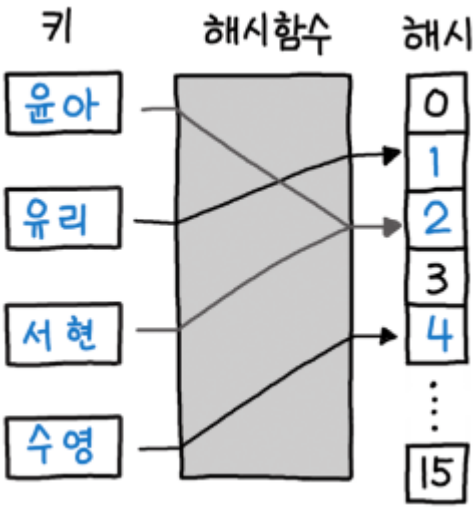
해시 테이블

1. 해시

- **해시 함수**란 임의 크기 데이터를 고정 크기 값으로 매핑하는 데 사용할 수 있는 함수를 말한다.

- 해시 테이블의 핵심은 해시 함수다. ABC, 1324BC, AF32B로 각각 3글자, 6글자, 5글자이지만, 화살표로 표시한 특정 함수를 통과하면 2바이트의 고정 크기 값으로 매핑된다. 여기서 화살표 역할을 하는 함수가 바로 해시 함수다.
  - ABC → A1
  - 1324BC → CB
  - AF32B → D5

- 해시 함수를 사용하는 것을 **해싱(Hashing)**이라 하며, 해싱은 정보를 가능한 한 빠르게 저장하고 검색하기 위해 사용하는 중요한 기법 중 하나다.
  - 성능 좋은 해시 함수들의 특징
    - 해시 함수 값 **충돌**의 최소화
    - 쉽고 빠른 연산
    - 해시 테이블 전체에 해시 값이 균일하게 분포
    - 사용할 키의 모든 정보를 이용하여 해싱
    - 해시 테이블 사용 효율이 높을 것



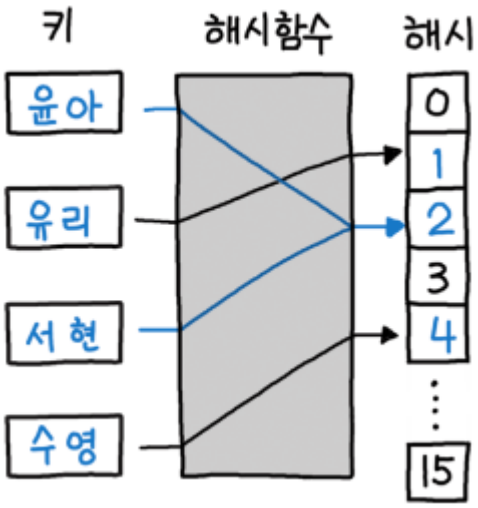
2. 해시 충돌

- 비둘기집 원리
  - n개 아이템을 m개 컨테이너에 넣을 때, n>m이라면 적어도 하나의 컨테이너에는 반드시 2개 이상의 아이템이 들어있다는 원리를 말한다.

  - 비둘기집 원리에 따라 9개의 공간이 있는 곳에 10개의 아이템이 들어온다면 반드시 1번 이상은 충돌이 발생하게 된다. 좋은 해시 함수라면 충돌을 최소화하여 단 1번의 충돌만 일어나게 하겠지만, 좋지 않은 해시 함수의 경우 심하면 9을 모두 충돌해서, 9개의 공간 중 1개밖에 사용하지 못할 수도 있다.

- 아래의 그림에서 '윤아'와 '서현'은 해시 값이 2로 같은 값이 되어 충돌이 발생했다. 충돌이 발생한다는 것은 그만큼 추가 연산을 필요로 하기 때문에 가급적 충돌은 최소화하는

것이 좋다.



