해싱

15 주차-강의

남춘성

SUNG KYUN KWAN UNIVERSITY

해싱(Hashing)

SUNG KYUN KWAN UNIVERSITY

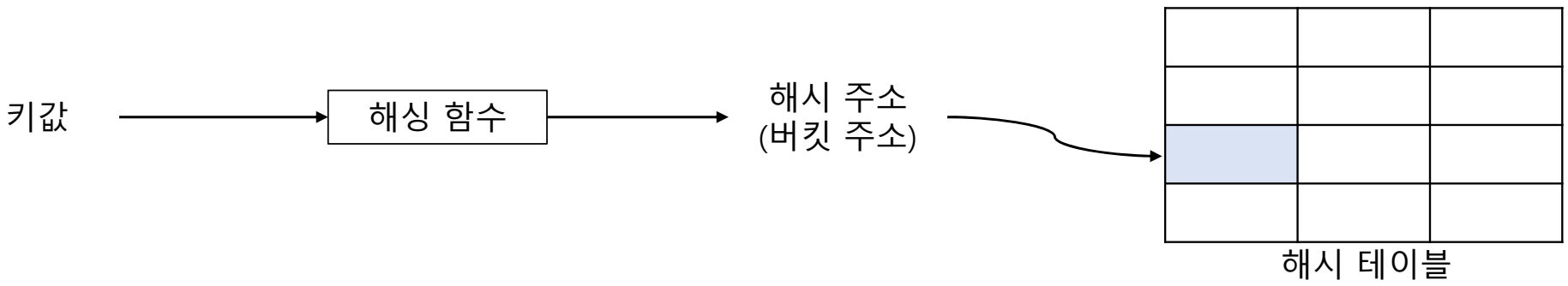
- 산술적인 연산을 이용하여 키가 있는 위치를 계산하여 바로 찾아가는 계산 검색 방식

- 검색 방법
  - 키 값에 대해서 해싱 함수를 계산하여 주소 생성
  - 생성 주소에 해당하는 해시 테이블로 이동
    - 해당 주소에 찾는 항목이 있으면 검색 성공, 없으면 검색 실패

- 해싱 함수(hashing function)
  - 키 값을 원소의 위치로 변환하는 함수

- 해시 테이블(hash table)
  - 해싱 함수에 의해서 계산된 주소의 위치에 항목을 저장한 표

해싱 검색 수행방법



해싱 용어 저리

- 충돌 (collision)
  - 서로 다른 키 값에 대해서 해싱 함수에 의해 주어진 버킷 주소가 같은 경우
  - 충돌이 발생한 경우에 비어있는 슬롯에 동거자 관계로 키 값 저장

- 동거자 (synonym)
  - 서로 다른 키 값을 가지지만 해싱 함수에 의해서 같은 버킷에 저장된 키 값들

- 오버플로우 (overflow)
  - 버킷에 비어있는 슬롯이 없는 포화 버킷 상태에서 충돌이 발생하여 해당 버킷에 키 값을 저장할 수 없는 상태

키 및 적재 밀도

SUNG KYUN KWAN UNIVERSITY

- **키 값 밀도**
  - 사용 가능한 전체 키 값들 중에서 현재 해시 테이블에 저장되어서 실제 사용되고 있는 키 값의 개수 정도

$$\text{키값 밀도} = \frac{\text{실제 사용 중인 키값의 개수}}{\text{사용 가능한 키값의 개수}}$$

- **적재 밀도**
  - 해시 테이블에 저장 가능한 키 값의 개수 중에서 현재 해시 테이블에 저장되어서 실제 사용되고 있는 키 값의 개수 정도

$$\text{적재 밀도} = \frac{\text{실제 사용 중인 키값의 개수}}{\text{해시 테이블에 저장 가능한 전체 키값의 개수}}$$

$$= \frac{\text{실제 사용 중인 키값의 개수}}{\text{버킷 개수} \times \text{슬롯 개수}}$$

해싱함수 조건

SUNG KYUN KWAN UNIVERSITY

- 해싱 함수는 계산이 쉬워야 한다.
  - 비교 검색 방법을 사용하여 키 값의 비교연산을 수행하는 시간보다 해싱 함수를 사용하여 계산하는 시간이 빨라야 해싱 검색을 사용하는 의미가 된다.

