* 해시 테이블 (Hash Table)

: 연관배열 구조를 이용하여 키(key)에 결과값(value)을 저장하는 자료구조

: 빠르게 데이터를 검색할 수 있는 자료구조

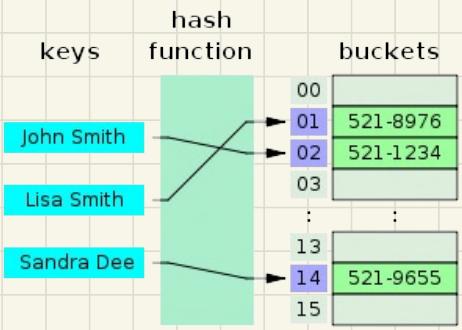
-> How? 내부적인 저장소인 버킷을 사용하여 데이터를 저장하기 때문

- 연관배열 구조(associative array)

: 키(key) 1개와 값(value) 1개가 1:1로 연관되어 있는 자료구조

-> 키(key)를 이용하여 값(value)을 도출할 수 있음

* 해시 테이블의 구조 (Hash Table Data Structure)



데이터 (key, Value) = ("John Smith", " 521-1234")를 크기가 16인 해시테이블에 저장

1) index = hash_function ("John Smith") % 16 <- index값 계산

2) array[index] = " 521-1234 " <- 전화번호 저장

- 키(key) : 고유한 값으로, 해시 함수의 input에 해당

다양한 길이의 값이 될 수 있음

- 해시함수(Hash Function) : 키를 해시로 바꿔주는 역할

키를 해시로 변경하여 저장소를 효율적으로 운영할 수 있도록 도움을 줌

-> 해시충돌(=서로 다른 키가 같은 해시가 되는 경우)이 일어나는 확률을 줄이는 함수를 만드는 것이 중요

- 해시(Hash) : 해시 함수의 결과물

저장소에서 값과 매칭되어 저장됨

- 값(value) : 저장소에 최종적으로 저장되는 값

키와 매칭되어 저장, 삭제, 검색, 접근이 가능해야 함

- 저장소(Bucket, Slot) : 실제 값이 저장되는 장소

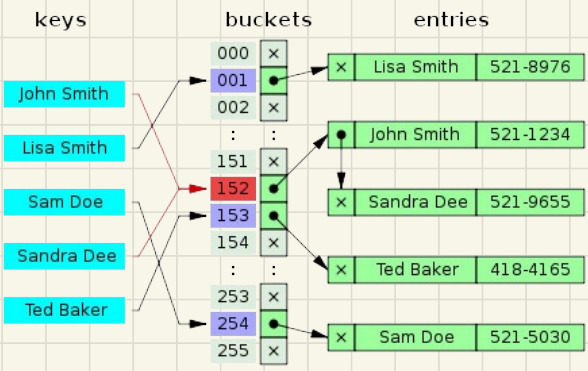
* 해시 충돌 (Hash Collision)

- 예시) 키 "John Smith"를 해시 함수를 돌려 나온 값과 키 "Sandra Dee"를 해시 함수를 돌려 나온 값이 동일하여 발생하는 현상

- 해시 함수로 해시를 만드는 과정에서 서로 다른 key가 같은 해시로 변경되면, 같은 공간에 다른 2개의 값이 저장
  -> key-value가 1:1로 매핑되어야 하는 해시 테이블의 특성에 위배
- 해시 충돌은 필연적으로 나타날 수 밖에 없음

* 해시 충돌 문제 해결방안

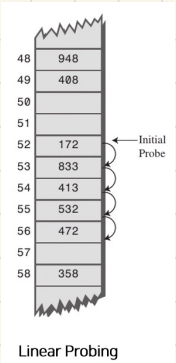
1) 분리 연결법 (Separate Chaining)

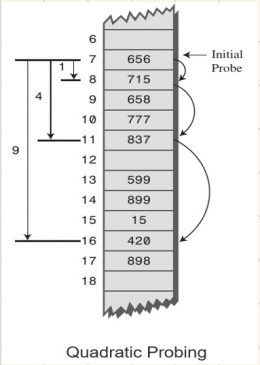


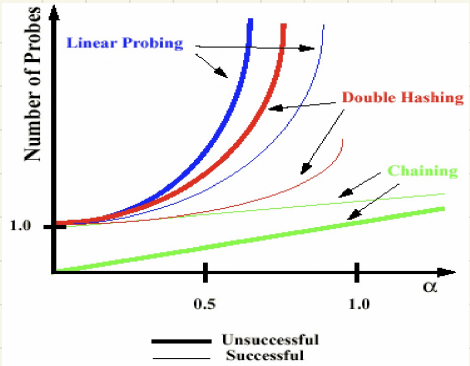
- 분리 연결법 : 동일한 버킷의 데이터에 대해 자료구조를 활용해 추가 메모리를 사용하여 다음 데이터의 주소를 저장하는 것

- 장점 : 해시 테이블의 확장 필요없이 간단하게 구현이 가능
  손쉽게 삭제할 수 있음
- 단점 : 데이터의 수가 많아지면 동일한 버킷에 연결되는 데이터가 많아져 해시의 효율성이 감소함

2) 개방 주소법 (Open Addressing)







- 개방 주소법 : 추가적인 메모리를 사용하는 분리 연결법과 달리 비어있는 해시 테이블의 공간을 활용하여 모든 원소들이 스스로의 hash 테이블에 저장하는 방식.

- 개방 주소법을 구현하기 위한 대표적인 방법

(1) Linear Probing (선형 탐색)

: 현재의 버킷 index로부터 고정폭 만큼씩 이동하여 차례대로 검색해 비어 있는 버킷에 데이터를 저장

(2) Quadratic Probing (제곱 탐색)

: 해시의 저장순서 폭을 제곱으로 저장하는 방식

예) 처음 충돌이 발생한 경우에는 1만큼 이동 -> 계속 충돌이 발생하면 2^2, 3^2 칸씩 옮기는 방식

(3) Double Hashing Probing (이중 해시)

: 해시된 값을 한번 더 해싱하여 해시의 규칙성을 없애버리는 방식

해시된 값을 한번 더 해싱하여 새로운 주소를 할당하기 때문에 다른 방법들보다 많은 연산을 하게 됨

* 해시테이블(HashTable) 시간복잡도

- 각각의 Key값은 해시함수에 의해 고유한 index를 가지게 되어 바로 접근할 수 있으므로 평균 O(1)의 시간복잡도로 데이터를 조회할 수 있음
- 데이터의 충돌이 발생한 경우, 분리 연결법에 연결된 리스트들까지 검색을 해야 하므로 O(N)까지 시간복잡도가 증가할 수 있음

* 해시 테이블의 장단점

- 장점

-> key-value가 1:1로 매핑되어 있기 때문에 삽입, 삭제, 검색의 과정에서 모두 평균적으로 O(1)의 시간복잡도를 가지고 있으므로 바로 접근 가능

- 단점

-> (1) 해시 충돌이 발생 (분리 연결법, 개방 주소법으로 해결)

(2) 순서/관계가 있는 배열에는 어울리지 않음

(3) 공간 효율성이 떨어짐 => 따라서 데이터가 저장되기 전에 저장공간을 미리 만들어 놔야함

(4) 해시 함수의 의존도가 높음

만약 해시함수가 복잡하다면 해시를 만들어 내는데 오래 걸릴 것