**CS 스터디 알고리즘**

**기본 동작 :** [[그림으로 개념을 이해하는 알고리즘] Hash Table (해시테이블) (tistory.com)](https://jtoday.tistory.com/73)

**해시 정의 :** 임의의 크기를 가진 데이터를 고정된 크기의 데이터로 변화 시켜 저장하는 것,

키에 대한 해시값을 구하는 과정을 hashing(해싱)이라고 하며 이때 사용하는 함수를 해시함수라 한다.

**해시 함수 :** 키를 고정된 길이의 hash로 변경해주는 역할을 한다. -> 이 과정을 **hashing** 이라 함

**해시값 만들기**

**해시테이블의** 크기 = M **원소**는 k라 가정

**1. 나눗셈법 :** 원소를 해시테이블의 크기로 나누어 나머지 값을 테이블의 주소로 사용하는 방법 -> 테이블의 크기보다 원소의 개수가 많으면 충돌이 일어난다. -> **h(k) = k%m**

**2. 곱셈법 :**

0<A<1 인 A 에 대해서 다음과 같이 구할 수 있다.

h**(k)=N(k\*A\*mod(%) 1) \* m -> m은 얼마든 상관없지만 보통 2의 제곱수로 정한다.**

kAmod1 kA 의 소수점 이하 부분을 말하며 이를 NN 에 곱하므로 0부터 NN 사이의 값이 된다. 이 방법의 장점은 NN 이 어떤 값이더라도 잘 동작한다는 것이며 A 를 잘 잡는 것이 중요하다.

**3. Universal Hasing : //** [[Algorithm] 유니버설 해싱 (Universal Hashing) (tistory.com)](https://coding-wonderland.tistory.com/14)

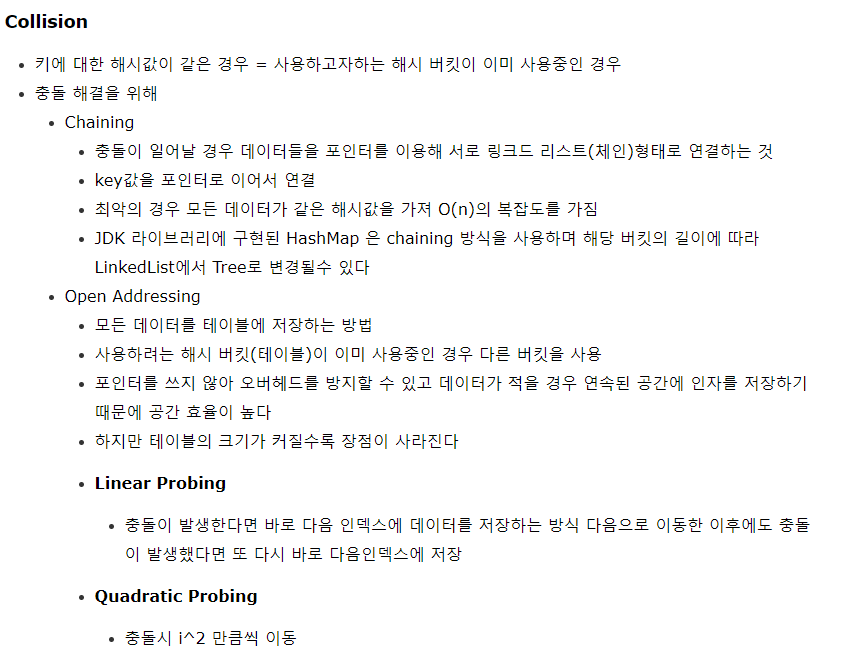
-> 같은 자리에 여러 개의 키가 해시되는 것을 막기 위하여 실제 저장되는 키들과 독립적인 해시 함수를 **무작위**로 선택하는 것 -> **실행 중 계속 무작위가 아니라 프로그램이 돌 때마다 그 때마다 무작위로 선택해서 진행하므로 검색과 삭제하는데 상관이 없다.** -> Math.Random() 느낌

