14주차 3차시 해싱

【학습목표】

1. 해싱을 예를 들어 설명할 수 있다.
2. 해싱과 다른 검색 알고리즘과의 차이를 설명할 수 있다.

학습내용1 : 해싱의 이해

1. 해싱(Hashing)

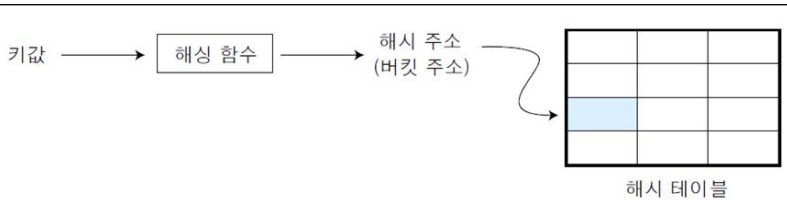
* 산술적인 연산을 이용하여 키가 있는 위치를 계산하여 바로 찾아가는 계산 검색 방식

* 검색 방법
- 키 값에 대해서 해싱 함수를 계산하여 주소를 구하고, 구한 주소에 해당하는 해시 테이블로 바로 이동
- 해당 주소에 찾는 항목이 있으면 검색 성공, 없으면 검색 실패

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

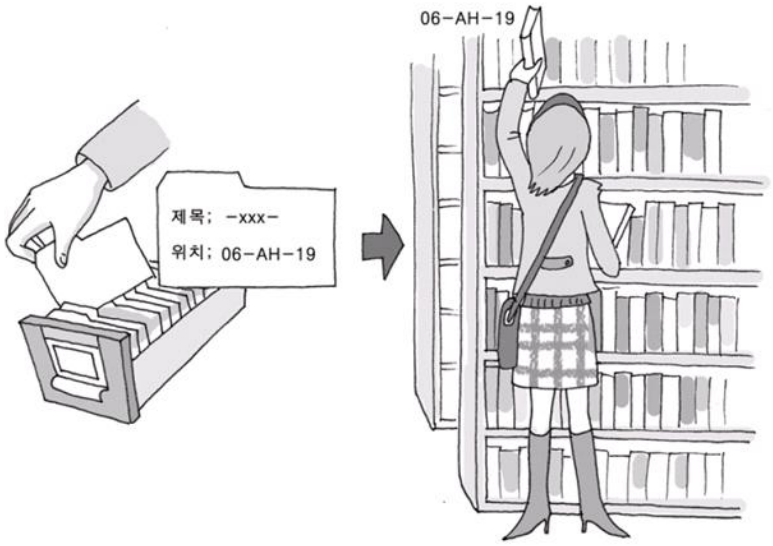
* 해시 테이블(hash table)
- 해싱 함수에 의해서 계산된 주소의 위치에 항목을 저장한 표

