Chap 8. Hashing (2)

Contents

8.3 Dynamic Hashing

8.3.1 Motivation for Dynamic Hashing

- It is necessary to increase *the size of a hash table* whenever its *loading density* exceeds a pre-specified threshold
  - Ex) b buckets with divisor D = b → $2b+1$ buckets with divisor $2b+1$

- But, rebuilding the hash table takes time

➡ Reduce the rebuild time using ***dynamic hashing***

- Two forms of dynamic hashing
  - *Using directory*
  - *Directoryless*

- For both forms,
  - h maps keys into non-negative integers
  - The range of h is sufficiently large
  - ***h*(*k*, *p*)** : the integer formed by the p LSBs of $h(k)$

- Example

k	$h(k)$
A0	100 000
A1	100 001
B0	101 000
B1	101 001
C1	110 001
C2	110 010
C3	110 011
C5	110 101

- keys composed of two characters
- Hash function $h(k)$
  - transforms keys into 6-bit nonnegative integers
    - letters A~C → 100, 101, 110
    - digits 0~7 → 000, 001, … ,111
  - h(A0,1) = 0
    h(A1,3) =1
    h(B1,4) = 1001 = 9
    h(C1,6) = 110001 = 49

8.3.2 Dynamic Hashing Using Directories

- **Use a directory *d* of pointers to buckets**

- Directory depth
  - the number of bits of $h(k)$ used to index the directory
  - $h(k, t)$: t is a directory depth

- The size of directory depends on directory depth
  - $h(k,2)$ → directory size = $2^2 = 4$
  - $h(k,5)$ → directory size = $2^5 = 32$

- The number of buckets is at most equal to the directory size.

k	$h(k)$
A0	100 000
A1	100 001
B0	101 000
B1	101 001
C1	110 001
C2	110 010
C3	110 011
C5	110 101

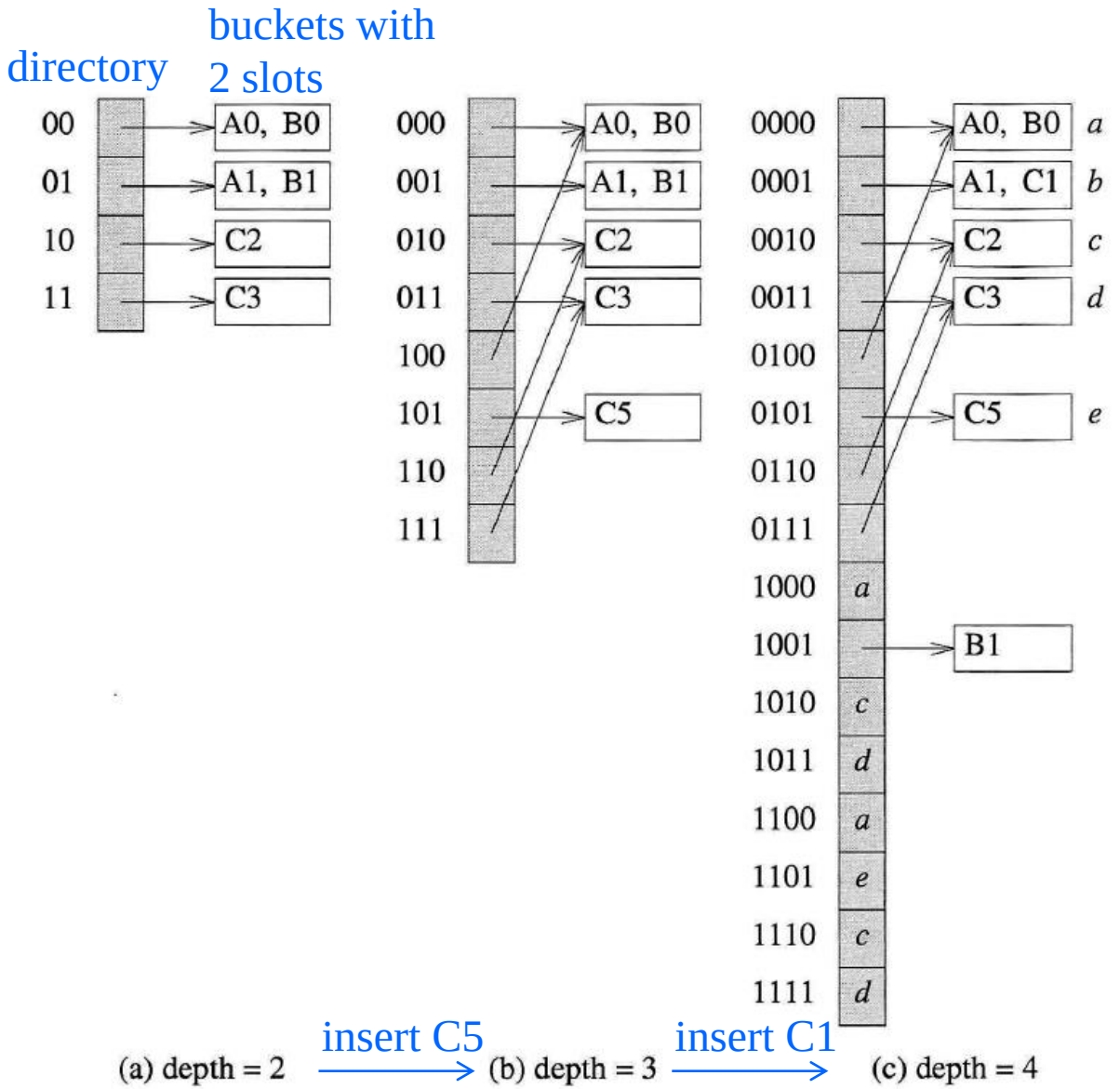
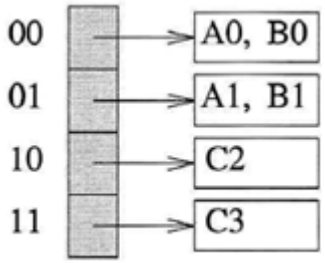