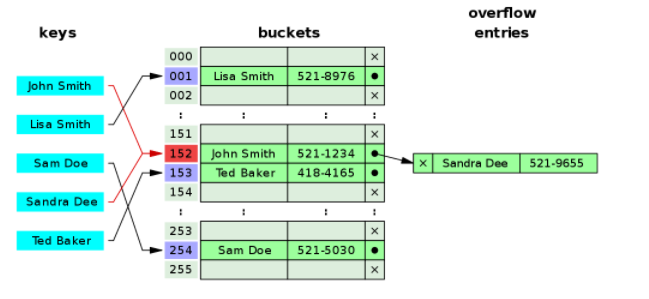
**해시 테이블**

**개념 :** 해시함수를 사용하여 변환한 값을 색인으로 삼아 키와 데이터를 저장하는 자료구조를 만한다. 기본연산으로는 **탐색, 삽입, 삭제**가 있다.

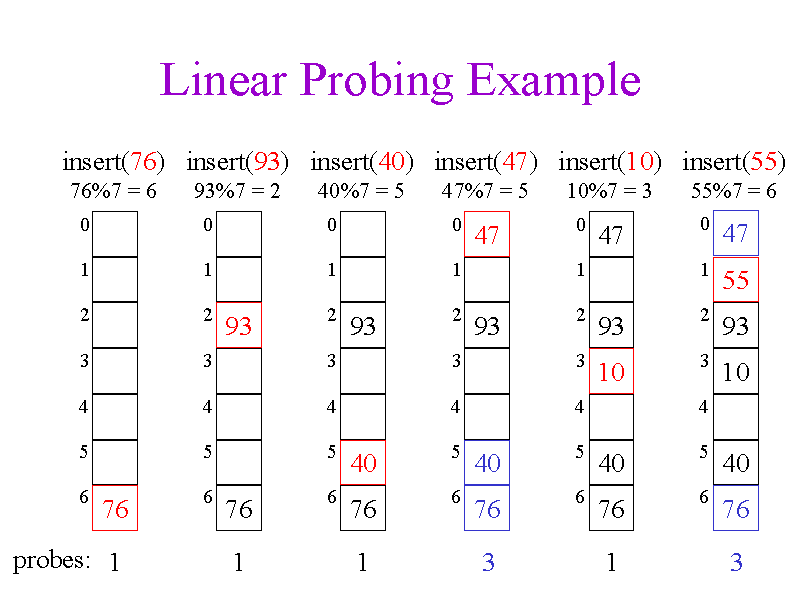
**적재율 :** 적재율이란 해시 테이블의 크기 대비, 키의 개수를 말한다. 즉, 키의 개수를 K, 해시 테이블의 크기를 N 이라고 했을 때 적재율은 K/N 이다. Direct Address Table은 키 값을 인덱스로 사용하는 구조이기 때문에 적재율이 1 이하이며 적재율이 1 초과인 해시 테이블의 경우는 반드시 충돌이 발생하게 된다. ex -> **배열 8개에 9개 넣을 수 없듯이**

**해시 충돌 해결 //** [[자료구조] 해시 테이블(Hash Table) | 배하람 블로그 (baeharam.github.io)](https://baeharam.github.io/posts/data-structure/hash-table/)

****

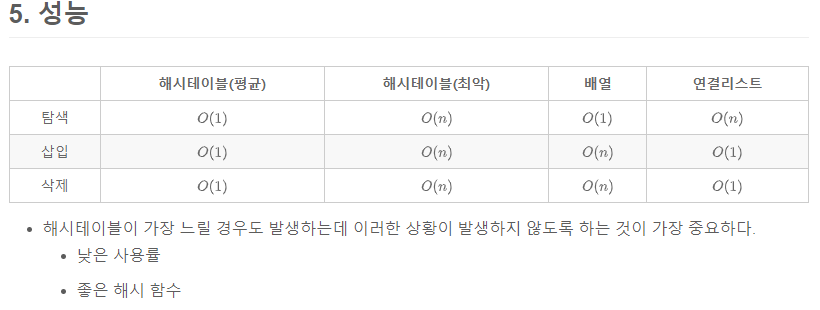
**Chainnig** ****

### 선형탐사(Linear Probing)



선형탐사는 가장 기본적인 충돌해결기법으로 위에서 설명한 기본적인 동작방식이다. 선형탐사는 바로 인접한 인덱스에 데이터를 삽입해가기 때문에 데이터가 밀집되는 클러스터링(Clustering) 문제가 발생하고 이로인해 탐색과 삭제가 느려지게 된다.

