

# Hashing – 보충자료

Data Structures

# Motivation for Dynamic hashing

---

- 적재율(loading factor)이 임계값을 넘을 때, 현재 크기의 대략 두 배 크기의 테이블로 재해싱을 해야 한다
- 예를 들어,
  - $b$ 로 나누는 제산(*division*) 해시 함수  $\rightarrow b$ 개의 버킷
  - 삽입으로 인해 적재율이 임계값을 넘게 되면 해시 테이블의 크기를  $2b+1$ 로 증가시킨다
  - 이와 동시에 나누는 수도  $2b+1$ 로 변경해야 한다
  - 각 엔트리에 해당하는 홈버킷들이 바뀌었을 수 있으므로 새로 만든 테이블에 다시 삽입하여 재조정하여야 한다
- 동적해싱(dynamic hashing)은 재조정을 한 번 할 때마다 오직 하나의 버킷 안에 있는 엔트리들에 대해서만 홈버킷을 변경하게 하여 재조정 시간을 줄이는 방법이다

# Two forms of dynamic hashing

- 디렉토리(directory)를 사용하는 동적해싱
- 디렉토리가 없는 동적해싱
- $h$  : 키를 음이 아닌 정수로 사상시키는 해시 함수
- $h(k,p)$  :  $h(k)$ 의  $p$ 개의 최하위비트(least significant bit)에 의해 표현되는 정수

$k$	$h(k)$
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 8.7: an example of hash function

# Dynamic hashing using directories

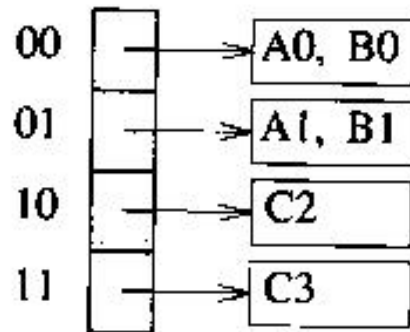
---

- 버킷들에 대한 포인터를 저장하고 있는 디렉토리  $d$ 를 이용
- 디렉토리 깊이  $t$  : 디렉토리를 인덱싱할 때 사용되는  $h(k)$ 의 비트 수
- $h(k,2)$ 를 사용하여 인덱싱하면 디렉토리 크기는  $2^2 = 4$ 가 된다

# Dynamic hashing using directories

---

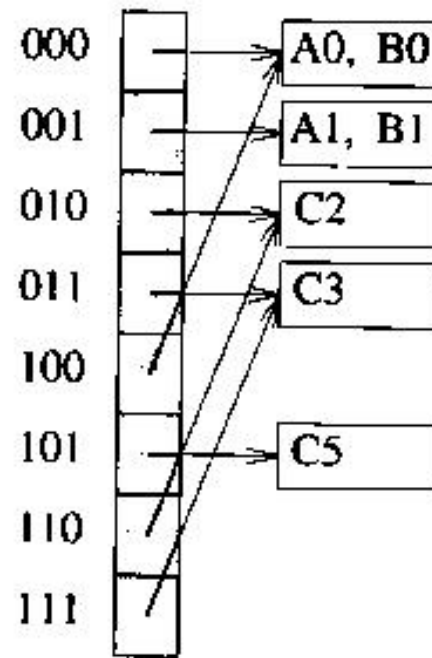
- 키  $k$ 를 탐색하려면,  $d[h(k,t)]$ 가 가리키는 버킷을 탐색한다
- 삽입에 의해 버킷 오버플로우가 발생할 때
  - 오버플로우 된 버킷의 모든 키에 대해  $h(k,u)$ 가 같지 않게 하는 최하위 비트  $u$ 를 결정한다.
  - $u$ 가 디렉토리 깊이  $t$ 보다 크면 디렉토리 깊이를 증가시킨다 -> 디렉토리 크기 증가
  - 새로 추가된 부분에 원래 디렉토리에 있던 포인터들을 복사한다
  - 오버플로우 된 버킷을 분할한다.
- 테이블에 있는 모든 엔트리들을 재해싱하는 것이 아니라 오버플로우 된 버킷에 있는 엔트리들만 재해싱!!