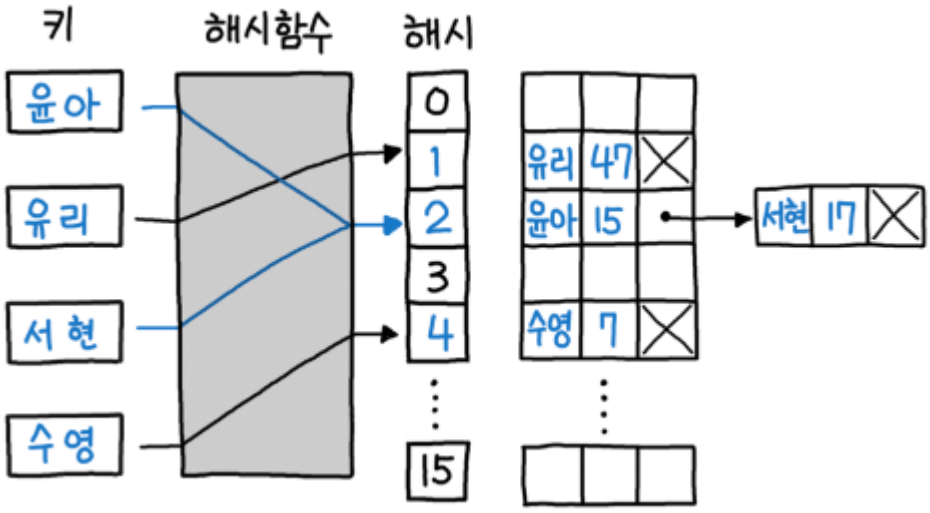
3. 충돌 해결법

1) 개별 체이닝 (Separate Chaining)

- 위의 그림을 표로 정리하면 다음과 같고, '윤아'와 '서현'을 해싱한 결과 충돌하고 있다.

키	값	해시	충돌 여부
윤아	15	2	충돌
유리	47	1	
서현	17	2	충돌
수영	7	4	

- 개별 체이닝은 충돌 발생 시 연결 리스트로 연결하는 방식이다. 개별 체이닝 방식은 인기가 높고, 원래 해시 테이블 구조의 원형이기도 하며 가장 전통적인 방식이다. 흔히 해시 테이블이라고 하면 바로 이 방식을 말한다.



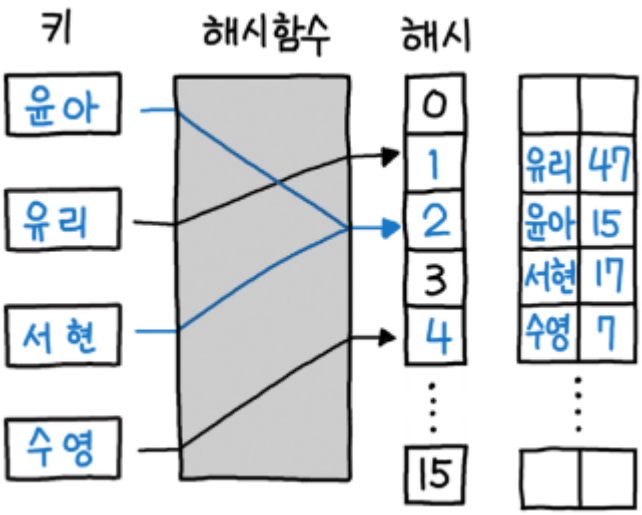
- 충돌이 발생한 '윤아'와 '서현'은 '윤아'의 다음 아이템인 '서현'인 형태로 서로 연결 리스트로 연결되었다.

- 개별 체이닝의 간단한 원리 요약
  1. 키의 해시 값을 계산한다.
  2. 해시 값을 이용해 배열의 인덱스를 구한다.
  3. 같은 인덱스가 있다면 연결 리스트로 연결한다.

2) 오픈 어드레싱 (Open Addressing)

- 오픈 어드레싱 방식은 충돌 발생 시 탐사를 통해 빈 공간을 찾아나서는 방식이다. 사실상 무한정 저장할 수 있는 체이닝 방식과 달리, 오픈 어드레싱 방식은 전체 슬롯의 개수 이상은 저장할 수 없다. 충돌이 일어나면 테이블 공간 내에서 탐사(Probing)를 통해 빈 공간을 찾아 해결하며, 이 때문에 개별 체이닝 방식과 달리, 모든 원소가 반드시 자신의 해시값과 일치하는 주소에 저장된다는 보장은 없다.

