

散列表(哈希)

复习主存的散列表

(1) 参数 B 是桶的数目

(2) 散列函数可以将任意一个键映射成一个 0 到 $B-1$ 的整数，这个整数即是对应的桶号，记作 $h(K)$

(3) 如果记录的查找键为 K ，那么这个记录将存储到桶号为 $h(K)$ 的桶中，这里 h 即是散列（哈希）函数。

散列表

记录如此多，必须放在辅助存储器，这是本课程的关注点，其区别于主存散列表：

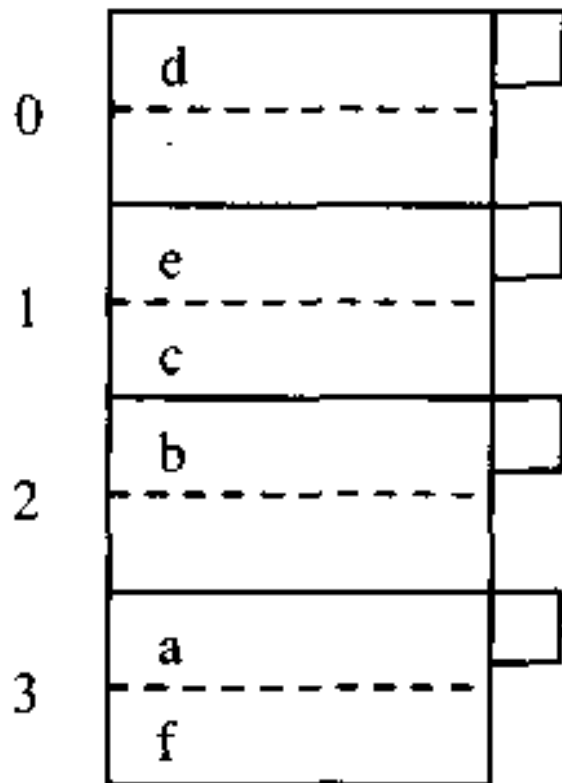
（1）桶数组由**存储块**组成，而不是内存链表。

（2）通过散列函数**h**散列到某个桶中的记录实际存储在存储块上。

（3）如果存储块满了，可以通过溢出块链来解决。

参见：例4.28（pp.125）

注意：要设计机制保证给一个整数*i*,对应的第一个存储块的位置就可以找到



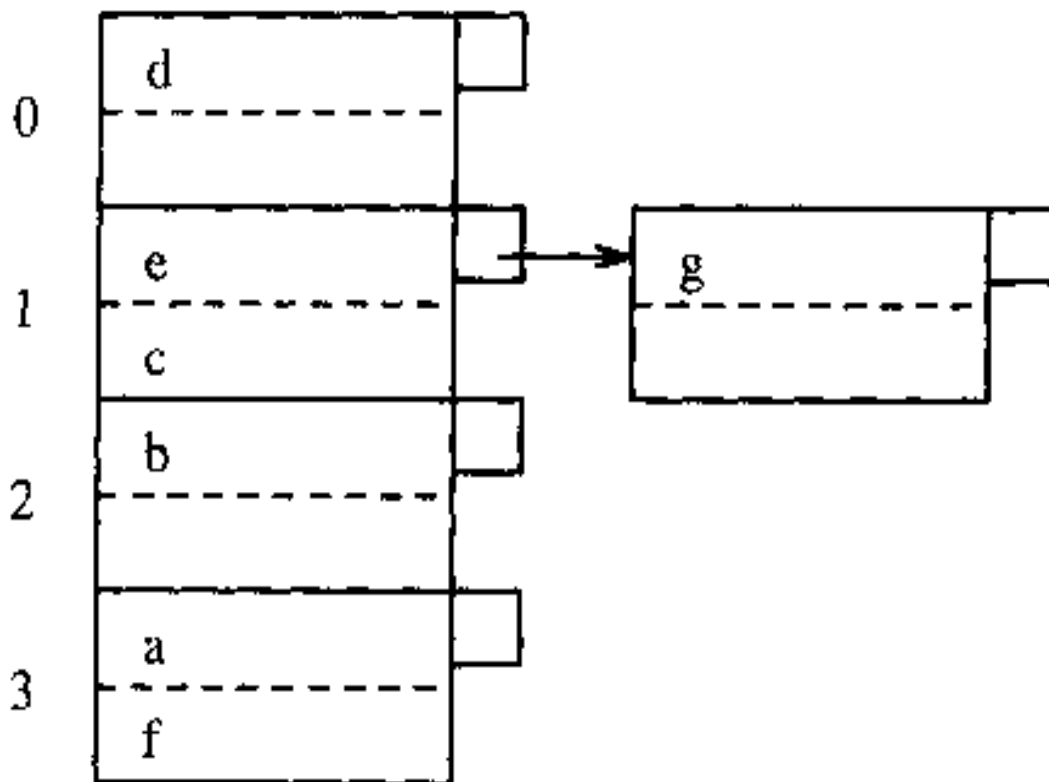
散列表——插入

将查找键为K的记录插入

(1) 首先计算 $h(K)$

(2) 如果 $h(K)$ 还有空间，直接插入；如果没有空间，就增加一个溢出块，并链接到对应的桶。

例4.29(pp.125)



结构示意图

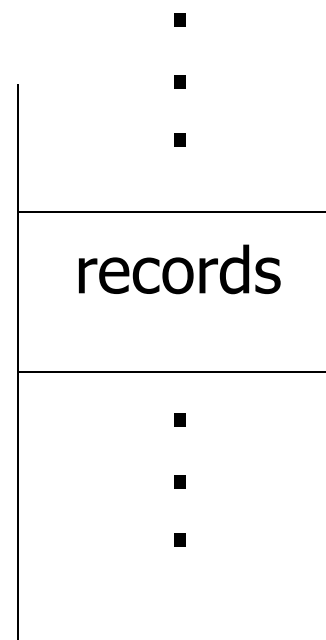
$\text{key} \rightarrow \text{h}(\text{key})$



桶列表
(通常1个桶
对应一个磁盘块)

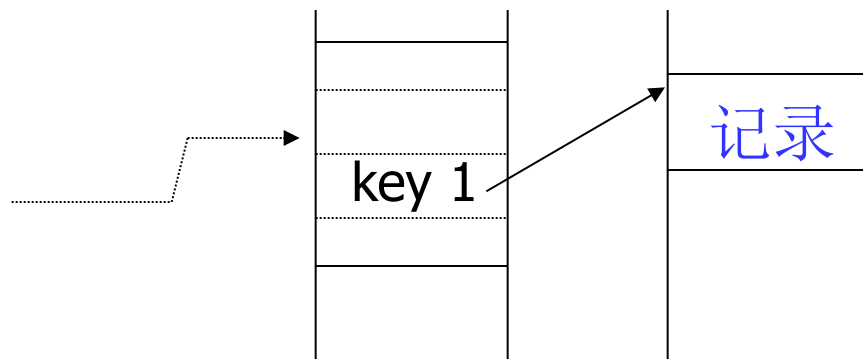
两种实现方式

(1) $\text{key} \rightarrow \text{h}(\text{key})$



两种实现方式

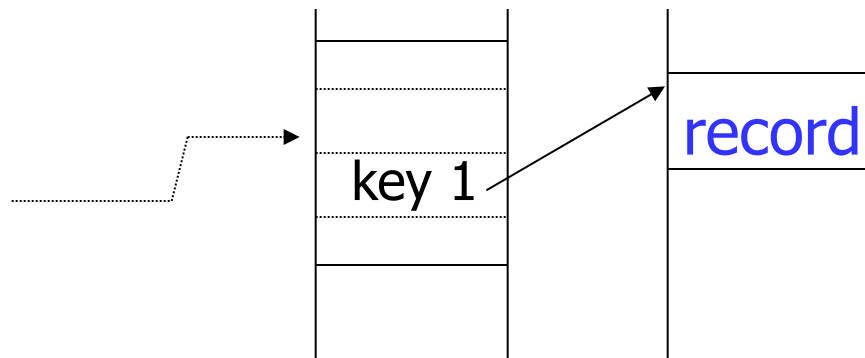
(2) $\text{key} \rightarrow \text{h}(\text{key})$