* 해싱 검색 수행 방법



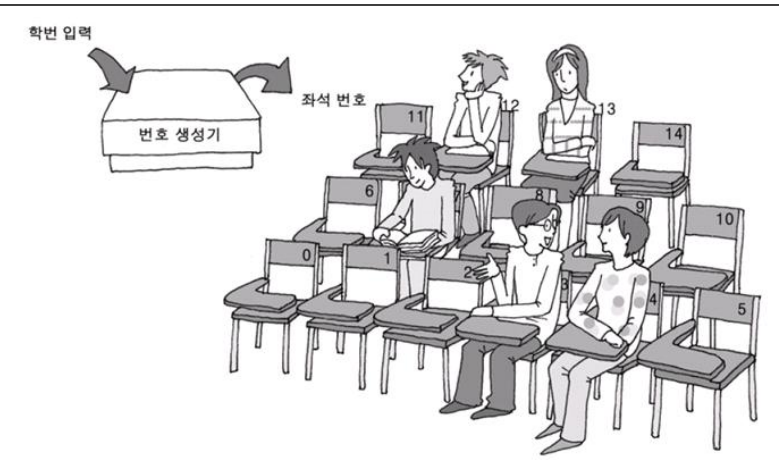
[그림 11-6] 해싱 검색의 수행 방법

* 해싱의 예제
- 도서관에서의 도서 검색 방식



[그림 11-7] 해싱의 예 : 위치 번호에 따른 도서 찾기

- 강의실 좌석 배정



[그림 11-8] 해싱의 예 : 학번에 따른 강의실 좌석 배정

2. 해싱 용어

* 충돌(collision)
- 서로 다른 키 값에 대해서 해싱 함수에 의해 주어진 버킷 주소를 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
- 버킷에 비어있는 슬롯이 없는 포화 버킷 상태에서 충돌이 발생하여 해당 버킷에 키 값을 저장할 수 없는 상태

* 버킷
- 몇 개의 레코드를 수용할 수 있는 블록 공간

[그림 11-9] 동거자의 예 : 같은 좌석에 앉은 짝꿍 [그림 11-10] 오버플로우의 예

* 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

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

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

학습내용2 : 해싱 함수

1. 해싱 함수의 조건

* 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.

