

# Hash

한컴에듀케이션



# Hash?

- Search의 한가지 방법
- key값과 일치하는 data를 빠른 시간에 검색하기 위해 사용
- key 값은 고유 값 (ex. 회원id, 주민번호)
- indexing 기법 활용

## Search : 원하는 data의 실제 저장 정보를 찾아나가는 방법

Linear Search

Binary Search

Hash

Trie

etc

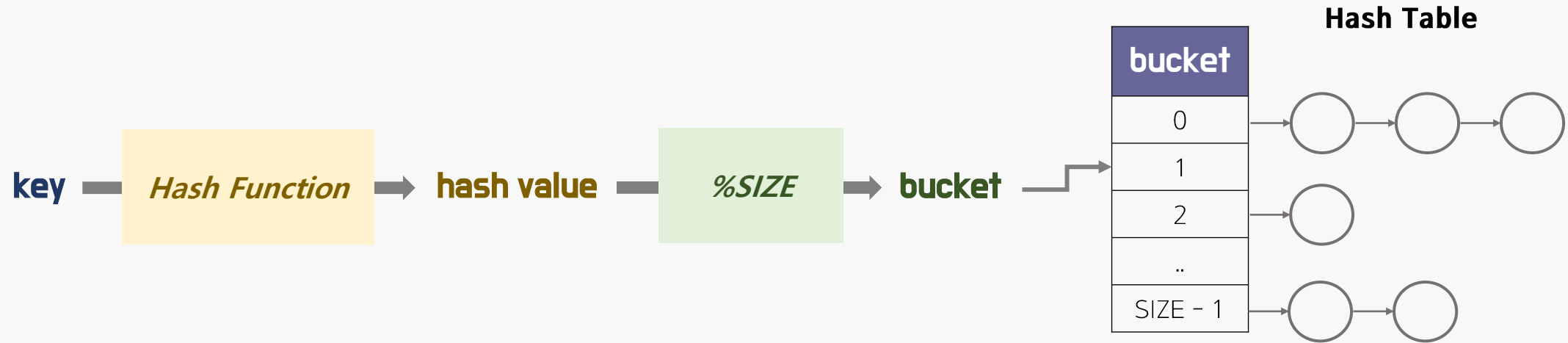
# Linear Search

data의 수, 검색 횟수가 적은 경우 사용



	name	no	age
1	son	7	3
2	roony	1	4
3	kane	2	65
4	terry	34	6
5	santez	5	7
6	herry	6	0
7	kim	7	9

# Hash



## key

int  
long long  
int[]  
char[]  
int[][]  
...

## hash value

unsigned int  
unsigned long long

## bucket

0 ~ SIZE - 1

## probing

구한 bucket에 저장된 모든 원소들을 검색하며 원하는 data를 찾는다.

동일한 key값은 항상 같은 bucket

다른 key값이어도 같은 bucket 가능 (충돌)

# Hash 설정

## 1. Hash Key

- data의 고유값을 나타낼 수 있는 정보로 설정

## 2. Hash Function

- hash value 생성 : *unsigned int* or *unsigned long long*
- 일반적으로 key값을 전부 활용하여 진법 변환
- key값의 일부만 활용하는 경우도 존재