Figure 8.8: Dynamic hash tables with directories

< Case 1 >

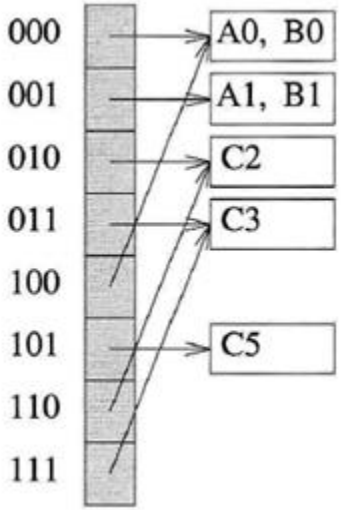
k	$h(k)$
A0	100 000
A1	100 001
B0	101 000
B1	101 001
C1	110 001
C2	110 010
C3	110 011
C5	110 101

(a) depth =2



insert C5

(b)



h(C5, 2) = 01

bucket overflow

the least u that $h(k, u)$ is not the same for all keys in the overflowed bucket ? ***u = 3***

In the case least u is ***greater than*** the directory depth

① increase directory depth to $u = 3$

② ***double*** the directory size

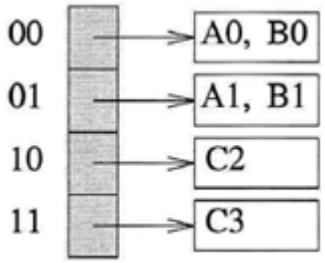
③ the pointers in the original directory are duplicated
$d[i+4]=d[i]$, $0 \le i < 4$

④ split the overflowed bucket using $h(k, u)$
001: A1, B1, 101:C5

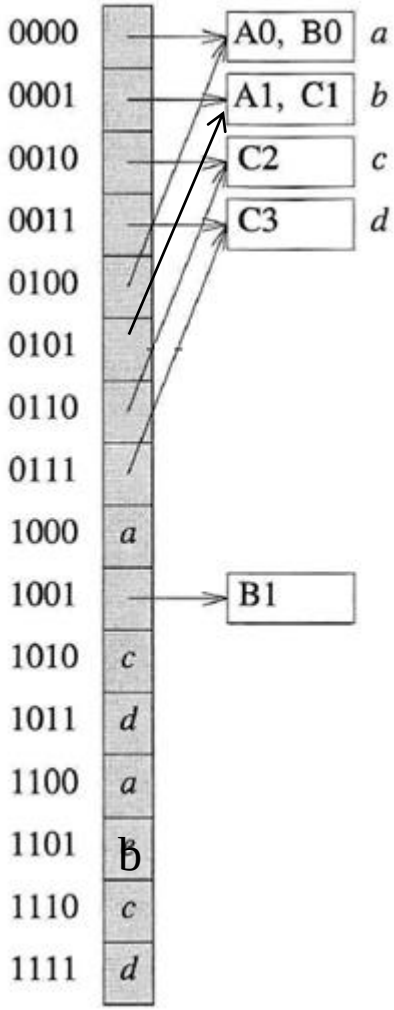
< Case 2 >

(a) depth =2

k	$h(k)$
A0	100 000
A1	100 001
B0	101 000
B1	101 001
C1	110 001
C2	110 010
C3	110 011
C5	110 101