•한 버킷에 두 개씩 저장  
한다고 가정할 때

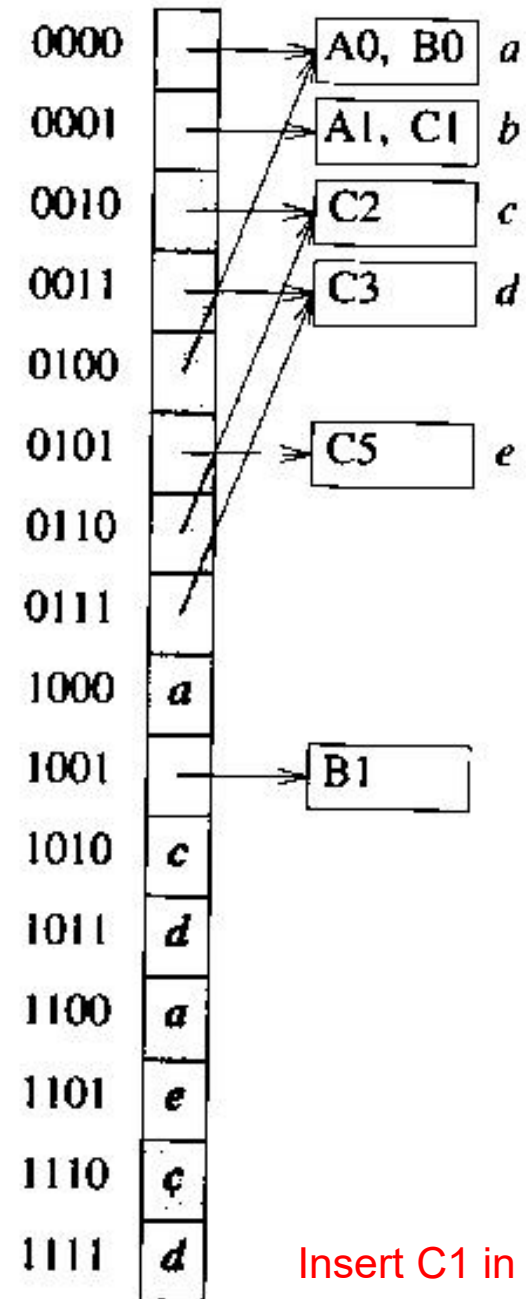
$k$	$h(k)$
A0	100 000
A1	100 001
B0	101 000
B1	101 001
C1	110 001
C2	110 010
C3	110 011
C5	110 101

(a) depth = 2



Insert C5 in (a)

(b) depth = 3



Insert C1 in (b)

(c) depth = 4

# Directoryless dynamic hashing

---

- $ht$ : 버킷의 배열  
배열  $ht$ 는 가능한 한 매우 크다고 가정한다
- 활성화된 버킷에 대한 정보를 위한 두 변수  $q$  와  $r$  ( $0 \leq q < 2^r$ )
  - $r$ : 해시 테이블을 인덱스하기 위해 사용되는  $h(k)$ 의 비트 수
  - $q$ : 다음에 분할 할 버킷
- 언제든지 0 부터  $2^r + q - 1$  까지의 버킷만 활성화
- 오버플로우는 버킷  $2^r + q$ 을 활성화시켜 해결한다 체인  $q$ 에 있던 엔트리들을 버킷  $2^r + q$  을 사용하여 재조정하고,  $q$ 를 1만큼 증가시킨다
- $q$ 가  $2^r$ 이 되면  $r$ 을 1만큼 증가시키고  $q$ 를 0으로 재설정

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

B4 101 100

B5 101 101

00	B4
	A0
01	A1
	B5
10	C2
	-
11	C3
	-

000	A0	
	-	
001	A1	
	B5	→ overflow bucket
010	C2	
	-	
011	C3	
	-	
100	B4	
	-	new active bucket

000	A0	
	-	
001	A1	
	C1	
010	C2	
	-	
011	C3	
	-	
100	B4	
	-	
101	B5	
	C5	new active bucket

a)  $r = 2, q = 0$

(b) Insert C5,  $r = 2, q = 1$

(c) Insert C1,  $r = 2, q = 2$



# search

---

---

```
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