## 3. Bucket

- 나눗셈 법 :  $hash\ value \% SIZE$

• 곱셈 법

## 4. Collision 처리 방식

- Chaining

bucket 별로 리스트화 하여 관리

• Linear Probing

## 5. Hash Table Size

- chaining 시

hash table에 등록되는 data수 이상의 2의 제곱수

• linear probing 시

최소 data수의 두배 이상

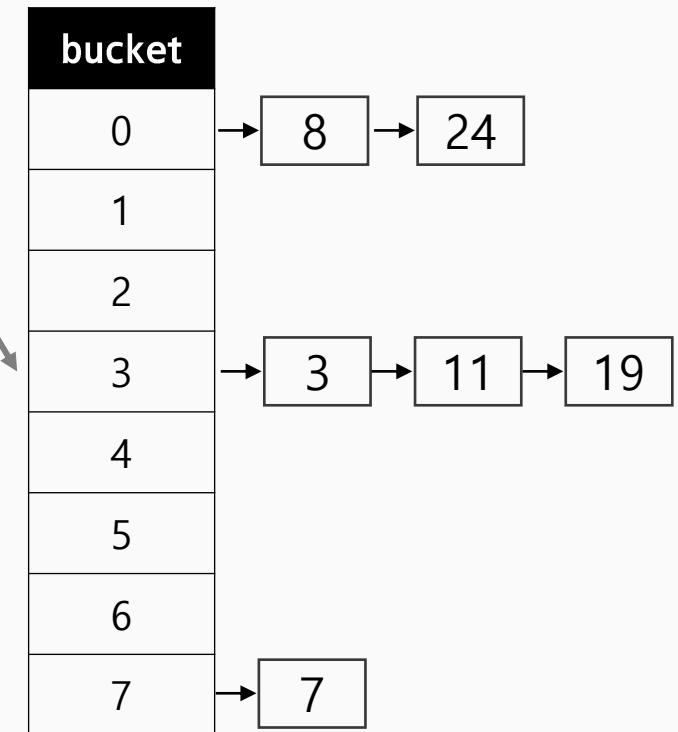
# Example 1

SIZE = 8  
key 값 = 정수 한 개



key	hash value	bucket
3	3	3
8	8	0
7	7	7
11	11	3
19	19	3
24	24	0

Hash Table



# Example2

SIZE = 8

key 값 = 정수 한 개

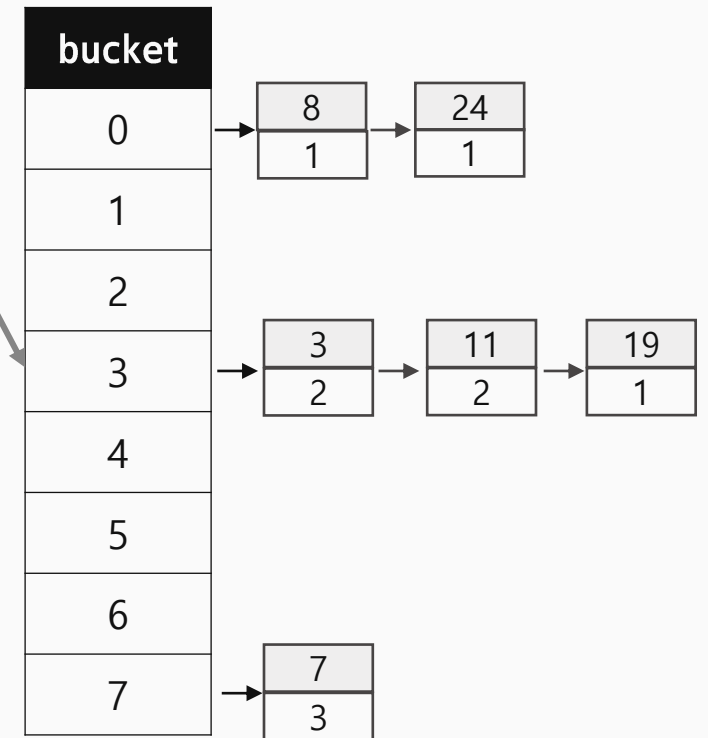
value 값 = 검색 횟수



검색 : 3, 8, 11, 11, 3, 19, 24, 7, 7, 7

key	hash value	bucket
3	3	3
8	8	0
7	7	7
11	11	3
19	19	3
24	24	0

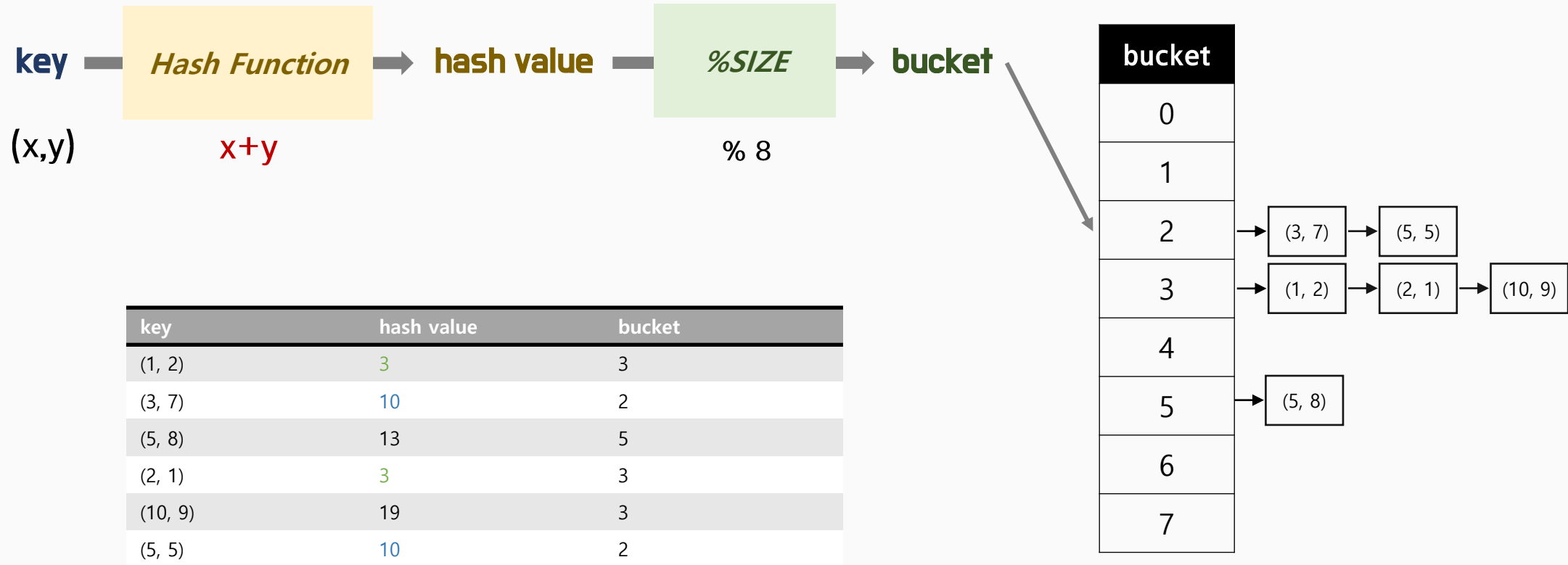
Hash Table



# Example3

SIZE = 8

key 값 = (0~99,999 , 0~99,999) 범위의 (x, y) 좌표



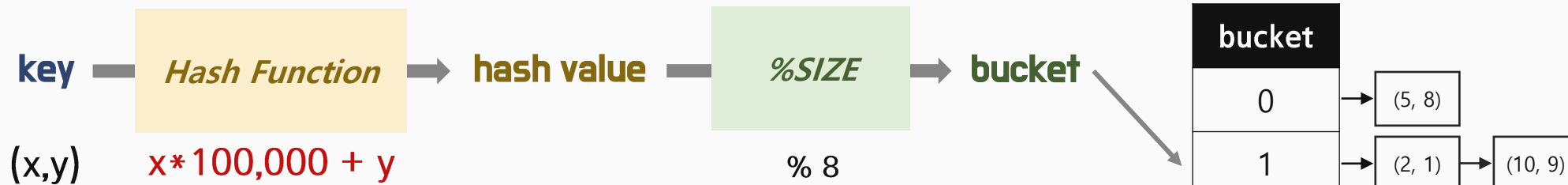


# Example4

SIZE = 8

key 값 = (0~99,999 , 0~99,999) 범위의 (x, y) 좌표

Hash Table



key	hash value	bucket
(1, 2)	10,002	2
(3, 7)	30,007	7
(5, 8)	50,008	0
(2, 1)	20,001	1
(10, 9)	100,009	1
(5, 5)	50,005	5