**평균 시간복잡도** : O(1) -> 해시값 자체를 index로 사용하기 때문에



**HashMap 사용법,** 선언

**Hash(Map)<Key, Value> 이름 = new HashMap<Key,Value>();** ex)<Integer,~>

값 추가

**이름.put(~,~);**

값 삭제

**이름.remove(key);** -> Key값에 해당하는 부분을 지운다. (오직 key값으로 remove)

**이름.clear();** -> 모든 값 제거

값 확인

**이름.get(key) ->** key 값에 해당하는 value 값을 반환한다.

크기 확인

**이름.size(); 이름.isEmpty();**

값 있는지 확인

**이름.containsKey(Key) 이름.contains(value)** true false 반환.

전체 key값 배열로 확인하기

**Hashtable.keySet();** ex) for(String a : 이름.keyset()){~~~}

**이름.getOrDefault(Key, defaultValue) ->** Key값을 해쉬맵에서 찾아 해당 Key의 Value값을 반환한다. 없으면 defaultValue 값을 반환한다.

**이름.putIfAbsent**(키값, 밸류값) 해당 키에 대한 값이 없을 때 추가

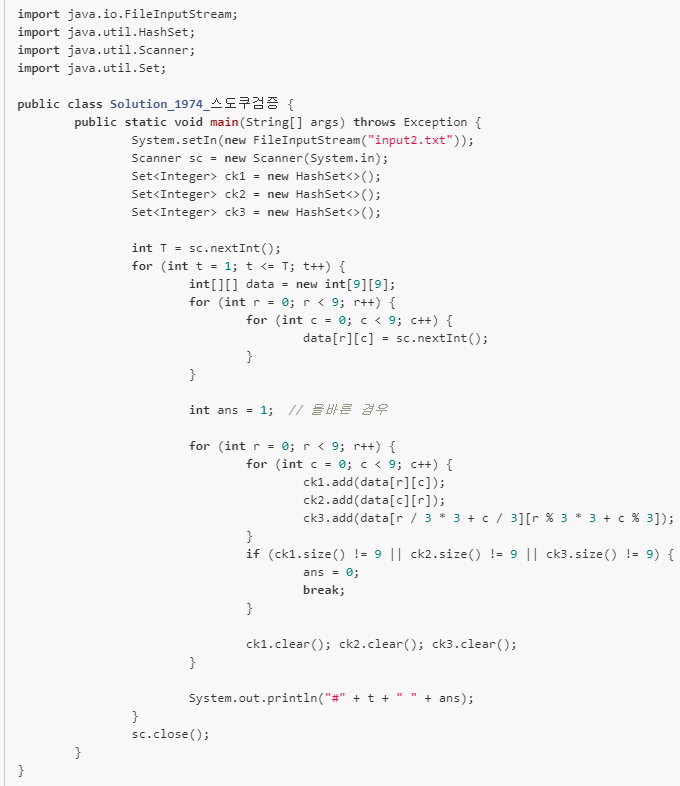
**EntrySet으로 Key Value값 모두 가져 오기**

for(Entry<String, Integer> elem : hMap.entrySet()){

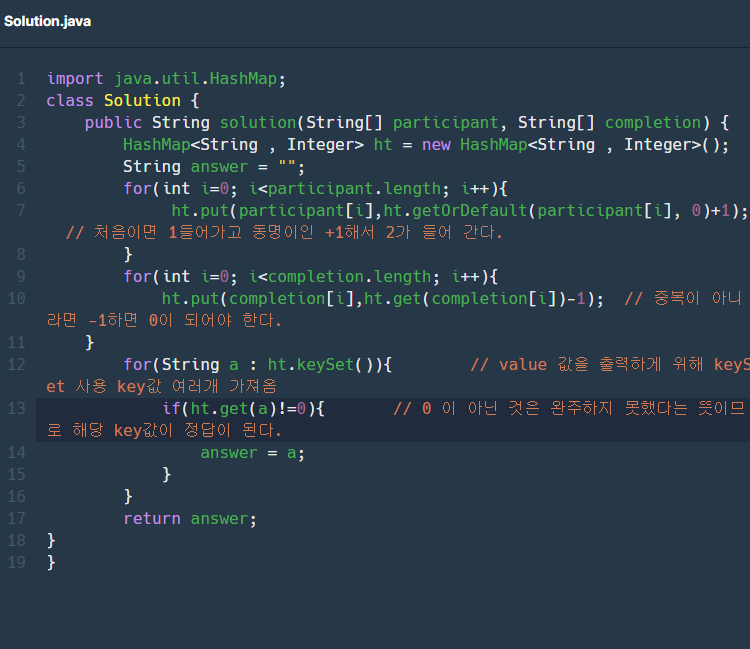
            System.out.println(elem.getKey());

        }

**자바 Hash<Set> Api 사용 : D2 -> 스도쿠문제**

****

**자바 HashMap 사용 -> 프로그래머스 레벨 1 완주하지 못한 선수**

****