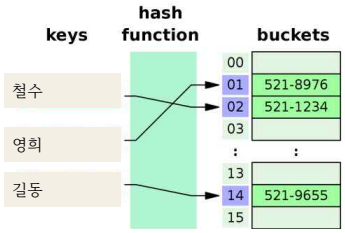
◉ 해시테이블



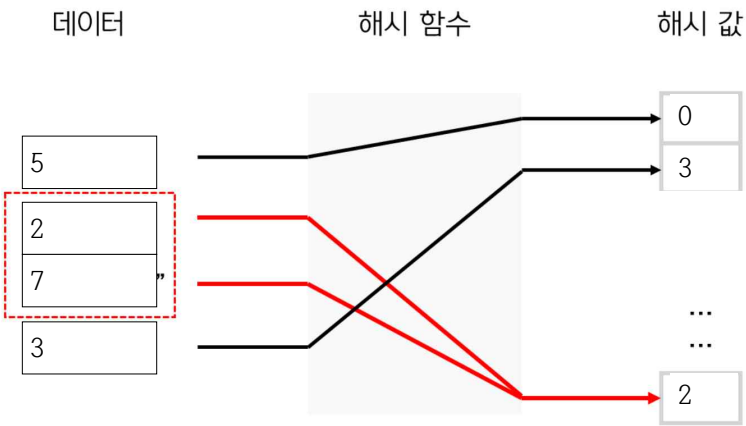
해시테이블이란 해시함수를 이용해 키를 값에 매핑하는 자료구조입니다.

◉ 해시함수란 ?

임의 크기 데이터를 고정 크기 값으로 매핑하는데 사용할 수 있는 함수를 말합니다.
쉽게 말해 임의의 값을 넣어도 예상 크기 내에서 결과가 나오는 함수를 말하는데요.
예를 들면 '나머지를 반환하는 함수'가 좋은 해시 함수의 예입니다.

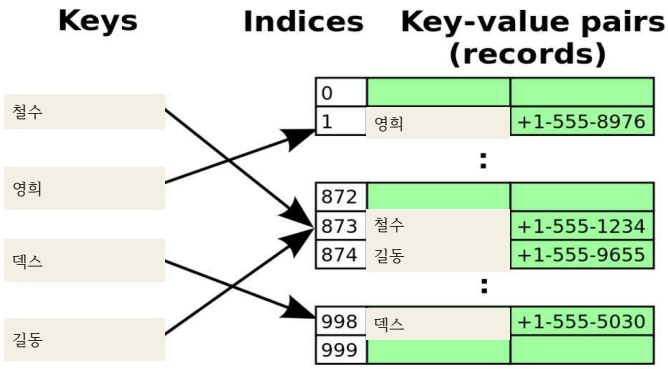
```
public int  modFive( n) {
   return  n% 5;      // 0~4 정해진 범위내에서의 결과발생
}
System.out.println( modFive(1) );
System.out.println( modFive(2) );
System.out.println( modFive(3) );
System.out.println( modFive(4) );
System.out.println( modFive(5) );
System.out.println( modFive(6) );
```

◉ 충돌

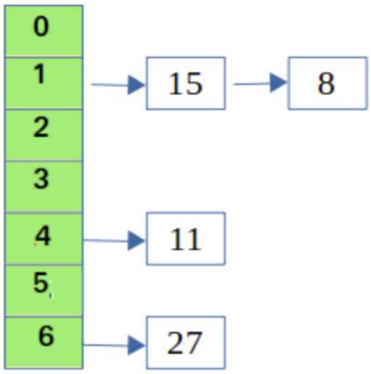


- 아무리 좋은 해시 함수라도 충돌을 피하기는 어려운데요!
- 입력값이 달라도 똑같은 결과가 나오면 충돌 이라고 합니다.
- 이를 다루는 방식은 체이닝(Chaining), 오픈 어드레싱(Open Addressing)이 있는데요.

- 체이닝은 충돌 발생 시 '연결'해가는 것이고, 오픈 어드레싱은 탐사를 통해 '빈 공간을 찾아나서는' 방식입니다.



오픈 어드레싱



체이닝

```
public class HashNode {
    public int key;
    public int value;
    public HashNode next;

    public HashNode(int key, int value) {
        this.key = key;
        this.value = value;
        this.next = null;
    }
}

public class HashTable {
    private int size;
    private HashNode[] table;

    public HashTable(int size) {
        this.size = size;
        this.table = new HashNode[size];
    }

    private int _hashFunction(int key) {
        return key % size;
    }

    public void put(int key, int value) {
        int index = _hashFunction(key);
        if (table[index] == null) {
            table[index] = new HashNode(key, value);
        } else {
            HashNode node = table[index];
            while (node.next != null) {
                node = node.next;
            }
            node.next = new HashNode(key, value);
        }
    }

    public int get(int key) {
        int index = _hashFunction(key);
        HashNode node = table[index];
        while (node != null) {
            if (node.key == key) {
                return node.value;
            }
            node = node.next;
        }
        return -1;
    }

    public void remove(int key) {
        int index = _hashFunction(key);
        HashNode node = table[index];
        HashNode prevNode = null;
        while (node != null) {
            if (node.key == key) {
                if (prevNode == null) {
                    table[index] = node.next;
                } else {
                    prevNode.next = node.next;
                }
                return;
            }
            prevNode = node;
            node = node.next;
        }
    }
}
```