insert C1 →



h(C1,2) = 01

bucket overflow

the least u that $h(k,u)$ is not the same for all keys in the overflowed bucket ? ***u = 4***

In the case least u is ***greater than*** the directory depth

① increase directory depth to u =4

② ***quadruple*** the directory size

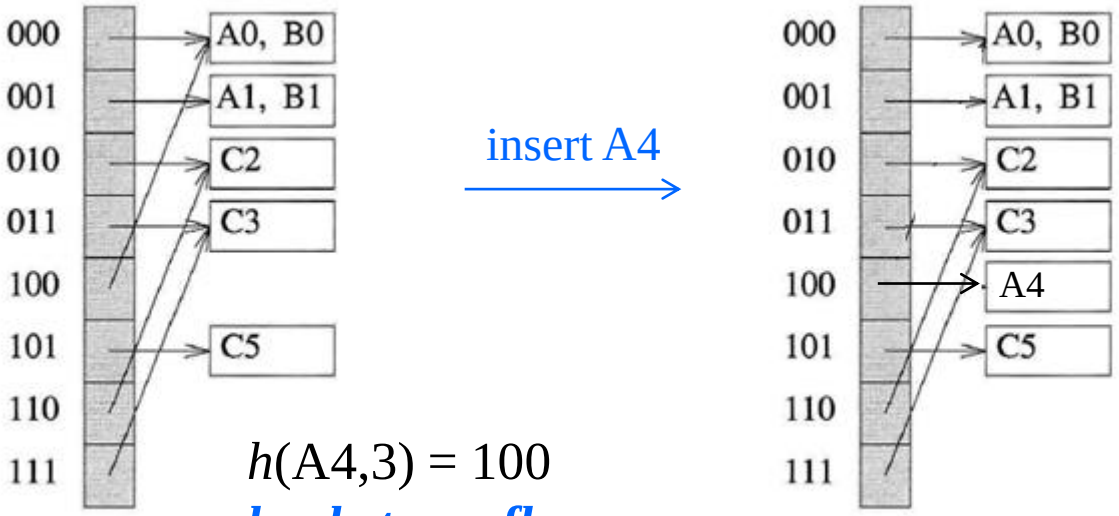
③ the pointers in the original directory are replicated 3 times,

④ split the overflowed bucket using $h(k, u)$
0001: A1, C1, 1001:B1

< Case 3 >

(b) depth =3

k	$h(k)$
A0	100 000
A1	100 001
B0	101 000
B1	101 001
C1	110 001
C2	110 010
C3	110 011
C5	110 101



h(A4,3) = 100

bucket overflow

the least u that $h(k,u)$ is not the same for all keys in the overflowed bucket ? ***u = 3***

In the case least u is ***less than* or *equal*** to the directory depth

① The size of directory is not changed

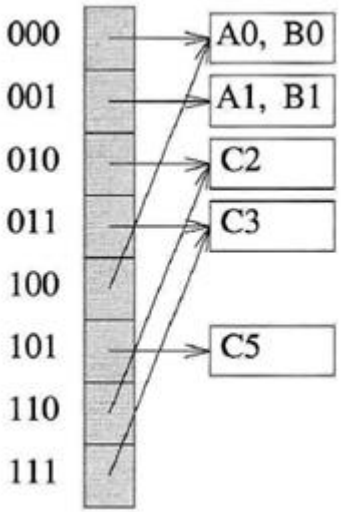
② split the overflowed bucket using $h(k, u)$
000: A0, B0, 100:A4

③ update d[100] to point to the new bucket

< Case 4 >

k	$h(k)$
A0	100 000
A1	100 001
B0	101 000
B1	101 001
C1	110 001
C2	110 010
C3	110 011
C5	110 101