索引

两种实现方式

(2) $\text{key} \rightarrow \text{h}(\text{key})$



索引

- 第 (2)种是辅助索引的实现方式

哈希函数举例

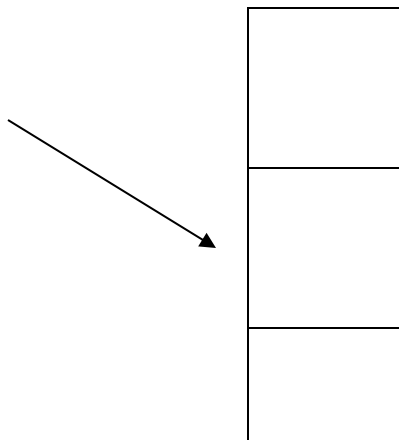
- 键 = ' $x_1 x_2 \dots x_n$ ' n 个字节的字符串
- 有 B 个桶
- 哈希函数 h :
$$(\text{ASC}(x_1) + \text{ASC}(x_2) + \dots + \text{ASC}(x_n)) \text{ modulo } B$$

提示:

- 在一个桶内要保持键排序吗?
- 是的, 如果CPU代价比较珍贵, 并且插入/删除不是很频繁

插入、删除操作

$h(K)$



举例 2条记录/桶

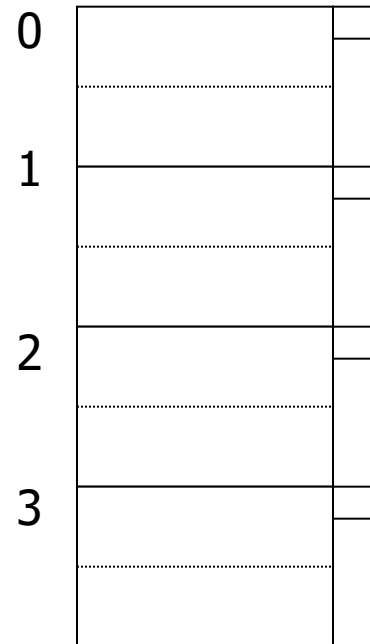
插入:

$$h(a) = 1$$

$$h(b) = 2$$

$$h(c) = 1$$

$$h(d) = 0$$



举例 2条记录/桶

插入:

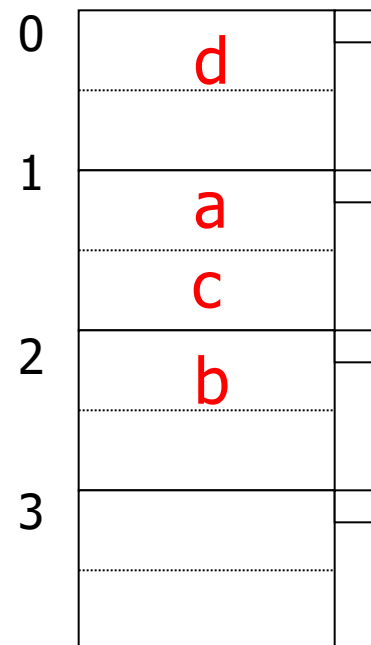
$$h(a) = 1$$

$$h(b) = 2$$

$$h(c) = 1$$

$$h(d) = 0$$

$$h(e) = 1$$



举例 2条记录/桶

插入:

