



1


해시 테이블의 개념

해시 테이블의 개념

1 저장과 검색의 복잡도

- ▶ 검색 트리는 원소가 저장될 자리가 이미 트리에 존재하는 원소와 비교하여 결정되는 반면
해시 테이블은 원소가 저장될 자리가 **원소의 값**에 의해 결정되는 자료구조
- ▶ 저장된 자료와 비교하여 자리를 찾지 않고
단 한번의 계산으로 자신의 자리를 찾음

1 저장과 검색의 복잡도

- 
- ▶ 배열
 - $O(n)$
 - ▶ 이진 탐색 트리
 - 최악의 경우 $O(n)$
 - 평균 $O(\log n)$
 - ▶ 균형 잡힌 이진 탐색 트리(예 : 레드 블랙 트리)
 - 최악의 경우 $O(\log n)$
 - ▶ B-트리
 - 최악의 경우 $O(\log n)$
 - Balanced binary search tree보다 상수 인자가 작음
 - ▶ 해시 테이블
 - **평균 $O(1)$**

- ▶ 해시 함수를 이용하여 레코드가 저장되어 있는 주소를 직접 구하여 검색
- ▶ 키 (key)를 비교하지 않고 계산에 의해 검색하는 방법
- ▶ 삽입과 삭제가 빈번한 자료에 적합
- ▶ 키값을 가지고 있어서 데이터를 찾기 편리한 구조

2 해싱(Hashing)

- ▶ 평균 상수 시간에 삽입, 삭제, 검색
- ▶ 매우 빠른 응답을 요하는 응용에 유용
 - 예) 119 긴급구조 호출과 호출번호 관련 정보 검색
주민등록 시스템

3 해시 테이블(Hash Table)

- ▶ 해시 테이블은 최소 원소를 찾는 것과 같은 작업은 지원하지 않음
- ▶ 레코드를 한 개 이상 보관하는 버킷들의 집합
- ▶ 해시 테이블은 데이터가 저장되는 버킷들의 배열로 만들어지며 한 버킷은 하나 이상의 레코드를 수용할 수 있음

3 해시 테이블(Hash Table)

- ▶ 키값과 Value 데이터를 가진 구조를 해시 테이블이라 부르고 해시 테이블을 이용해서 탐색
- ▶ 키값을 해시 함수에 넣어서 계산하면 해시 테이블의 주소 값이 나오는데 계산된 값은 해시값 또는 해시 주소라고 함
 - 예) 주소와 우편번호의 관계

