77，128

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
									60			46					77

插入77时，hash(77)=9（77=4×17+9 ）

a[9]不空，冲突1次，因d_0=hash(77)=9，

得d_1=8，加增量8，h_1=17

随机探测

伪随机序列：0，11，2，5，4，15，6，9，8，3，10，13，12，7，14，1，0　　[m=18，hash(x)=x mod 17]

输入序列：60，46，77，128

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
128									60			46					77

插入128时，hash(128)=9（128=7×17+9 ）

a[9]不空，冲突1次，因d_0=hash(77)=9，得d_1=8，加增量8，得h_1=17

a[17]不空，冲突2次，加增量3，得h_2=12；

a[12]不空，冲突3次，加增量10，得h_3=1；

a[1]为空， 128的散列地址=1，存放在a[1]位置

删除

散列表的删除算法（与解决冲突的方式有关）

1. 使用链接法解决冲突时，删除算法与一般链表删除相似。

2. 用开放地址法解决冲突时，删除算法较难。

（1）不能简单地置删除节点为空，这样会切断探测“链”。

（2）可以置删除节点为“已删除标记”，查找时跳过这些节点，在插入时，这些节点可以重新使用。

（3）定期对整个表重新散列.

删除

散列表的删除算法（与解决冲突的方式有关）

1. 使用链接法解决冲突时，删除算法与一般链表删除相似。

2. 用开放地址法解决冲突时，删除算法较难。

（1）不能简单地置删除节点为空，这样会切断探测“链”。

（2）可以置删除节点为“已删除标记”，查找时跳过这些节点，在插入时，这些节点可以重新使用。

（3）定期对整个表重新散列.

小结

1. 复述散列表的作用和意义

2. 学会散列函数的设计方法

3. 编程实现散列表的构造、插入和查找

4. 能够选择并会使用开放地址法解决散列冲突

1. 在什么场合适宜选择散列表结构？（考虑问题的规模、问题所涉及的运算）
2. 散列表的性能如何度量？