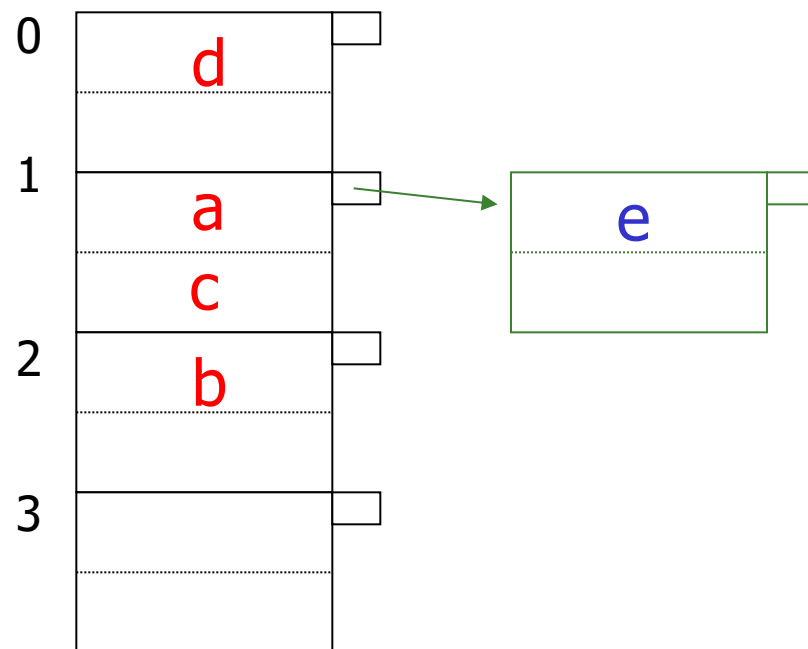
$$h(a) = 1$$

$$h(b) = 2$$

$$h(c) = 1$$

$$h(d) = 0$$

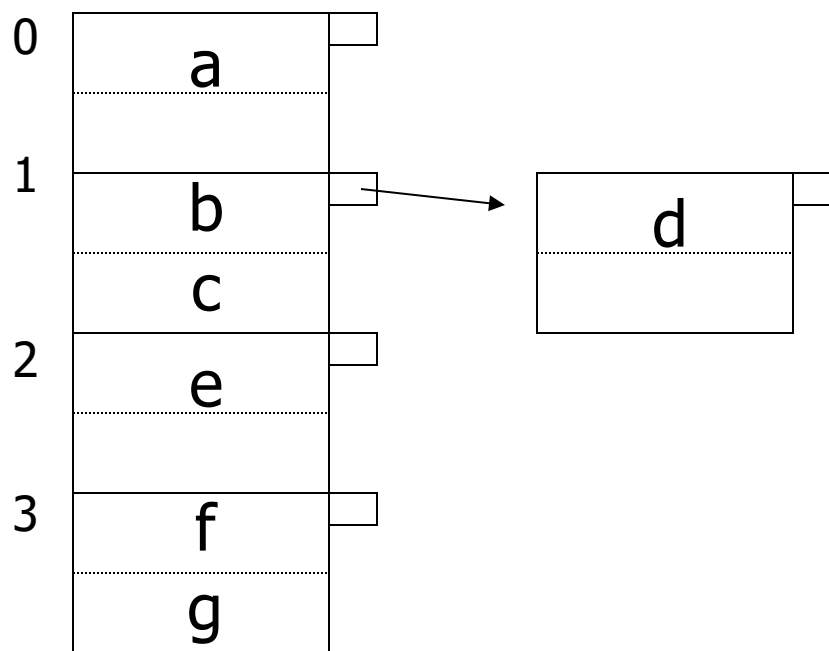
$$h(e) = 1$$



举例：删除

删除：

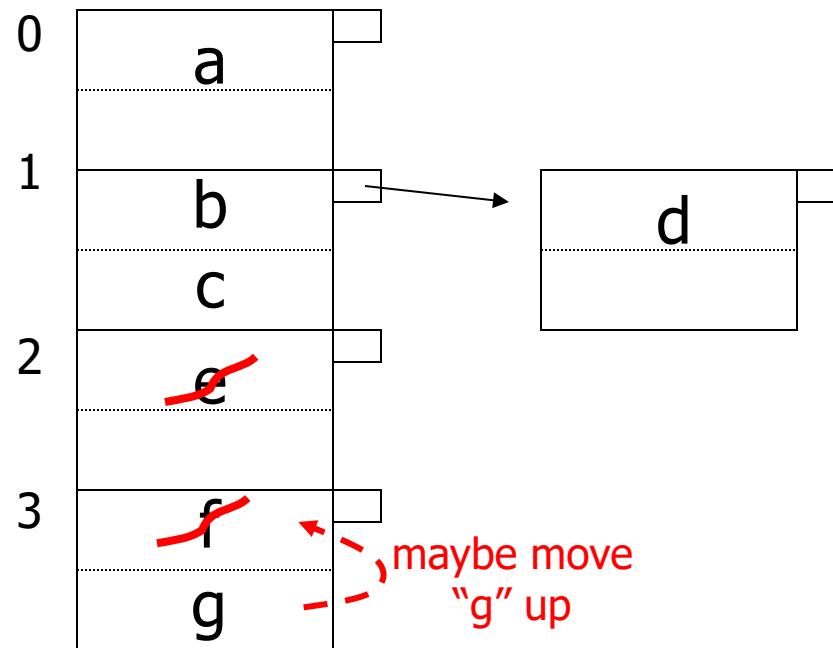
e
f



举例：删除

删除：

e
f
c



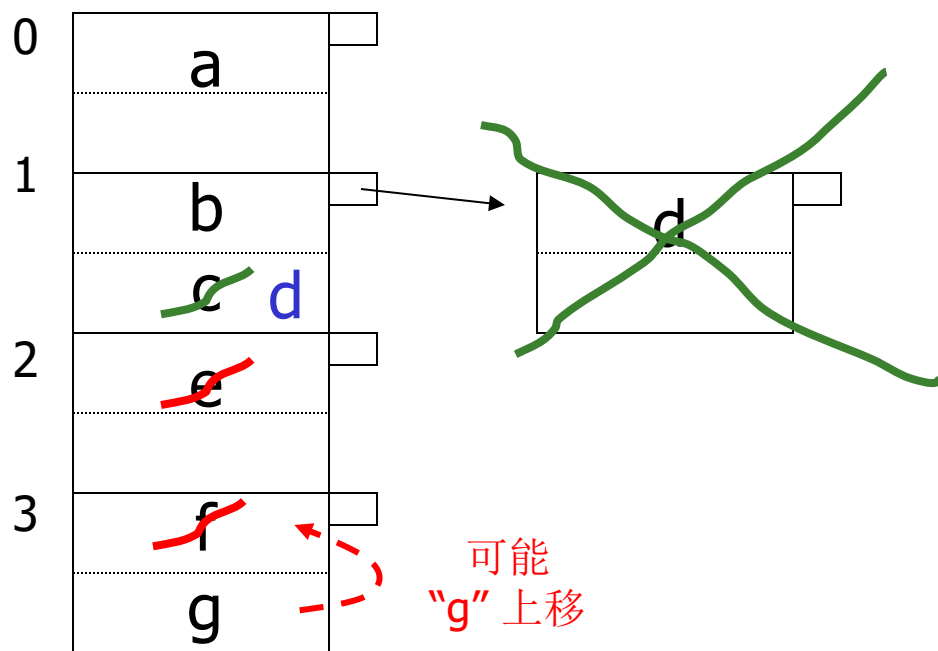
举例：删除

删除：

e

f

c



经验规则:

- 通常把每个桶的空间利用率保持在50%到 80%之间

$$\text{利用率} = \frac{\text{已有的键数量}}{\text{可以存储的键数目}}$$

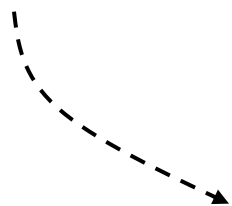
- 如果 < 50%, 浪费空间
- 如果 > 80%, 溢出显著, 它主要依赖哈希函数的选择 (因此哈希函数是有学问的!)

如何处理增长?

- 溢出和重组
- 动态散列

如何处理增长?

- 溢出 和 重组
- 动态散列



- 可扩展的散列
- 线性散列

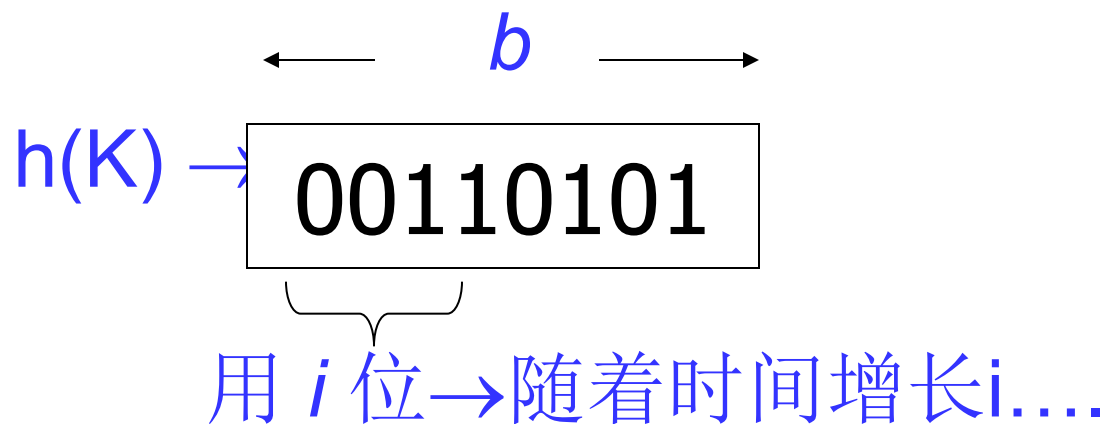
如何处理增长?

(1) 可扩展散列表 (**B**太小时, 即将其翻倍)

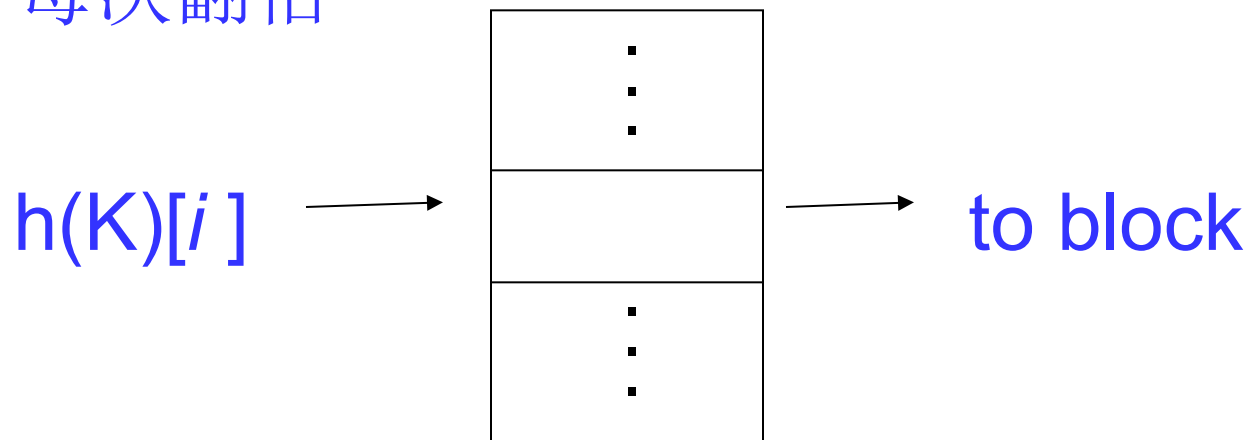
(2) 线性散列表 (每次**B**加1)

