查　找

大连理工大学

刘 馨 月

主要内容

- 静态查找
- 动态查找
- 散列

散　列

散列

- 散列函数
- 散列冲突

散列冲突

对于不同的关键字可能得到同一哈希地址，即 $key_1 \neq key_2$，而 $f(key_1) = f(key_2)$，这种现象称为散列冲突（哈希冲突）。

造成原因：

1. 主观设计不当

2. 客观存在

散列冲突

例，除留余数法中

关键字	28	35	63	77	105
$p = 21$					
哈希地址	**7**	**14**	**0**	**14**	**0**

散列冲突

A. 主观设计不当

提高素质

因地制宜

B. 客观存在

如何设计都不可能完全避免冲突的出现？

哈希地址是有限的，而记录是无限的。

散列冲突解决方法

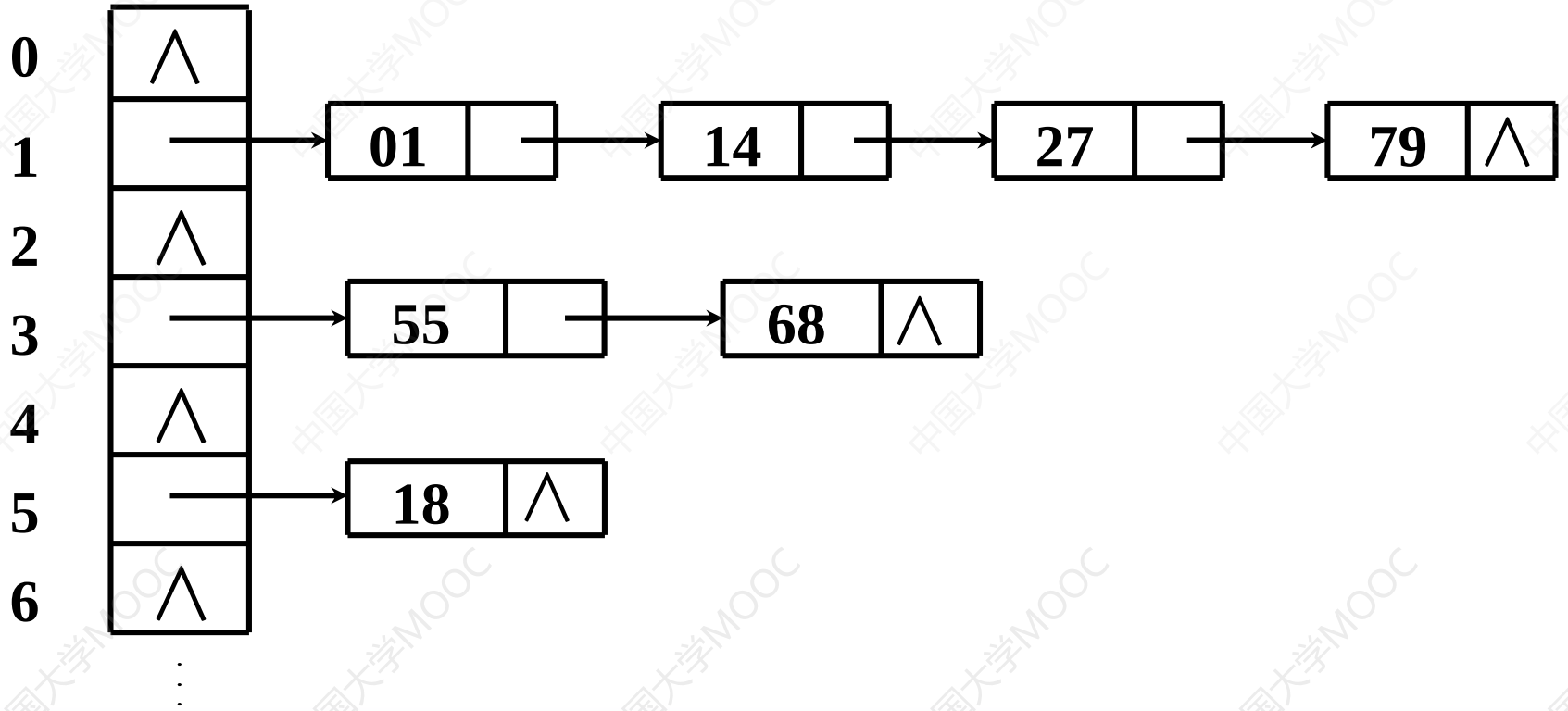
- 开散列方法（拉链法）
- 闭散列方法（开地址法）

开散列方法（拉链法）

将具有同一散列地址的记录存储在一条线性链表中，称为同义词链表。

例，除留余数法中，设关键字为(18 ,14 , 01 , 68 , 27 , 55 , 79)