(b) depth =3



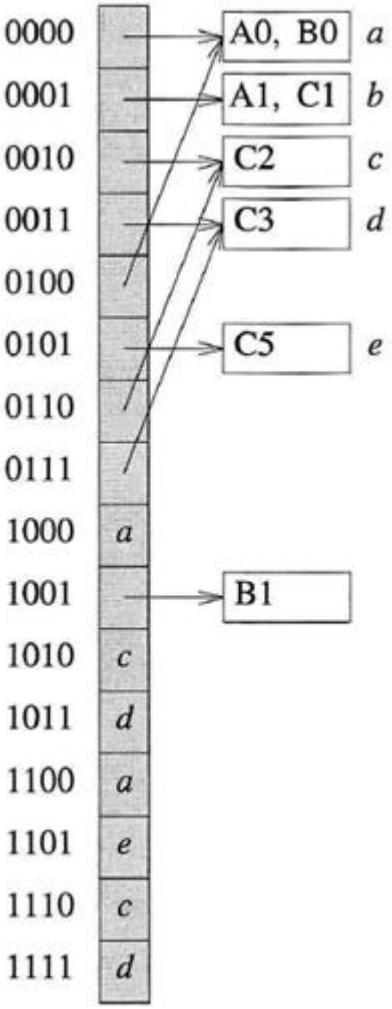
h(C1,3) = 001

bucket overflow

…

insert C1

(c)



8.3.3 Directoryless Dynamic Hashing

- Known as *linear dynamic hashing*

- An array, *ht*, of buckets is used, instead of using directory, *d*, of bucket pointers.

- Assumption
  - this array is as large as possible and so there is no possibility of increasing its size dynamically.

- Two variables q and r, $0 \leq q < 2^r$
  - r is the number of bits of $h(k)$ used to index into the hash table
  - q is the bucket that will split next

- At any time, *only buckets 0 through 2^r+q-1 are active*

k	$h(k)$
A0	100 000
A1	100 001
B0	101 000
B1	101 001
C1	110 001
C2	110 010
C3	110 011
C5	110 101