可扩展散列: 三个要点

(a) 哈希函数产生的 b 位输出中的只用前 i 个位

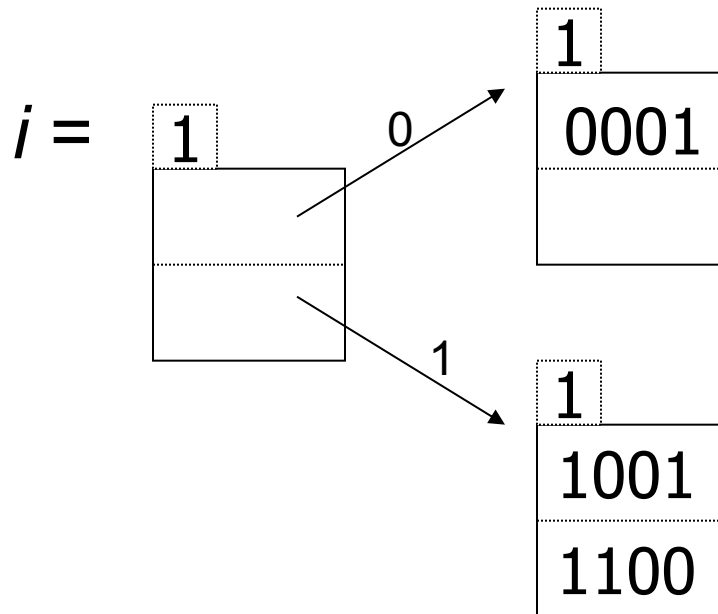


(b) 增加一个指向块的指针数组（桶数组），
每次翻倍



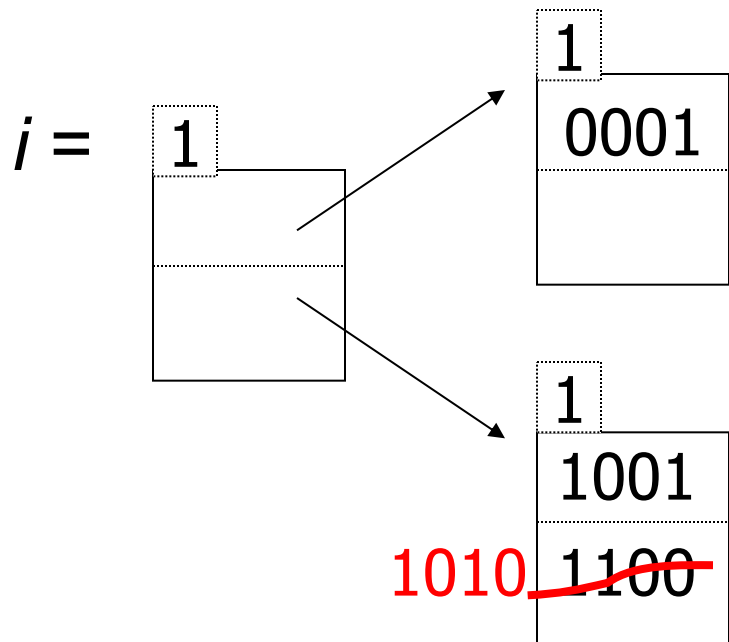
(c)若干个桶可以共享一个块

举例: $h(k)$ 是4位; 2个键/桶

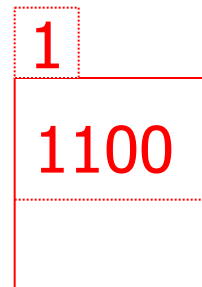


插入 1010

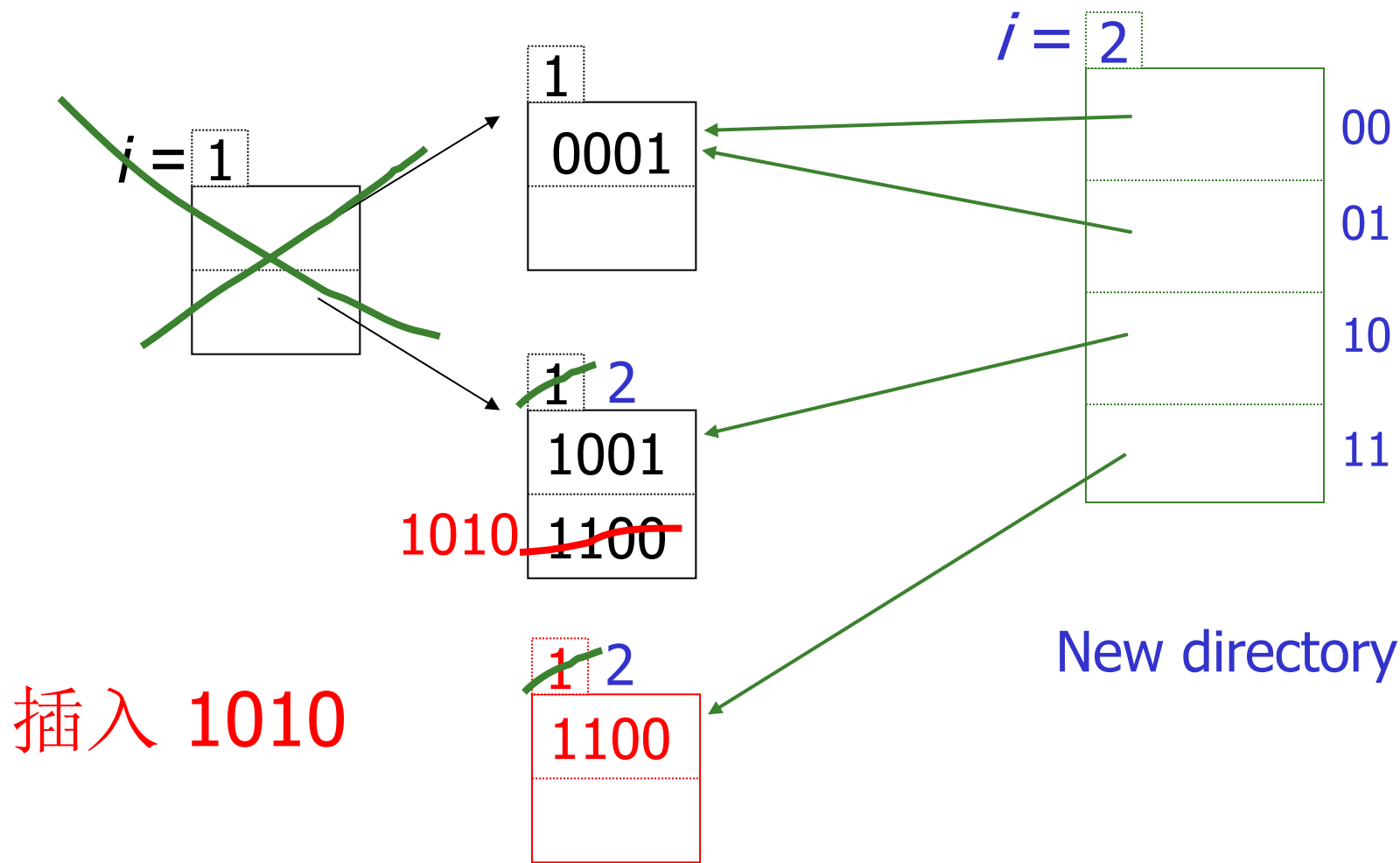
举例: $h(k)$ 是4 个位; 2 键/桶



插入 1010



举例: $h(k)$ 是4 个位; 2 键/桶



可扩展散列表——插入

(1) 为了插入键值为 K 的记录，计算 $h(K)$ ，取出前 i 位，并找到桶数组中序号为前 i 位对应的项

(注意： i 作为桶数组数据结构的一部分被保存)。

(2) 根据桶数组中该项对应的指针找到存储块 B 。

如果存储块 B 有存放记录的空间，直接插入；

如果空间不够，视每个存储块保存 j 值的不同 (j 是存储块保存的数据结构)，采取不同策略

散列表——可扩展散列表

$j < i$ 的情况，桶数组大小没有任何变化

- (1) 将块 **B** 分成两个存储块
- (2) 根据记录散列值的 $(j+1)$ 位，将 **B** 中的记录分配到这两个块中，该位为 0 的放在 **B** 中，该位是 1 的放在新块中
- (3) 将 $(j+1)$ 存入这两个存储块的数据结构中，以表明确定该存储块成员资格的二进制位数
- (4) 调整桶数组的指针，使原来指向 **B** 块的指针指向新的 **B** 块或新加块。

散列表——可扩展散列表

$j=i$ 的情况，桶数组大小要变化

(1) 将 i 加1，桶数组长度翻了一倍，桶数组中现在有 2^{i+1} 项

(2) 假定 ω 是以前确定桶序号的前 i 位二进制序列，那么新的两个桶序号应是 $\omega 0$ 和 $\omega 1$ ，它们的桶指针都指向原来的同一个存储块，即共享相同的存储块。