- 해싱 함수는 충돌이 적어야 한다.
  - 충돌이 많이 발생한다는 것은 같은 버킷을 할당 받는 키 값이 많다는 것이므로 비어있는 버킷이 많은데도 어떤 버킷은 오버플로우가 발생할 수 있는 상태가 되므로 좋은 해싱 함수가 될 수 없다.

- 해시 테이블에 고르게 분포할 수 있도록 주소를 만들어야 한다.

해싱 함수 종류 - I

- 중간 제곱 함수
  - 키 값을 제곱한 결과 값에서 중간에 있는 적당한 비트를 주소로 사용하는 방법
  - 제곱한 값의 중간 비트들은 대개 키의 모든 값과 관련이 있기 때문에 서로 다른 키 값은 서로 다른 중간 제곱 함수 값을 갖게 된다.
  - 예) 키 값 00110101 10100111에 대한 해시 주소 구하기

```
                    00110101 10100111
X                   00110101 10100111
-------------------------------------
    0000101100111110100100101111 0001
```

해싱 함수 종류 - II

SUNG KYUN KWAN UNIVERSITY

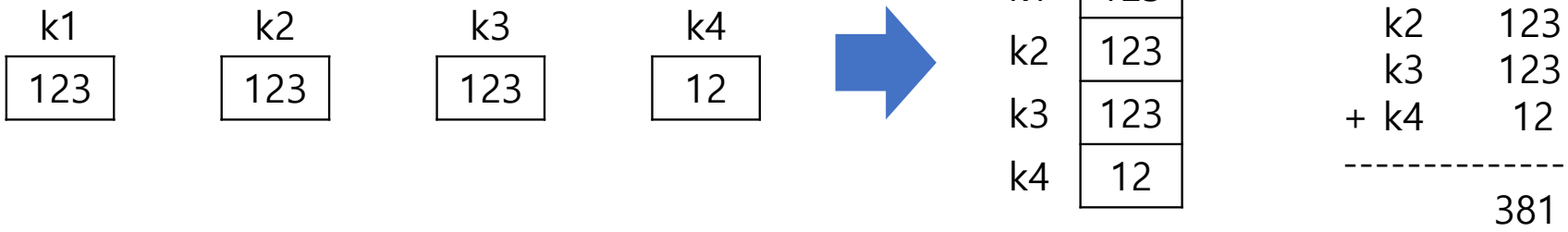
- **제산 함수**
  - 함수는 나머지 연산자 mod(C에서의 %연산자)를 사용하는 방법
  - 키 값 k를 해시 테이블의 크기 M으로 나눈 나머지를 해시 주소로 사용
  - 제산함수 : $h(k) = k \bmod M$
  - M으로 나눈 나머지 값은 0~(M-1)이 되므로 해시 테이블의 인덱스로 사용
  - 해시 주소는 충돌이 발생하지 않고 고르게 분포하도록 생성되어야 하므로 키 값을 나누는 해시 테이블의 크기 M은 적당한 크기의 소수(prime number) 사용

- **승산 함수**
  - 곱하기 연산을 사용하는 방법
  - 키 값 k와 정해진 실수 α를 곱한 결과에서 소수점 이하 부분만을 테이블의 크기 M과 곱하여 그 정수 값을 주소로 사용

해싱 함수 종류 - III

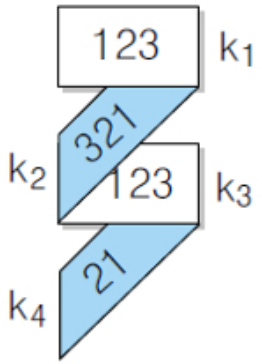
- **접지 함수**
  - 키의 비트 수가 해시 테이블 인덱스의 비트 수보다 큰 경우에 주로 사용
  - 이동 접지 함수
    - 각 분할 부분을 이동시켜서 오른쪽 끝자리가 일치하도록 맞추고 더하는 방법