- 오픈 어드레싱 방식 중에서 가장 간단한 방식인 선형 탐사(Linear Probing)방식은 충돌이 발생할 경우 해당 위치부터 순차적으로 탐사를 하나씩 진행한다. 특정 위치가 선점되어 있으면 바로 그 다음 위치를 확인하는 식이다. 이렇게 탐사를 진행하다가 비어있는 공간을 발견하면 삽입하게 된다.

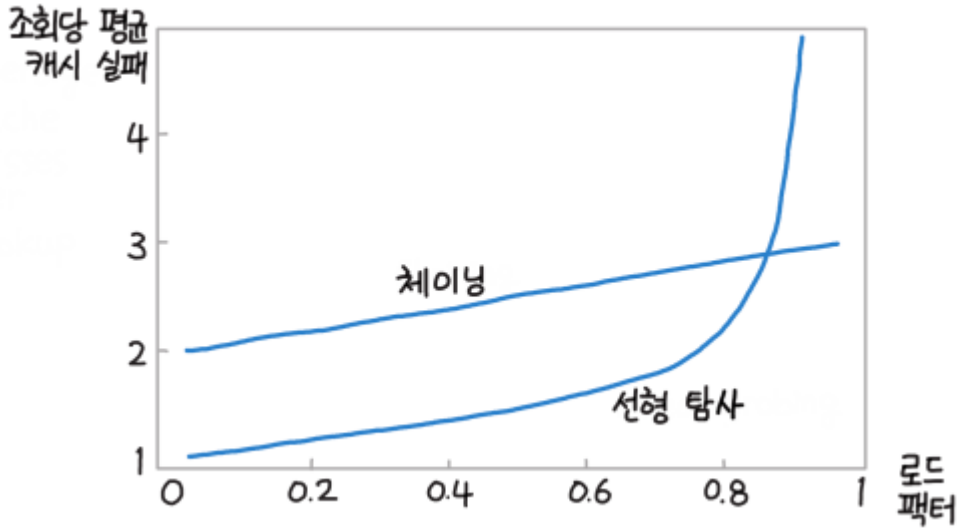


- 위의 그림에서 윤아 다음에 서현의 해시값이 동일한 2로 충돌이 발생했고, 다음 번 빈 위치를 탐색하며 그 다음 위치인 3에 서현이 들어가게 된다.
    - 선형 탐사의 한 가지 문제점은 해시 테이블에 저장되는 데이터들이 고르게 분포되지 않고 뭉치는 경향이 있다는 점이다. 해시 테이블 여기저기에 연속된 데이터 그룹이 생기는 이러한 현상을 **클러스터링(Clustering)**이라 하는데, 클러스터들이 점점 커지게 되면 인근 클러스터들과 서로 합쳐지는 일이 발생한다. 그렇게 되면 해시 테이블의 특정 위치에는 데이터가 몰리게 되고, 다른 위치에는 상대적으로 데이터가 거의 없는 상태가 될 수 있다. 이러한 현상은 탐사 시간을 오래 걸리게 하며, 전체적으로 해싱 효율을 떨어뜨리는 원인이 된다.

3) 파이썬의 해시 테이블 구현 방식은?

- 리스트와 함께 파이썬에서 가장 흔하게 쓰이는 자료형인 딕셔너리는 해시 테이블로 구현되어 있다. 면접 시 “해시 테이블로 구현된 파이썬의 자료형을 제시하라.”는 질문을 받는다면 주저 없이 딕셔너리라고 답할 수 있어야 한다.

- 파이썬의 해시 테이블은 충돌 시 **오픈 어드레싱** 방식으로 구현되어 있다.



- 위의 그림에서 볼 수 있듯이 오픈 어드레싱의 한 방식인 선형 탐사 방식은 일반적으로 체이닝에 비해 성능이 좋다. 그러나 슬롯의 80% 이상이 차게 되면 급격한 성능 저하가 일어난다. 따라서 파이썬과 같은 모던 언어들은 오픈 어드레싱 방식을 택해 성능을 높이는 대신, 로드 팩터를 작게 잡아 성능 저하 문제를 해결한다. 파이썬의 로드 팩터는 0.66으로 자바보다 작다.

언어	방식
C++	개별 체이닝
자바	개별 체이닝
파이썬	오픈 어드레싱