(3) 此时， $j < i$ ，分裂存储块B

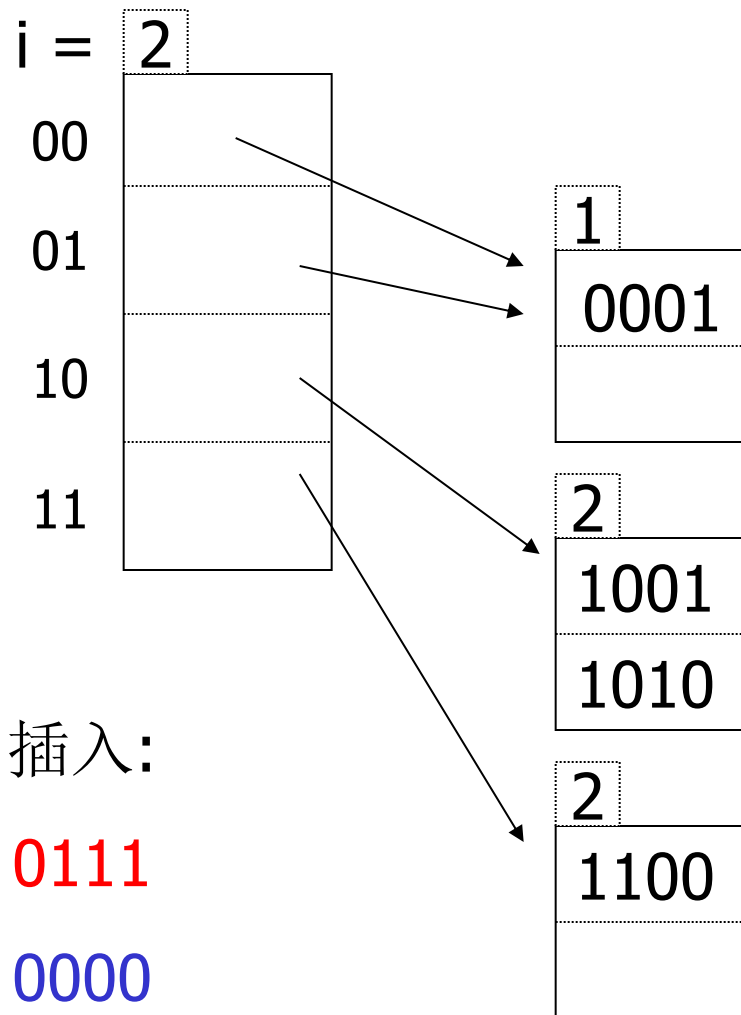
散列表——可扩展散列表

例子4.31，是可扩展散列的基本结构

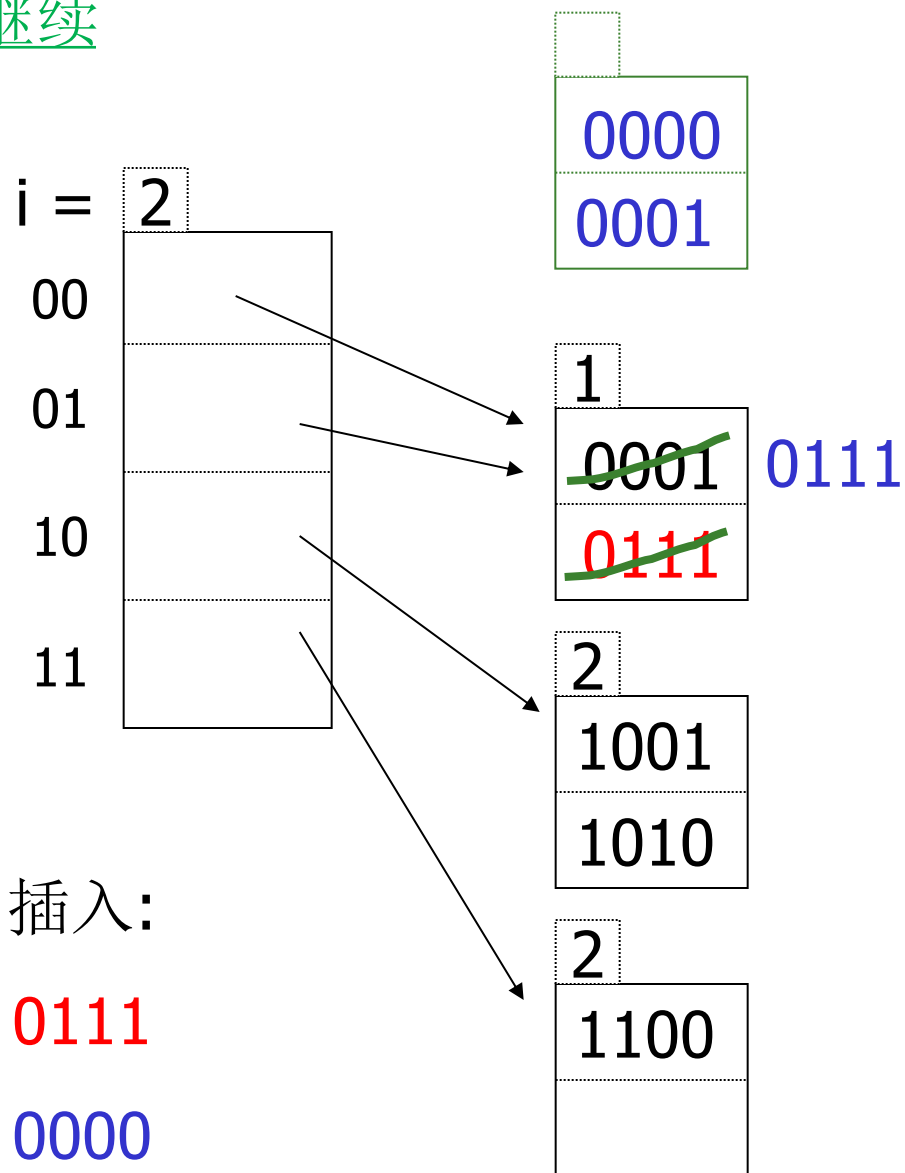
例子4.32，

依次插入1010, 0000, 0111, 1000，分析变化

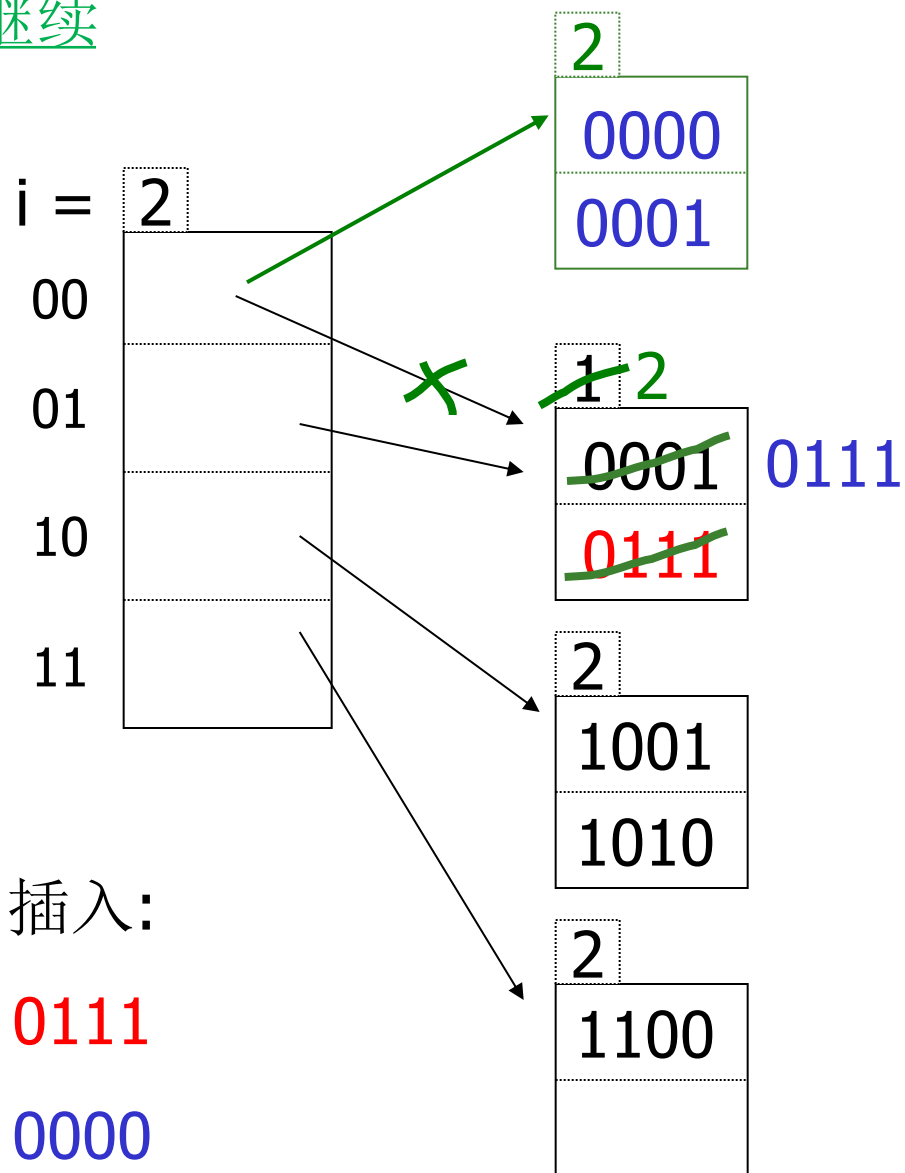
例子继续



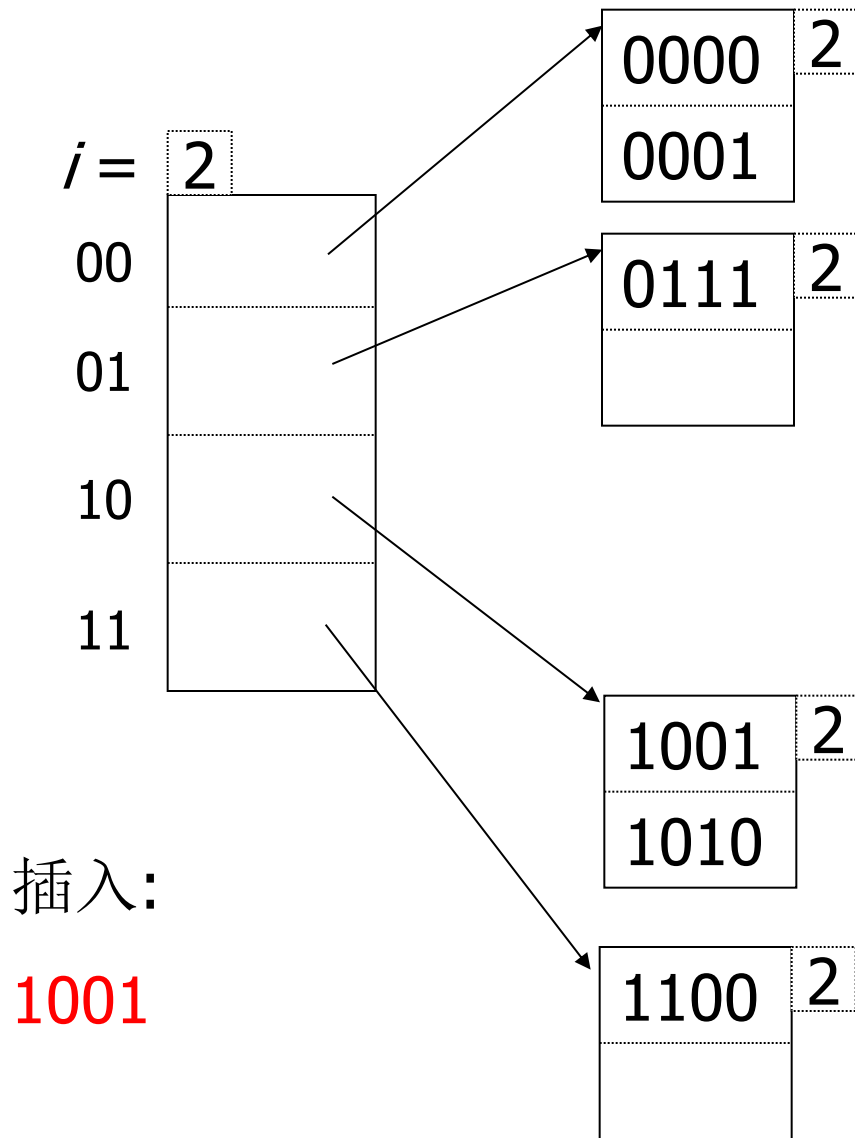
例子继续



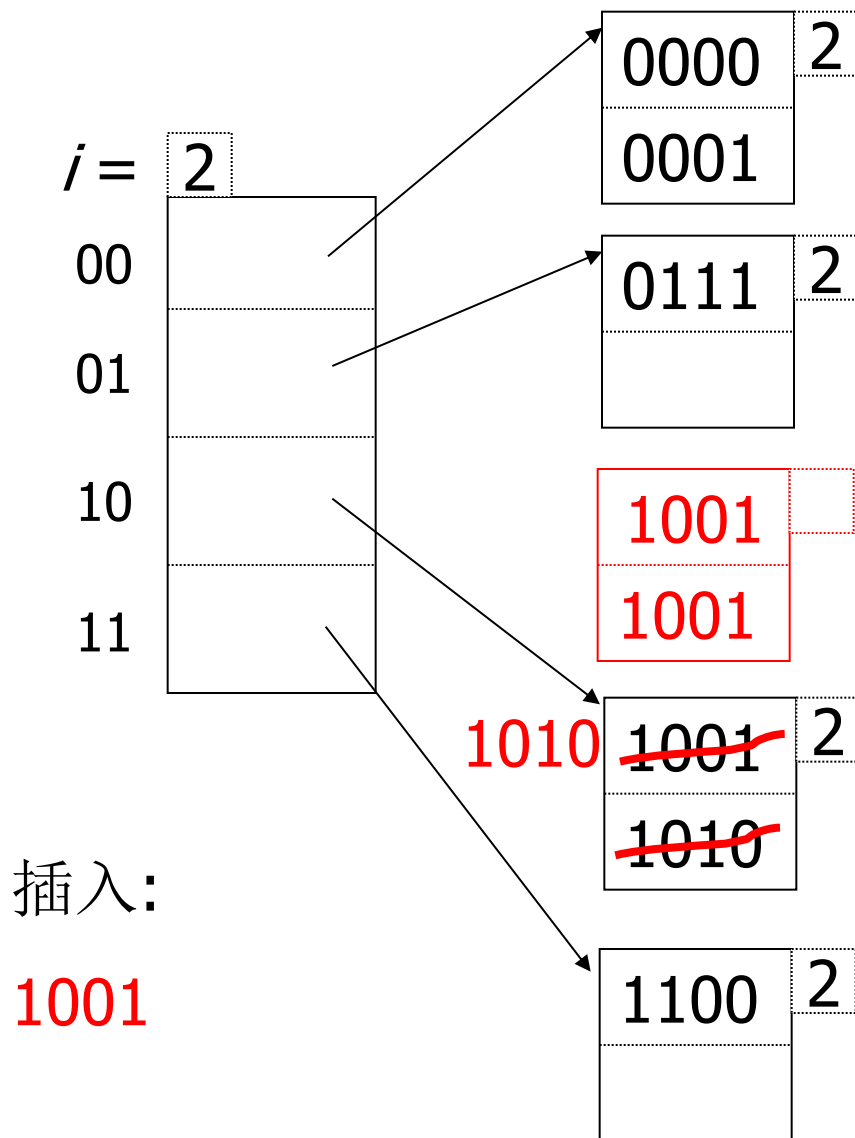
例子继续



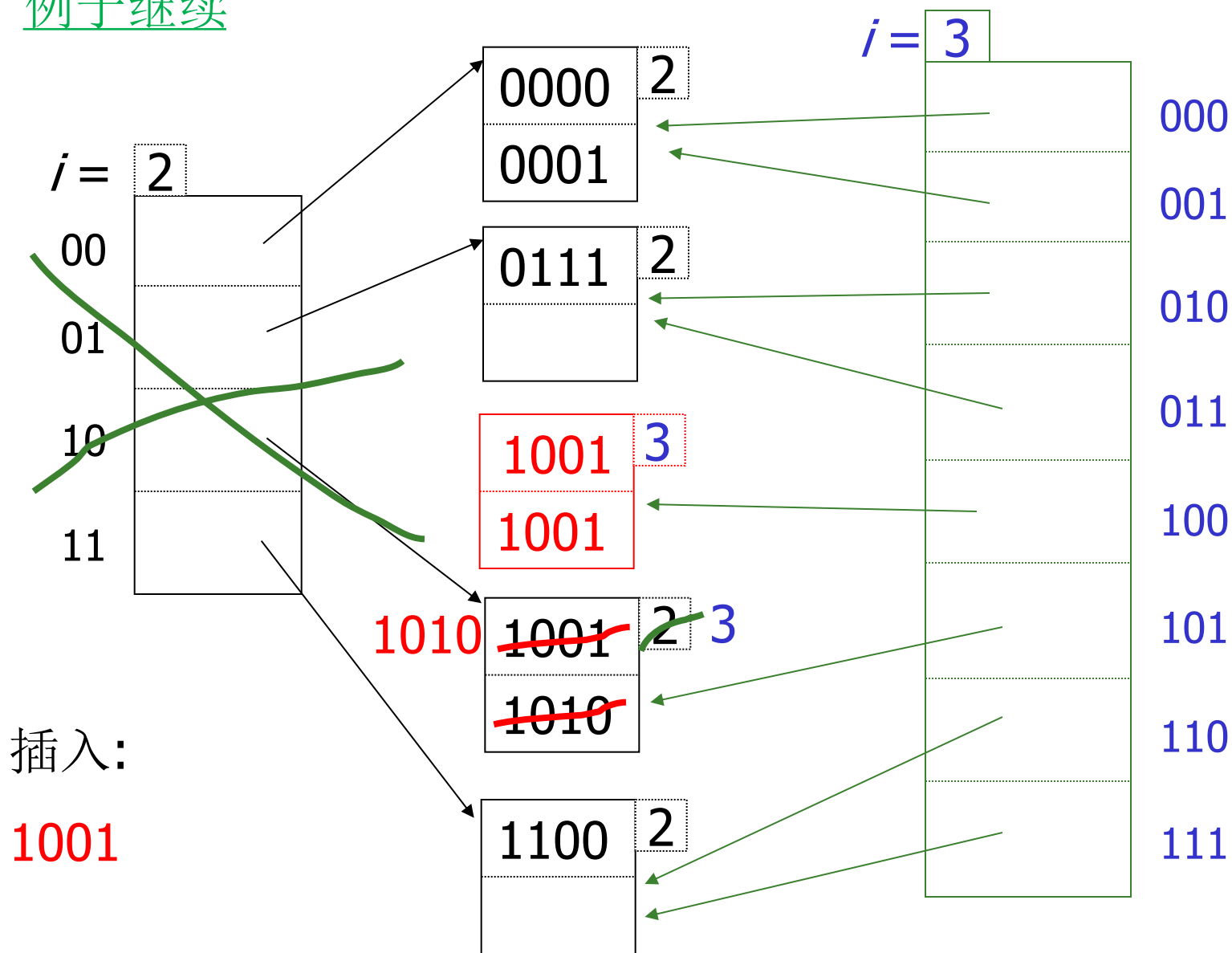
例子继续



例子继续



例子继续



散列表——可扩展散列表

优点：

（1）如果桶数组小到可以存放到内存，可以在一个存储块中找到键值为K的记录，效率高

缺点：

（1）桶数组翻倍时，可能无法全部放入主存，会使I/O大幅增加

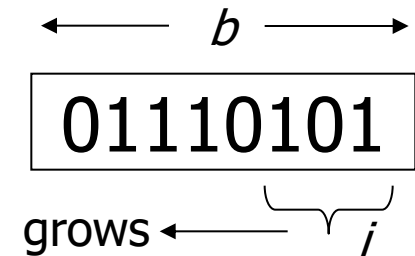
（2）某些块的分裂比逻辑上提前许多，从而大幅度浪费空间

线性散列表

■ 另一种动态散列（哈希）技术