예) 해시 테이블 인덱스가 3자리이고 키 값 k가 12312312312인 경우



해싱 함수 종류 - IV

- **경계 접지 함수**
  - 분할된 각 경계를 기준으로 접으면서 서로 마주보도록 배치하고 더하는 방법
  - 예) 해시 테이블 인덱스가 3자리이고 키 값 k가 12312312312인 경우



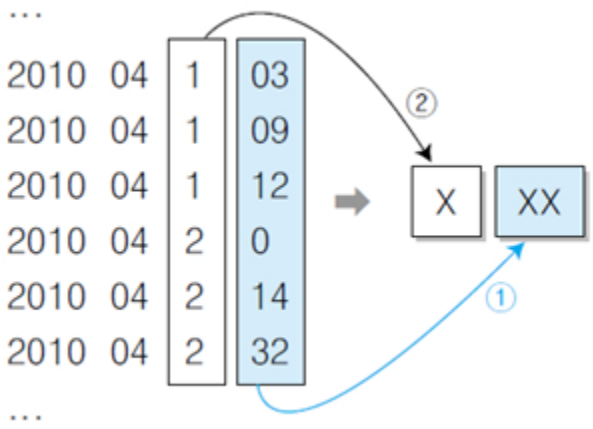
k1	123
k2	321
k3	123
k4	21

```
 k1      123
 k2      321
 k3      123
+k4       21
--------------
          588
```

k1	k2	k3	k4
123	321	123	21

해싱 함수 종류 - V

- **숫자 분석 함수**
  - 키 값을 이루고 있는 각 자릿수의 분포를 분석하여 해시 주소로 사용
  - 각 키 값을 적절히 선택한 진수로 변환한 후에 각 자릿수의 분포를 분석하여 가장 편중된 분산을 가진 자릿수는 생략하고, 가장 고르게 분포된 자릿수부터 해시 테이블 주소의 자릿수만큼 차례로 뽑아서 만든 수를 역순으로 바꾸어서 주소로 사용
  - 예) 키 값이 학번이고 해시 테이블 주소의 자릿수가 3자리인 경우



해싱 함수 종류 - VI

SUNG KYUN KWAN UNIVERSITY

- **진법 변환 함수**
  - 키 값이 10진수가 아닌 다른 진수일 때, 10진수로 변환하고 해시 테이블 주소로 필요한 자릿수만큼만 하위자리의 수를 사용하는 방법

- **비트 추출 함수**
  - 해시 테이블의 크기가 2k일 때 키 값을 이진 비트로 놓고 임의의 위치에 있는 비트들을 추출하여 주소로 사용하는 방법
  - 이 방법에서는 충돌이 발생할 가능성이 많으므로 테이블의 일부에 주소가 편중되지 않도록 키 값들의 비트들을 미리 분석하여 사용해야 한다.

오버플로우 처리 방법

SUNG KYUN KWAN UNIVERSITY

- 선형 개방 주소법 (선형 조사법(linear probing))
  - 해싱 함수로 구한 버킷에 빈 슬롯이 없어서 오버플로우가 발생하면, 그 다음 버킷에 빈 슬롯이 있는지 조사한다.

    - 빈 슬롯이 있으면 - 키 값을 저장
    - 빈 슬롯이 없으면 - 다시 그 다음 버킷을 조사
    - 이런 과정을 되풀이 하면서 해시 테이블 내에 비어있는 슬롯을 순차적으로 찾아서 사용하여 오버플로우 문제를 처리하는 방법