2. 해싱 함수의 종류

* 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
  ×                     00110101 10100111
  ─────────────────────────────────────────
    00001011001111101001001011110001
                     해시주소
```

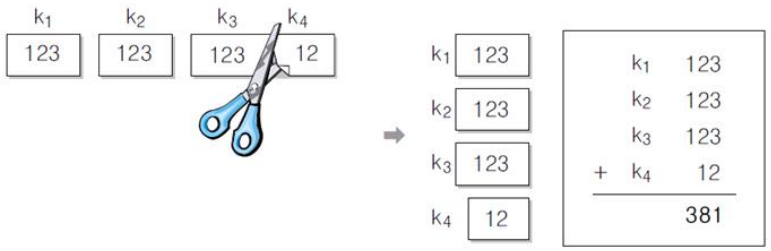
(해시주소: 11101001)

- 킷값에 대한 2진수를 제곱한 결과값에서 가운데 있는 8비트는 11101001이 된다 11101001의 10진수 값 233이 해시 테이블의 버킷 주소가 됨
- 제곱 결과값에서 주소로 사용하는 비트수는 해시 테이블의 버킷의 개수에 따라 결정됨
- 버킷이 256개가 있다면 버킷의 주소는 0~255까지 256개의 값을 사용. 256(2^8)의 값을 만들기 위해 필요한 최소의 비트수는 8비트가 되므로 제곱 결과값에서 8비트를 주소로 사용

* 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

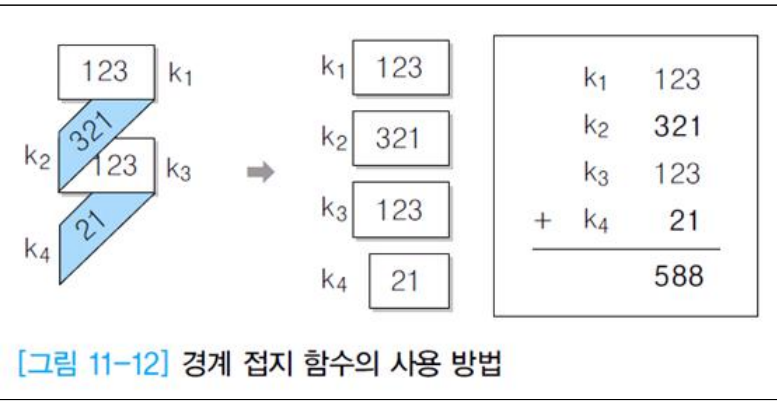
* 승산 함수
- 곱하기 연산을 사용하는 방법
- 키 값 k와 정해진 실수 α를 곱한 결과에서 소수점 이하 부분만을 테이블의 크기 M과 곱하여 그 정수 값을 주소로 사용

* 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



[그림 11-11] 이동 접지 함수의 사용 방법

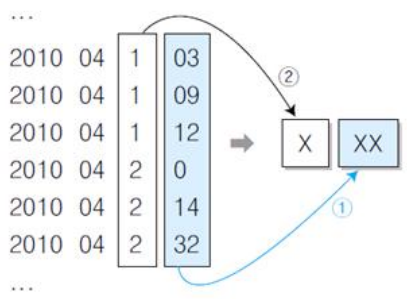
* 경계 접지 함수
- 분할된 각 경계를 기준으로 접으면서 서로 마주보도록 배치하고 더하는 방법
- 예) 해시 테이블 인덱스가 3자리이고 키 값 k가 12312312312인 경우



[그림 11-12] 경계 접지 함수의 사용 방법

* 숫자 분석 함수
- 키 값을 이루고 있는 각 자릿수의 분포를 분석하여 해시 주소로 사용하는 방법
- 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



[그림 11-13] 숫자 분석 함수의 사용 방법

* 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.

학습내용3 : 오버플로우 처리방법

1. 오버플로우 처리

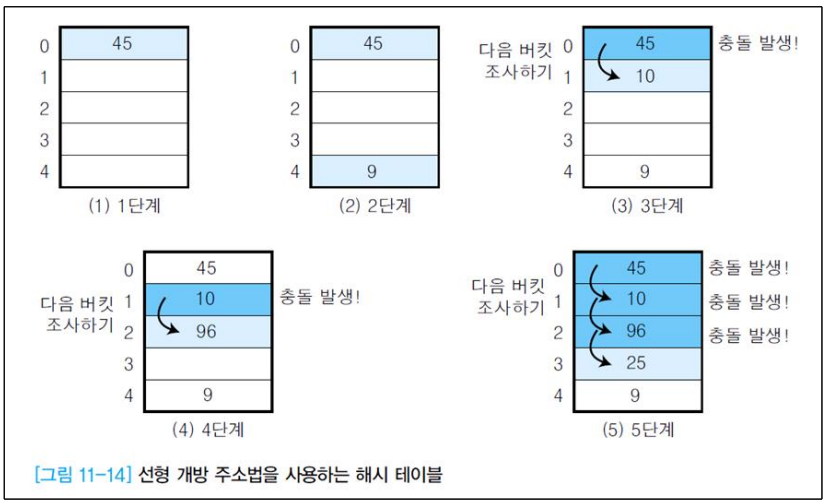
* 해시 테이블과 해싱 함수를 잘 선택하여 오버플로우가 발생하지 않도록 함

* 오버플로우 발생시 처리 방법
- 선형 개방 주소법 : 충돌이 일어난 킷값을 다른 비어 있는 버킷을 찾아 저장
- 체이닝 : 여러 개의 항목을 저장할 수 있도록 해시 테이블의 구조를 변경하는 방법

2. 선형 개방 주소법(선형 조사법)