三个点子：

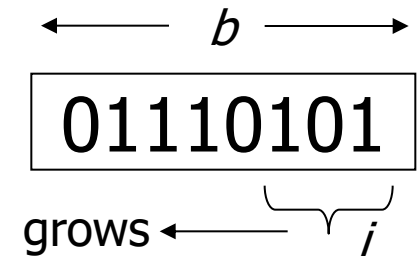
(a) Use i low order bits of hash



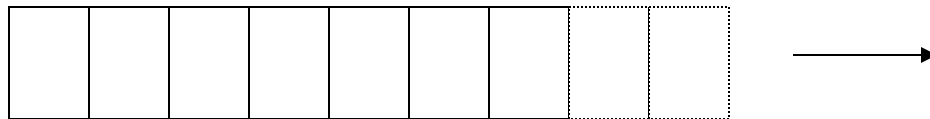
线性散列

三个点子:

(a) Use i low order bits of hash



(b) File grows linearly (每次只增加一个桶)



(c) 每个桶可以溢出 (对比可扩展散列)

线性散列的三个参数

i: 当前使用的散列函数值的位数

n: 当前的桶数

r: 散列表中记录的总数

线性散列的三个参数

n的选择:

使所有存储块的记录总数和存储块数量是一个固定的比值，比如1.7。换句话说，平均每个块存储的记录数是一个固定值。

i的确定:

桶序号的二进制的位数 $i = \lceil \log_2 n \rceil$, 这些位总是从散列函数值（二进制）的最右端开始取。

线性散列的基本插入操作

假定散列函数值的最右侧 i 位用来给桶数组编号，一个键值为 K 的记录要插入到编号是 a_1a_2, \dots, a_i 的桶中，即 a_1a_2, \dots, a_i 是 $h(K)$ 的后 i 位。