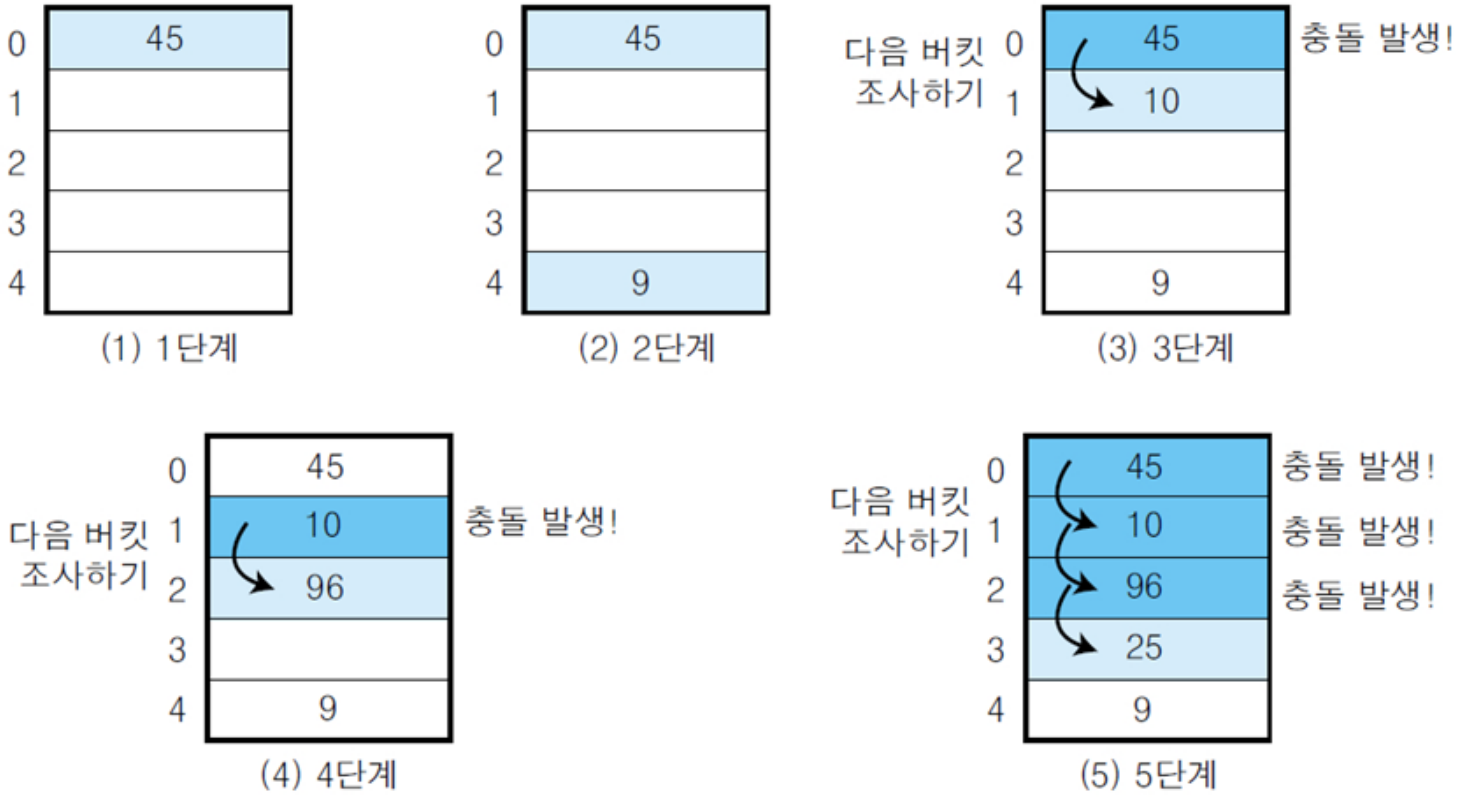
오버플로우 처리예) 선형 개방 주소법일 경우 - I

SUNG KYUN KWAN UNIVERSITY

- 해시 테이블의 크기 : 5
- 해시 함수 : 제산함수 사용. 해시 함수 h(k) = k mod 5

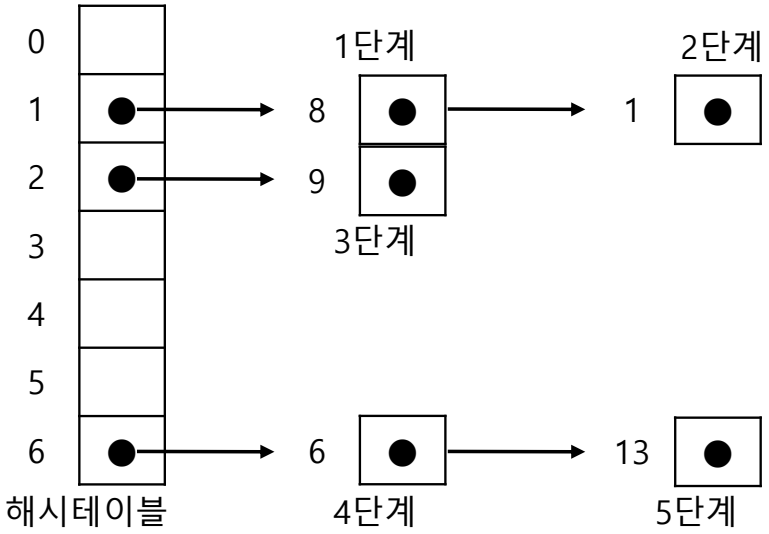
- 저장할 키 값 : {45, 9, 10, 96, 25}
  - 키 값 45 저장 : h(45) = 45 mod 5 = 0 ⇒ 해시 테이블 0번에 45 저장
  - 키 값 9 저장 : h(9) = 9 mod 5 = 4 ⇒ 해시 테이블 4번에 9 저장
  - 키 값 10 저장 : h(10) = 10 mod 5 = 0 ⇒ 충돌 발생!
    ⇒ 다음 버킷 중에서 비어있는 버킷 1에 10 저장
  - 키 값 96 저장 : h(96) = 96 mod 5 = 1 ⇒ 충돌 발생!
    ⇒ 다음 버킷 중에서 비어있는 버킷 2에 96 저장
  - 키 값 25 저장 : h(25) = 25 mod 5 = 0 ⇒ 충돌 발생!
    ⇒ 다음 버킷 중에서 비어있는 버킷 3에 25 저장

오버플로우 처리예) 선형 개방 주소법일 경우 - II



오버플로우 처리예) 체이닝 - I

- 체이닝(Chaining)
  - 해시 테이블의 구조를 변경하여 각 버킷에 하나 이상의 키 값을 저장할 수 있도록 하는 방법
  - 오버플로우 문제를 연결 리스트로 해결
    - 각 버켓에 고정된 슬롯이 할당되어 있지 않음
    - 각 버켓에, 삽입과 삭제가 용이한 연결 리스트 할당