TSize = 13　散列地址为　（5 ， 1 ， 1 ， 3 ， 1 ， 3 ， 1 ）



闭散列方法（开地址法）

如果h(k)已经被占用，按如下序列探查：

(h(k)+p(1))%TSize, (h(k)+p(2))%TSize, …,(h(k)+p(i))%TSize,…

h(k)　　哈希函数

TSize　哈希表长

p　　探查函数 .

在 **h(k)+p(i-1))%TSize**的基础上，若发现冲突，则使用增量 **p(i)** 进行新的探测，直至无冲突出现为止。

散列冲突

闭散列方法（开地址法）

- 线性探查法
- 二次探查法
- 随机探查法
- 双散列函数法

线性探查法 (Linear Probing)

- 需要搜索或加入一个表项时，使用散列函数计算桶号（即地址）：

  $H_0 = hash(key)$

- 一旦发生冲突，在表中顺次向后寻找“下一个”空桶 H_i 的递推公式为：

  $H_i = (H_{i-1} + 1) \% m, \quad i = 1, 2, \ldots, m-1$

即用以下的线性探查序列在表中寻找“下一个”空桶的桶号：

$H_0+1, H_0+2, \ldots, m-1, 0, 1, 2, \ldots, H_0-1$

当发生冲突时，探查下一个桶。当循环 $m-1$ 次后就会回到开始探查时的位置，说明待查关键码不在表内，而且表已满，不能再插入新关键码。

线性探查法 (Linear Probing)

假设给出一组表项，它们的关键码为 **Burke，Ekers，Broad，Blum，Attlee，Alton，Hecht，Ederly**。采用的散列函数是：取其第一个字母在字母表中的位置。

Hash (*x*) = *ord* (*x*) - *ord* (‘*A*’)　　//*ord* ( ) 是求字符内码的函数

线性探查法 (Linear Probing)

可得 *Hash* (Burke) = 1　　*Hash* (Ekers) = 4
Hash (Broad) = 1　　*Hash* (Blum) = 1
Hash (Attlee) = 0　　*Hash* (Hecht) = 7
Hash (Alton) = 0　　*Hash* (Ederly) = 4

- 设散列表为*HT*[26]，*m* = 26。采用线性探查法处理溢出，则上述关键码在散列表中散列位置如图所示。

0	1	2	3	4
Attlee	Burke	Broad	Blum	Ekers
(1)	(1)	(2)	(3)	(1)

5	6	7	8	9
Alton	Ederly	Hecht		
(6)	(3)	(1)		

线性探查法 (Linear Probing)

- 用平均检索长度 (ASL)衡量散列方法的检索性能。
- 根据检索成功与否，它又有检索成功的平均检索长度ASLsucc和检索不成功的平均检索长度ASLunsucc之分。
- 检索成功的平均检索长度 ASLsucc 是指检索到表中已有表项的平均探查次数。它是找到表中各个已有表项的探查次数的平均值。
- 检索不成功的平均检索长度 ASLunsucc 是指在表中检索不到待查的表项，但找到插入位置的平均探查次数。它是表中所有可能散列到的位置上要插入新元素时为找到空桶的探查次数的平均值。

线性探查法 (Linear Probing)

- 在使用线性探查法对示例进行检索时，检索成功的平均检索长度为：

$$ASLsucc = \frac{1}{8}\sum_{i=1}^{8} Ci = \frac{1}{8}(1+1+2+3+1+6+3+1) = \frac{18}{8}.$$

- 检索不成功的平均检索长度为：

$$ASLunsucc = \frac{1}{26}\left(\sum_{i=0}^{7}(9-i)+\sum_{i=8}^{25}1\right) = \frac{9+8+7+6+5+4+3+2+18}{26} = \frac{62}{26}.$$