把 a_1a_2, \dots, a_i 当作二进制整数，值为 m 。如果 $m < n$ ，则插入到编号为 m 的桶中；否则插入到 $m - 2^{i-1}$ 的桶中（即把 a_1 改成0）。

例4.33, pp.130

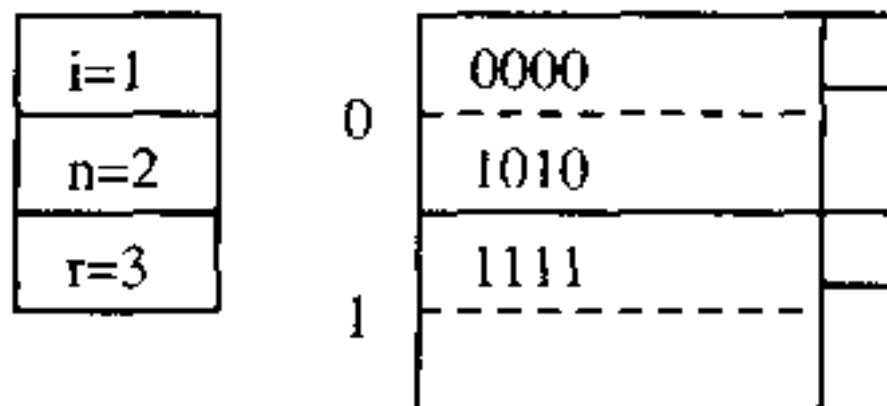


图4-36 线性散列表

例4.33: $b=4$ bits, $n=2$, $i=1$,
2 keys/bucket, $r/n \leq 1.7$

0000	1111
1010	

0 1

← Future growth
buckets

例4.33: $b=4$ bits, $n=2$, $i=1$,
2 keys/bucket, $r/n \leq 1.7$

• insert 0101

0000
1010

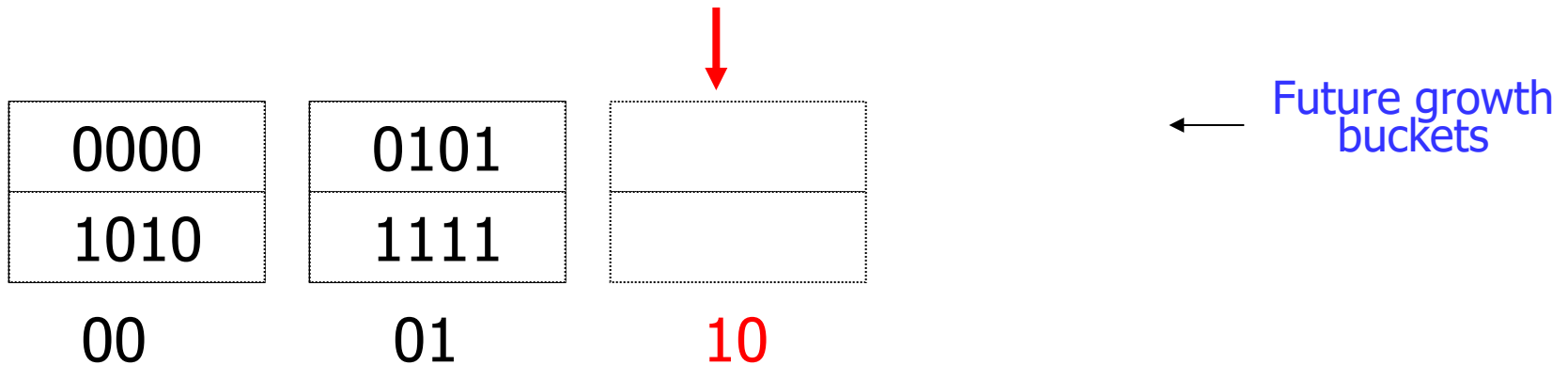
0

1111
0101

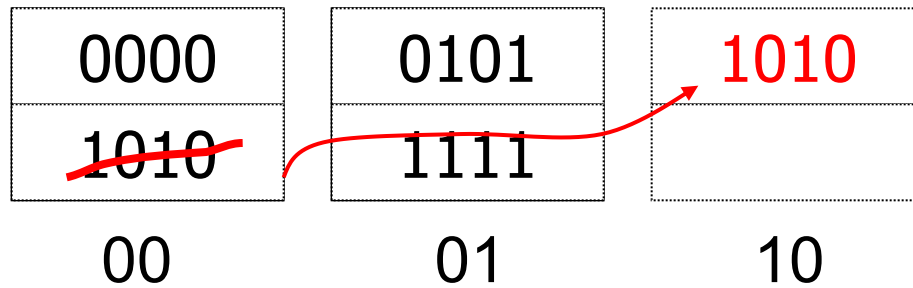
1

← Future growth
buckets

- 注意：
- 由于 $4/2=2$,大于1.7
 - 将n提高到3, 因为 $\lceil \log_2 3 \rceil = 2$,因此 $i=2$,
 - 将桶号0, 1变成00, 01, 同时增加一个新桶, 对应的二进制桶号是10