bucket	
0	→ (5, 8)
1	→ (2, 1) → (10, 9)
2	→ (1, 2)
3	
4	
5	→ (5, 5)
6	
7	→ (3, 7)

# Hash Function

## hash value를 고유하게 생성하는 방법

- 일반적으로는 hash value를 최대한 고유하게 생성하여 충돌 확률을 낮춘다.
- 만약, hash value 범위가 unsigned long long 범위를 벗어나면 %연산이 들어가므로 고유하지 않음  
key값의 모든 정보를 사용하지 않으면 고유하지 않음

### 1. 개수가 고정되어 있을 때, 수의 범위를 0부터로 맞춰주고 (max값+1) 진법 변환

- -100 ~ 100 범위의 1개 정수
- 0 ~ 100 범위의 4개 정수
- -100 ~ 100 범위의 3개 정수
- 소문자 10자리

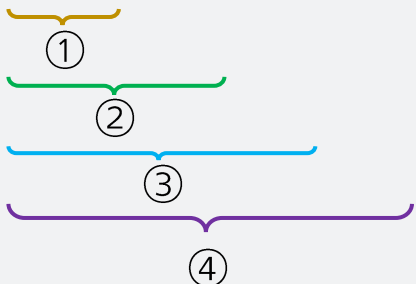
### 2. 개수가 고정되지 않을 때, 수의 범위를 1부터로 맞춰주고 (max값+1) 진법 변환

- 소문자 4~10자리
- 대소문자 4~10자리

# Hash Function 구현

## Hornor's method 활용

$k$ 진법  $\rightarrow$  10진법

$$5k^3 + 3k^2 + 2k^1 + 4k^0$$
$$= (((0*k+5)*k+3)*k+2)*k+4$$


hash=0

- ① hash=hash\*k + 5
- ② hash=hash\*k + 3
- ③ hash=hash\*k + 2
- ④ hash=hash\*k + 4

## Example code

key값: 0 ~  $k-1$  범위의 정수  $n$ 개

```
ull hashFunc(int key[]) {  
    ull hash = 0;  
    for (int i = 0; i < n; i++)  
        hash = hash * k + key[i];  
    return hash;  
}
```