    - 버켓 내에서는 연결 리스트 순차 탐색
  - (예) 크기가 7인 해시테이블에서
    - h(k)=k mod 7의 해시 함수 사용
    - 입력 (8, 1, 9, 6, 13) 적용



1단계 (8) : h(8) = 8 mod 7 = 1 (저장)
2단계 (1) : h(1) = 1 mod 7 = 1 (충돌발생->새로운 노드 생성 저장)
3단계 (9) : h(9) = 9 mod 7 = 2 (저장)
4단계 (6) : h(6) = 6 mod 7 = 6 (저장)
5단계 (13) : h(13) = 13 mod 7 = 6 (충돌 발생->새로운 노드 생성 저장)

오버플로우 처리예) 체이닝 - II

SUNG KYUN KWAN UNIVERSITY

- 해시 테이블의 크기 : 5
- 해시 함수 : 제산함수 사용. 해시 함수 h(k) = k mod 5
- 저장할 키 값 : {45, 9, 10, 96, 25}
  - 키 값 45 저장 : h(45) = 45 mod 5 = 0
    ⇒ 해시 테이블 0번에 노드를 삽입하고 45 저장
  - 키 값 9 저장 : h(9) = 9 mod 5 = 4
    ⇒ 해시 테이블 4번에 노드를 삽입하고 9 저장
  - 키 값 10 저장 : h(10) = 10 mod 5 = 0
    ⇒ 해시 테이블 0번에 노드를 삽입하고 10 저장
  - 키 값 96 저장 : h(96) = 96 mod 5 = 1
    ⇒ 해시 테이블 1번에 노드를 삽입하고 96 저장
  - 키 값 25 저장 : h(25) = 25 mod 5 = 0
    ⇒ 해시 테이블 0번에 노드를 삽입하고 25 저장

오버플로우 처리예) 체이닝 - III