二次探查法 (Quadratic Probing)

•为改善“堆积”问题，减少为完成搜索所需的平均探查次数，可使用二次探查法。

•通过某一个散列函数对表项的关键码 x 进行计算，得到桶号，它是一个非负整数。

$$H_0 = hash(x)$$

•二次探查法在表中寻找“下一个”空桶的公式为：

$$H_i = (H_0 + i^2) \% m,$$

$$H_i = (H_0 - i^2) \% m, \quad i = 1, 2, \ldots, (m-1)/2$$

•式中的 m 是表的大小，它应是一个值为 4k+3 的质数，其中 k 是一个整数。这样的质数如 3, 7, 11, 19, 23, 31, 43, 59, 127, 251, 503, 1019, …。

二次探查法 (Quadratic Probing)

- 探查序列形如 H_0, H_0+1, H_0-1, H_0+4, H_0-4, …。
- 在做 (H_0 - i^2) % m 的运算时，当 H_0 - i^2 < 0 时，运算结果也是负数。实际算式可改为

  j = (H_0 - i^2) % m, while (j < 0) j += m

- 示例：给出一组关键码 { Burke, Ekers, Broad, Blum, Attlee, Alton, Hecht, Ederly }。

散列函数为：Hash (x)＝ord (x)－ord ('A')