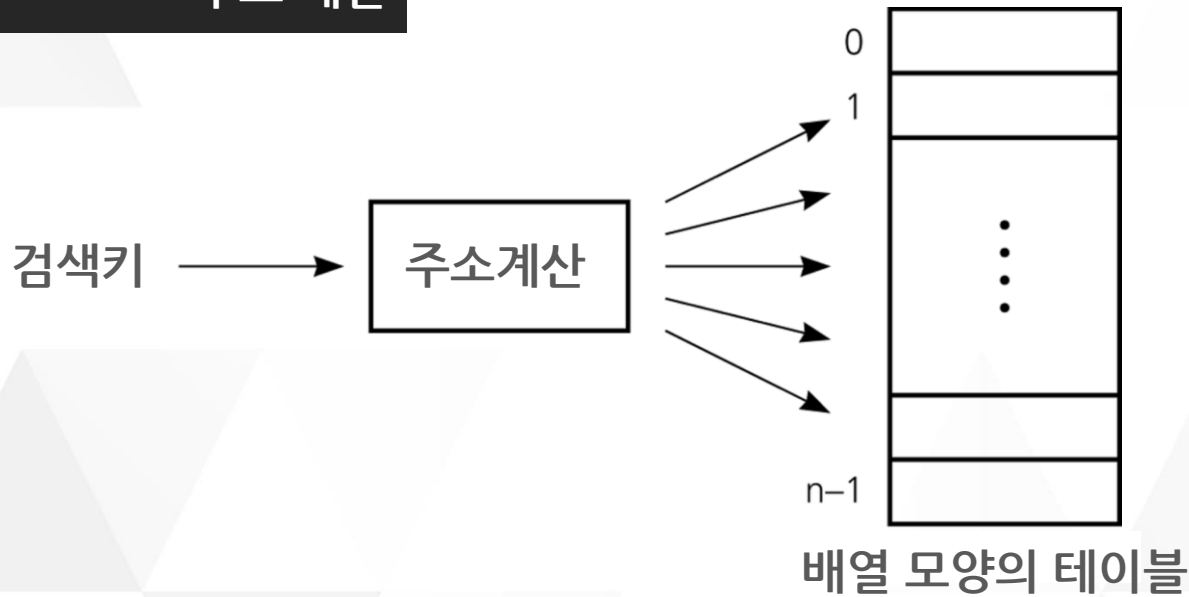
1

해시 테이블의 개념

3

해시 테이블(Hash Table)

주소 계산



3 해시 테이블(Hash Table)

해시 테이블의 예

- ▶ 크기 13인 해시 테이블에
5개의 원소가 저장된 예

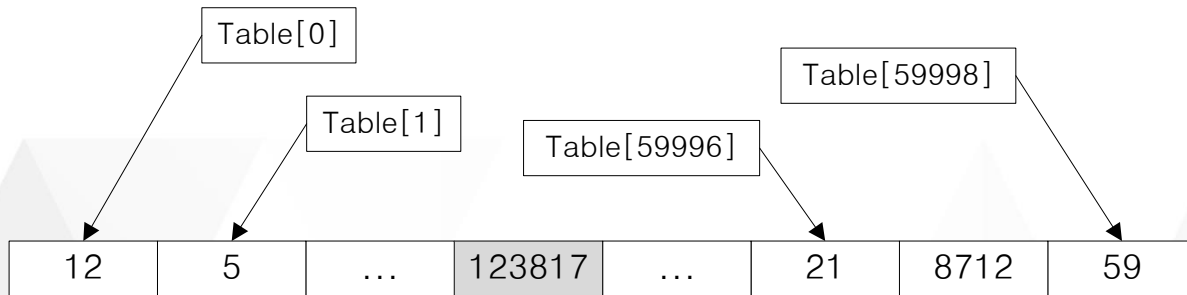
입력 : 25, 13, 16, 15, 7

해시함수 $h(x) = x \bmod 13$

| | |
|----|----|
| 0 | 13 |
| 1 | |
| 2 | 15 |
| 3 | 16 |
| 4 | |
| 5 | |
| 6 | |
| 7 | 7 |
| 8 | |
| 9 | |
| 10 | |
| 11 | |
| 12 | 25 |

해시 테이블의 필요성

- ▶ 다음 배열에서 데이터 123817을 찾으려면 어떻게 해야 할까? → 순차 탐색밖에 답이 없음

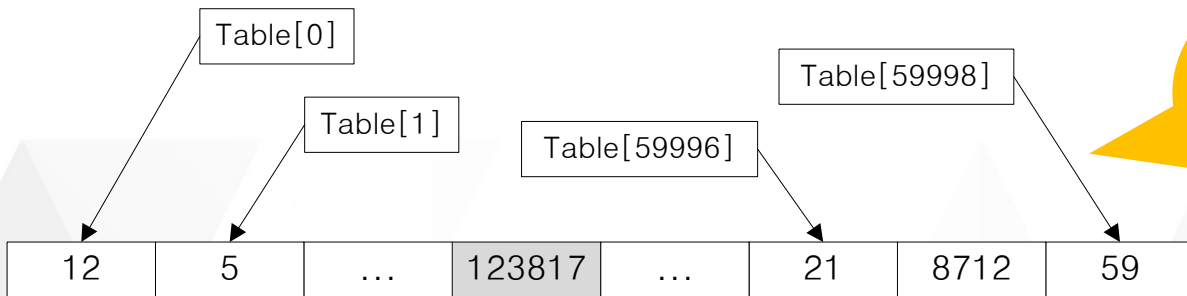


※출처: 뇌를 자극하는 알고리즘, 박상현, 한빛미디어

3 해시 테이블(Hash Table)