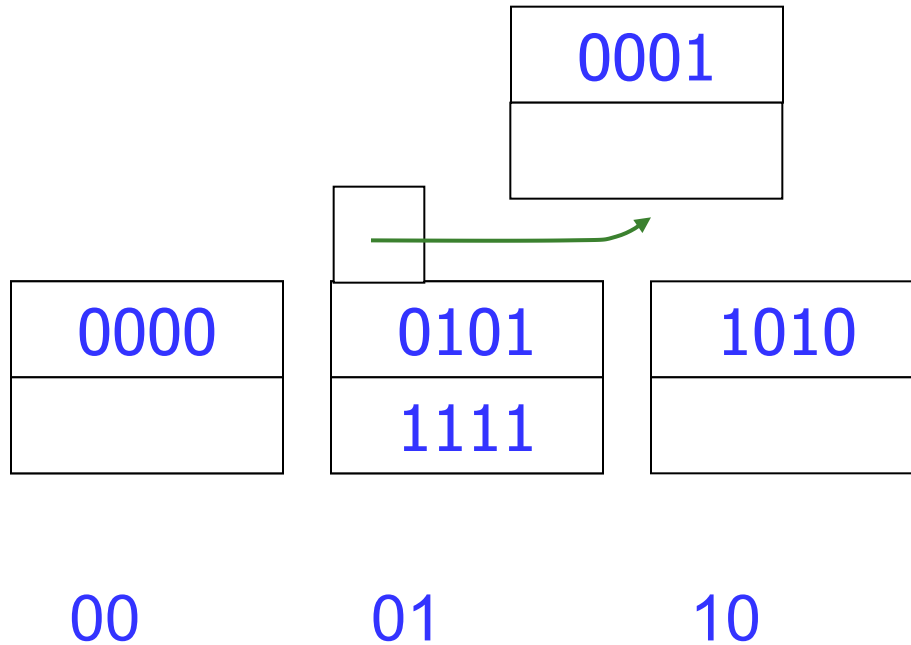


例4.33: $b=4$ bits, $n=3$, $i=2$,
2 keys/bucket, $r/n \leq 1.7$



← Future growth
buckets

例4.34: insert 0001,溢出块?



1. 插入后, 计算 $r/n=5/3=1.677$
2. 不需要加桶, 因为比值小于1.7!

总结：线性散列插入的桶分裂过程

- 1、每次插入后，计算 r/n 的比值，若超过设定的阈值（比如1.7），则增加一个新的桶到散列表中。
- 2、当 n 超过 2^i 时， i 递增1。
- 3、如果新加入的桶号的二进制表示是 $1a_2, \dots, a_n$ ，就分裂原来二进制桶号是 $0a_2, \dots, a_n$ 的记录到两个桶中。

散列表——线性散列表

桶增长较为缓慢，其基本思想如下

（1）桶数 n 的选择：使存储块的平均记录数和存储块所能容纳的记录总数成固定比例（ $r \leq 1.7n$ ）。

（2）存储块不总是可以分裂（比值没有超过1.7），但可以有溢出块。

散列表——线性散列表

桶增长较为缓慢，其基本思想如下

(3) 给定一条记录，首先根据散列函数获得散列值。

(4) 假定散列函数值的 i 位作为桶序号，不妨记为 a_1a_2, \dots, a_i ，即是 $h(K)$ 的右 i 位，将 a_1a_2, \dots, a_i 当作一个整数 m ：如果 $m < n$ ，则把记录放入 m 桶，如果 $n \leq m < 2^n$ ，此时 m 桶不存在，将桶序号的第一位变为0，放入对应的桶中。

例4.34: 继续

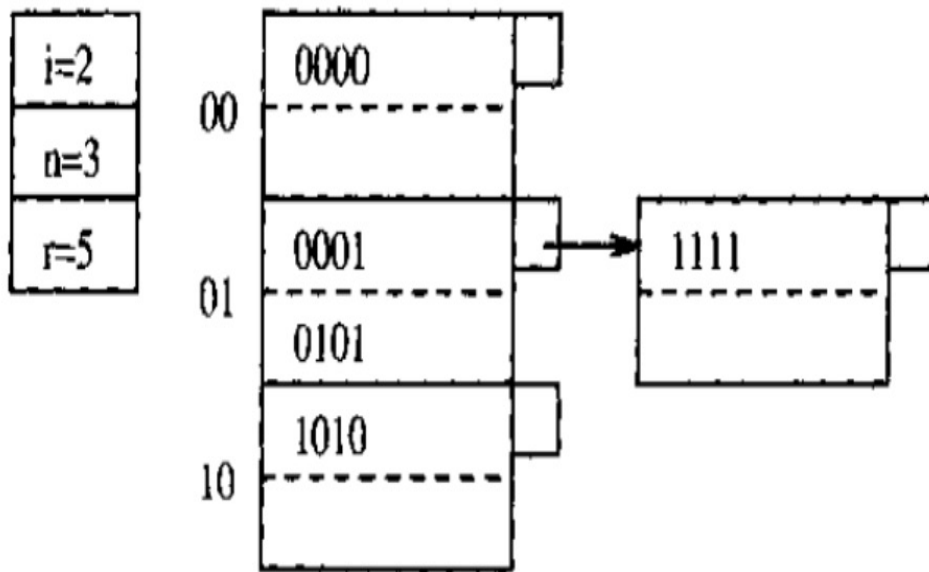


图4-38 必要时使用溢出块

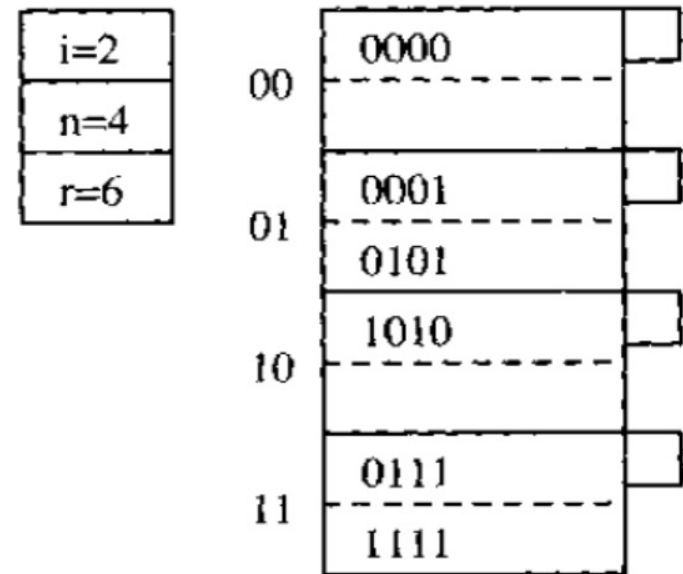


图4-39 增加第四个桶

插入0111?

(1) 放入01桶

(2) 比值是2，要增加一个新的桶11，然后分裂01桶的数据到两个桶

例4.35:查询

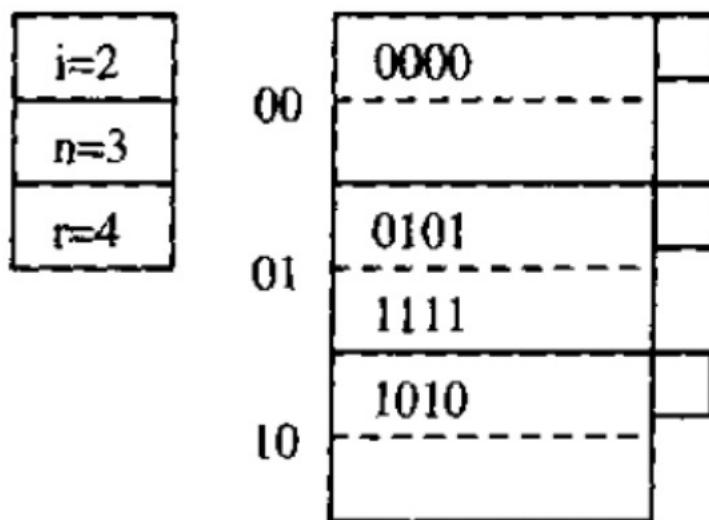


图4-37 增加第三个桶

查找1010, 1011?

- (1) 第一个在10桶中找到
- (2) 第二个先到11桶, 没有对应的桶, 改左侧最高位是0, 即01桶中查找, 发现没有, 结束。

课外阅读（图书馆ACM）

- （1） Operating System Support for Database Management
， Michael Stonebraker
- （2） Generalized Search Trees for Database Systems,
Joseph M. Hellerstein , Jeffrey F. Naughton , Avi Pfeffer

作业

`index_db.py`（增加一种索引，比如单级索引、多级索引、**B**树索引或散列表）

（1）文件结构

（2）构造函数

（3）`insert_index_entry`（）函数