用它计算可得

Hash (Burke) = 1　　Hash (Ekers) = 4

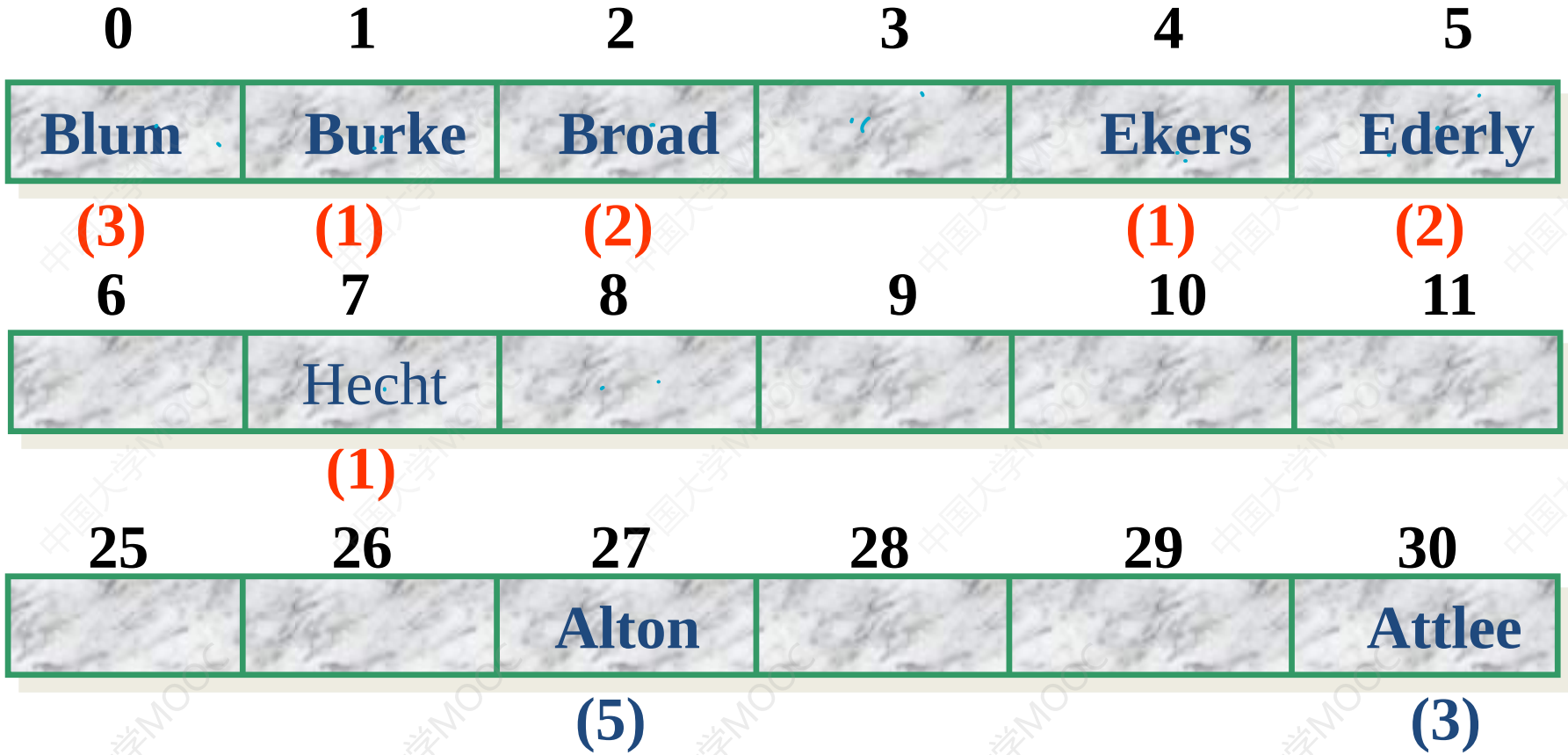
Hash (Broad) = 1　　Hash (Blum) = 1

Hash (Attlee) = 0　　Hash (Hecht) = 7

Hash (Alton) = 0　　Hash (Ederly) = 4

二次探查法 (Quadratic Probing)

- 因为可能桶号是 0 ~ 25, 取满足 4k+3 的质数，表的长度为 TableSize = 31，利用二次探查法得到的散列结果如图所示。



二次探查法 (Quadratic Probing)

- 使用二次探查法处理冲突时的检索成功的平均检索长度为：

$$ASLsucc = \frac{1}{8}\sum_{i=1}^{8} Ci = \frac{1}{8}(3+1+2+1+2+1+5+3) = \frac{18}{8}.$$

- 检索不成功的平均检索长度为：

$$ASLunsucc = \frac{1}{26}(6+5+2+3+2+2+20) = \frac{40}{26}$$

随机探查法 (伪随机探查)

- 理想的探查函数应当在探查序列中随机地从未访问过的空桶中选择下一个位置，即探查序列应当是散列表位置的一个随机序列。
- 但是，实际上不能随机地从探查序列中选择一个位置，因为在检索关键码的时候不能建立起同样的探查序列。
- 例如：

  M=13，伪随机数组前三个值为r_1=2， r_2=3， r_3=7。

  假定有两个关键码k_1和k_2，$h(k_1)$=4， $h(k_2)$=2。

  K_1的探查序列是 4，6，7，11；

  K_2的探查序列是 2，4，5，9。

双散列法

- 使用双散列方法时，需要两个散列函数。
- 第一个散列函数 Hash( ) 按表项的关键码 key 计算表项所在的桶号 H_0 = Hash(key)。
- 一旦冲突，利用第二个散列函数 ReHash( ) 计算该表项到达“下一个”桶的移位量。它的取值与 key 的值有关，要求它的取值应当是小于地址空间大小 TableSize，且与 TableSize 互质的正整数。

双散列法

- 若设表的长度为 m = TableSize，则在表中寻找“下一个”桶的公式为：

$j = H_0 = Hash(key)$，　$p = ReHash(key)$；

$j = (j + p) \% m$；

p是小于m且与m互质的整数

- 利用双散列法，按一定的距离，跳跃式地寻找“下一个”桶，减少了“堆积”的机会。
- 双散列法的探查序列也可写成：

$H_i = (H_0 + i * ReHash(key)) \% m$，

$i = 1, 2, \ldots, m-1$

- 最多经过m-1次探查，它会遍历表中所有位置，回到H_0位置。

散列冲突

- 用不同的方法处理冲突时散列表的平均检索长度如图所示。

处理溢出的方法		平均搜索长度 ASL	
		搜索成功 Sn	搜索不成功 Un（登入新记录）
闭散列法	线性探查法	$\frac{1}{2}\left(1+\frac{1}{1-\alpha}\right)$	$\frac{1}{2}\left(1+\frac{1}{(1-\alpha)^2}\right)$
	伪随机探查法 二次探查法 双散列法	$-\left(\frac{1}{\alpha}\right)\log_e(1-\alpha)$	$\frac{1}{1-\alpha}$
链地址法（同义词子表法）		$1+\frac{\alpha}{2}$	$\alpha+e^{-\alpha}\approx\alpha$

查　找

大连理工大学

刘 馨 月