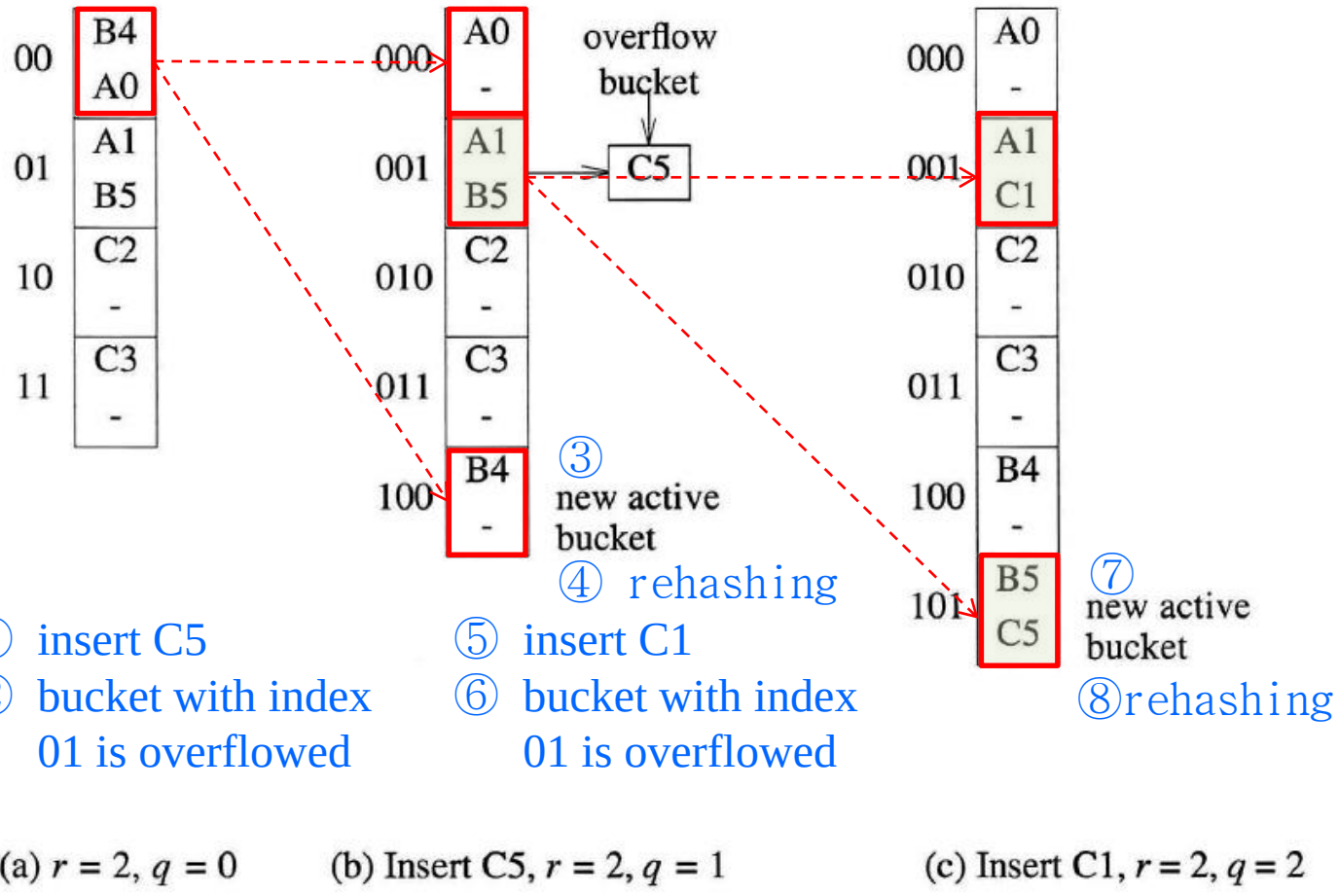
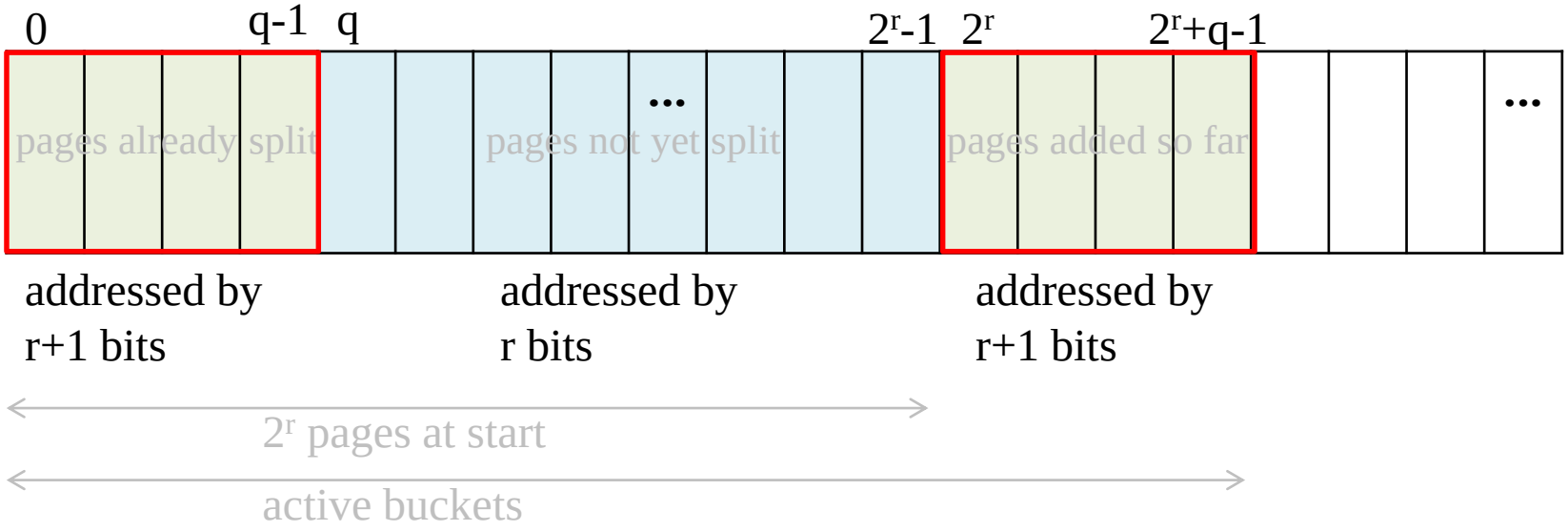


Figure 8.9: Inserting into a directoryless dynamic hash table



In case q becomes 2^r, we increment r by 1 and reset q to 0 .

```
if (h(k,r) < q) search the chain that begins at bucket h(k,r+1);
else search the chain that begins at bucket h(k,r);
```

Program 8.5: Searching a directoryless hash table