해시 테이블의 필요성

- ▶ 다음 배열에서 데이터 123817을 찾으려면 어떻게 해야 할까? → 순차 탐색밖에 답이 없음



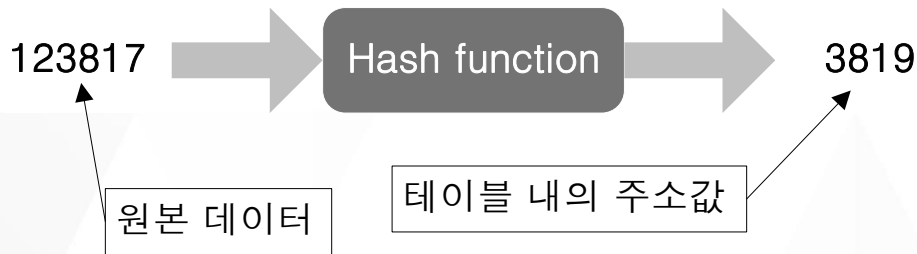
극한의 컴퓨팅 분야에서는
이와 같은 비효율은
허락되지 않음
다른 방법은 없을까?

※출처: 뇌를 자극하는 알고리즘, 박상현, 한빛미디어

3 해시 테이블(Hash Table)

해시 테이블의 필요성

- ▶ 데이터를 “해시(잘게 부수고 다시 뭉쳐서)”해서 테이블 내의 주소로 바꾸면 상수 시간 안에 탐색 가능

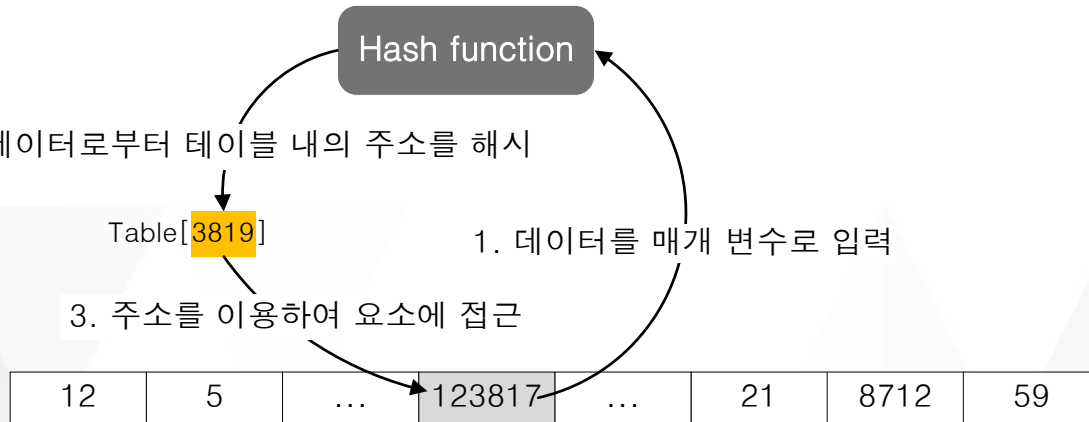


※출처: 뇌를 자극하는 알고리즘, 박상현, 한빛미디어

3 해시 테이블(Hash Table)