* 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

* 선형 개방 주소법을 이용한 오버플로우 처리 예제
- 해시 테이블의 크기 : 5
- 해시 함수 : 제산함수 사용. 해시 함수 h(k) = k mod 5
- 저장할 키 값 : {45, 9, 10, 96, 25 }
  ① 키 값 45 저장 : h(45) = 45 mod 5 = 0 ⇒ 해시 테이블 0번에 45 저장
  ② 키 값 9 저장 : h(9) = 9 mod 5 = 4 ⇒ 해시 테이블 4번에 9 저장
  ③ 키 값 10 저장 : h(10) = 10 mod 5 = 0 ⇒ 충돌 발생!
  ⇒ 다음 버킷 중에서 비어있는 버킷 1에 10 저장
  ④ 키 값 96 저장 : h(96) = 96 mod 5 = 1 ⇒ 충돌 발생!
  ⇒ 다음 버킷 중에서 비어있는 버킷 2에 96 저장
  ⑤ 키 값 25 저장 : h(25) = 25 mod 5 = 0 ⇒ 충돌 발생!
  ⇒ 다음 버킷 중에서 비어있는 버킷 3에 25 저장



[그림 11-14] 선형 개방 주소법을 사용하는 해시 테이블

3. 체이닝

* 해시 테이블의 구조를 변경하여 각 버킷에 하나 이상의 키 값을 저장할 수 있도록 하는 방법

* 버킷에 슬롯을 동적으로 삽입하고 삭제하기 위해서 연결 리스트 사용
- 각 버킷에 대한 헤드노드를 1차원 배열로 만들고 각 버킷에 대한 헤드노드는 슬롯들을 연결 리스트로 가지고 있어서 슬롯의 삽입이나 삭제 연산을 쉽게 수행할 수가 있다.
- 버킷 내에서 원하는 슬롯을 검색하기 위해서는 버킷의 연결 리스트를 선형 검색한다.

* 체이닝을 이용한 오버플로우 처리 예제
- 해시 테이블의 크기 : 5
- 해시 함수 : 제산함수 사용. 해시 함수 h(k) = k mod 5
- 저장할 키 값 : {45, 9, 10, 96, 25 }
  ① 키 값 45 저장 : h(45) = 45 mod 5 = 0
  ⇒ 해시 테이블 0번에 노드를 삽입하고 45 저장
  ② 키 값 9 저장 : h(9) = 9 mod 5 = 4
  ⇒ 해시 테이블 4번에 노드를 삽입하고 9 저장
  ③ 키 값 10 저장 : h(10) = 10 mod 5 = 0

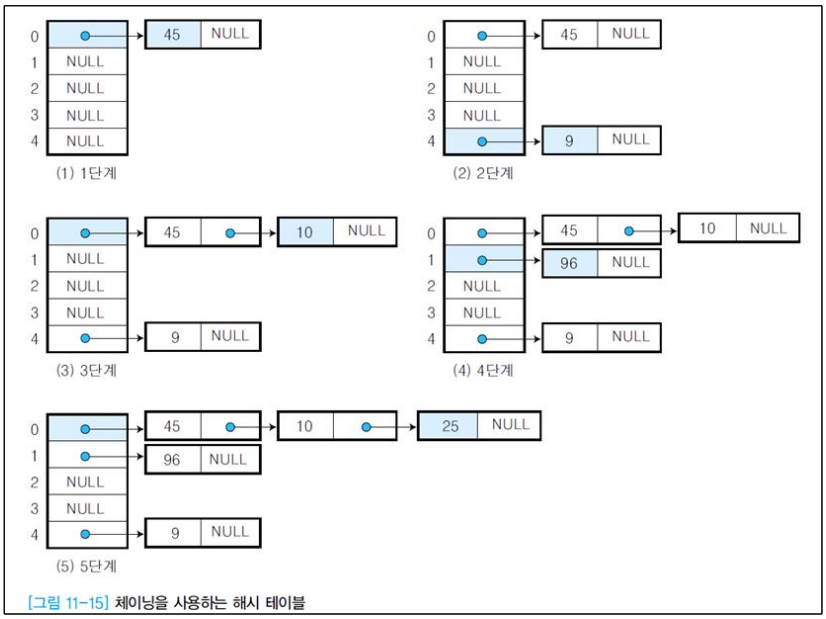
⇒ 해시 테이블 0번에 노드를 삽입하고 10 저장

④ 키 값 96 저장 : h(96) = 96 mod 5 = 1

⇒ 해시 테이블 1번에 노드를 삽입하고 96 저장

⑤ 키 값 25 저장 : h(25) = 25 mod 5 = 0

⇒ 해시 테이블 0번에 노드를 삽입하고 25 저장



[그림 11-15] 체이닝을 사용하는 해시 테이블

【학습정리】

1. 해시 테이블에서 오버플로우가 발생할 경우 해결하는 방법에는 충돌이 일어난 킷값을 다른 비어 있는 버킷을 찾아 저장하는 선형 개방 주소법과 여러 개의 항목을 저장할 수 있도록 해시 테이블의 구조를 변경하는 체이닝 방법이 있다.