해시 테이블의 필요성

▶ 해시 함수를 이용하여 얻어낸 주소에 데이터를 읽고 씀



※출처: 뇌를 자극하는 알고리즘, 박상현, 한빛미디어

2 해시 함수

1 해시 함수의 성질

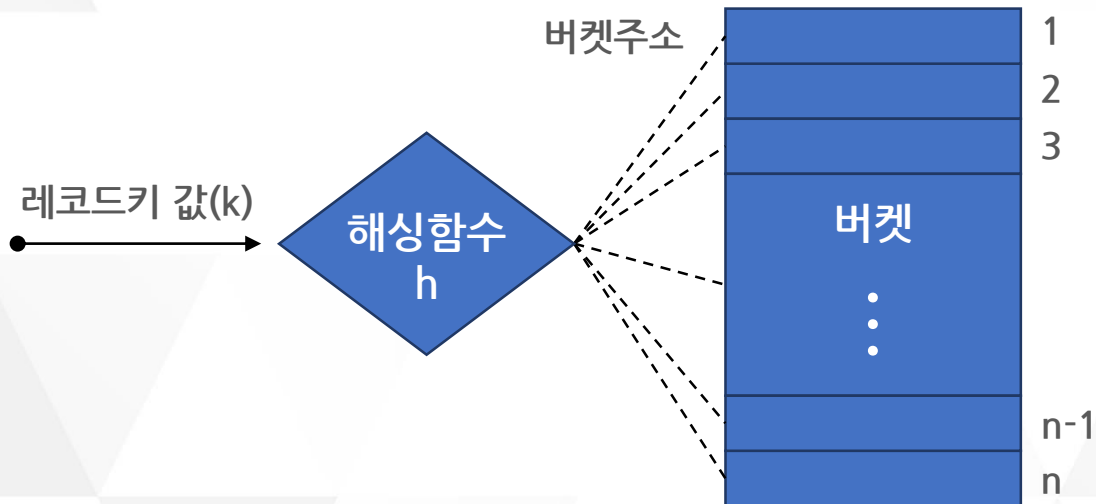
- ▶ 입력 원소가 해시 테이블에 골고루 저장되어야 함
- ▶ 계산이 간단해야 함
- ▶ 여러 가지 방법이 있으나 가장 대표적인 것은 나누기 방법과 곱하기 방법임
- ▶ 중간제곱법, 접지법도 있음
- ▶ 해시 테이블에 원소가 차 있는 비율(적재율)은 해시 테이블의 성능에 매우 중요한 영향을 미침
- ▶ 해시 테이블에 골고루 데이터가 존재해야 서로 다른 두 원소가 한 주소를 놓고 충돌할 확률이 작아짐

2 해시 함수(Hash Function)

- ▶ 하나의 문자열을 보다 빨리 찾을 수 있도록 주소에 직접 접근할 수 있는 짧은 길이의 값이나 키로 변환하는 알고리즘을 수식으로 표현한 것
- ▶ 주어진 키 값으로부터 레코드가 저장되어 있는 주소를 산출해 내는 수식

2 해시 함수(Hash Function)

- ▶ 해시 함수 $h(k)$ 는 어떤 k 에 대한 테이블 주소를 계산하기 위한 방법



2 해시 함수(Hash Function)

나누기 방법(Division Method)

- ▶ 해시 테이블 크기보다 큰 수를
해시 테이블 크기 범위에 들어오도록 수축시킴
- ▶ 해시 함수로 나눗셈을 이용하는 방법으로
키값 x 를 m 으로 나누고 나머지를 사용
- ▶ 키값 x 를 어떤 정해진 수 m 으로 나눈 나머지를
해시 주소로 사용

$$h(x) = x \bmod m$$

m : 해시 테이블 사이즈(대개 소수임)

2 해시 함수(Hash Function)

나누기 방법(Division Method)

▶ (예) $x=123$, $m=13$ 이면

$$\begin{aligned}h(x) &= 123 \bmod 13 \\ &= 6\end{aligned}$$

$x=100$, $m=12$ 이면

$$\begin{aligned}h(x) &= 100 \bmod 12 \\ &= 4\end{aligned}$$

2 해시 함수(Hash Function)

곱하기 방법(Multiplication Method)

- ▶ 먼저 입력값을 0과 1 사이의 소수로 대응시킨 다음 해시 테이블 크기 m 을 곱하여 0부터 $m-1$ 사이에 팽창시킴

$$h(x) = (xA \bmod 1) * m$$

$A: 0 < A < 1$ 인 상수

- m 은 굳이 소수일 필요 없음
따라서 보통 2^p 로 잡음(p 는 정수)

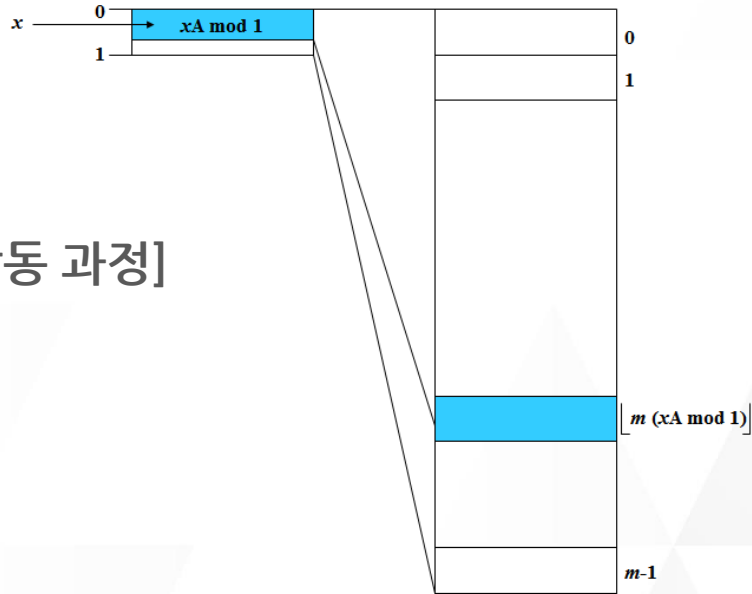
2

해시 함수

2

해시 함수(Hash Function)

곱하기 방법(Multiplication Method)



[곱하기 방법의 작동 과정]

2 해시 함수(Hash Function)

중간제곱법(Mid-square)

- ▶ 키값을 제공한 후에 그 결과의 중간에 있는 적당한 수의 비트를 취하여 버킷 주소로 하는 것
- ▶ 버킷 주소를 얻기 위해 사용되는 비트의 수는 해시 테이블의 크기에 달려 있음

2 해시 함수(Hash Function)

접지법(Folding)

- ▶ 종이를 접듯이 숫자를 접어 일정한 크기 이하의 수로 만드는 방법
- ▶ 키값을 마지막을 제외하고 같은 크기의 여러 부분으로 나누는데 각 부분들을 더하여 해시 주소를 만듦
- ▶ 각 부분을 더하는 방식은 Shift Folding과 Folding at the boundary 방법이 있음

2 해시 함수(Hash Function)

접지법(Folding)

- ▶ Shift Folding
 - 최하위 비트가 일치하도록 맞추고 더하는 방법
 - 각 부분의 오른쪽 끝을 맞추어 더한 값을 홈 주소로 결정하는 방식
- ▶ Folding at the boundary
 - 각 부분을 종이 접듯이 경계에서 겹친 다음 같은 자리에 위치한 수들을 더하는 방법
 - 각 부분의 경계선을 기점으로 잡아서 역으로 정렬하여 더하는 방식

2 해시 함수(Hash Function)

접지법(Folding)

(예)

식별자 $x=12320324111220$

3자씩 분리 123 203 241 112 20

Shift Folding

123

203

241

112

+ 20

699

Folding at the boundary

123

302

241

211

+ 20

897

3

해시 테이블의 충돌

1 충돌(Collision)

- ▶ 해시 테이블의 한 주소를 놓고
두 개 이상의 원소가 자리를 다투는 것
- ▶ 한 원소를 해싱해서 저장하려는데
다른 원소가 이미 그 자리를 차지한 상황

| | |
|----|----|
| 0 | 13 |
| 1 | |
| 2 | 15 |
| 3 | 16 |
| 4 | |
| 5 | |
| 6 | |
| 7 | 7 |
| 8 | |
| 9 | |
| 10 | |
| 11 | |
| 12 | 25 |

입력 : 25, 13, 16, 15, 7

← $h(29) = 29 \bmod 13 = 3$

29를 삽입하려 하자
이미 다른 원소가 차지하고 있음

해시함수 $h(x) = x \bmod 13$

- ▶ 충돌 해결 방법은 크게 두 가지가 있음
 - 체이닝(Chaining)
 - 개방주소 방법(Open Addressing)