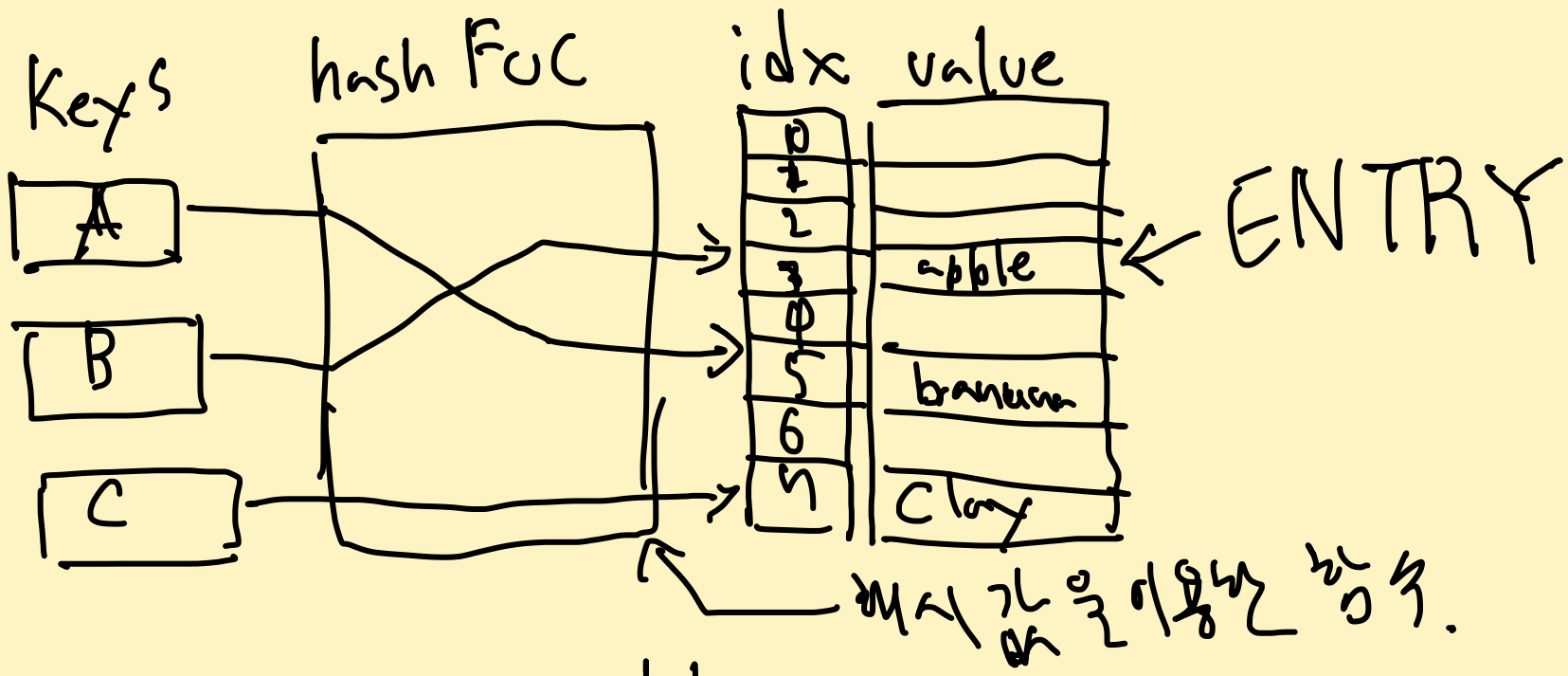
해시 테이블

해시 함수를 사용하여 키를 해시값으로 매핑하고
이 해시값을 색인(index) 삼아 데이터의 값(value)을
키와 함께 저장 하는 자료구조

이때, 데이터가 저장되는 곳을 버킷 or 슬롯이라함



Direct-address table

키의 전체개수와, 동일한 크기의 버킷을 가진 테이블
해시 충돌 발생X 전체키중 사용되는 키의
개수가 적을경우 메모리 효율성이 크게 떨어짐

해시테이블은 여러개의 같은 버킷을 가르킬 수 있음

해시충돌

해시 함수는 해시값의 개수보다 대개 많은 키값을 해시값으로 변환.

해시 함수가 서로 다른 두개의 키에 대해 동일한 해시값을 냄

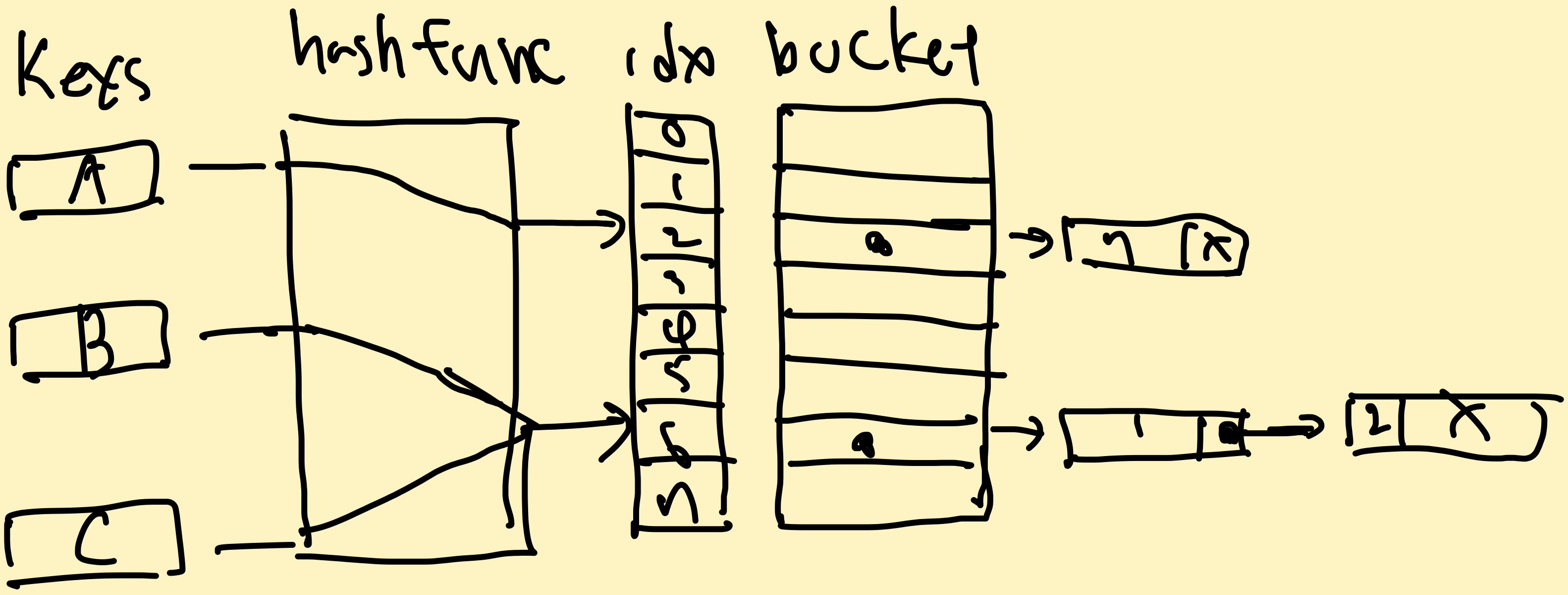
↳ 해시충돌

체인법 (chaining)

한 버킷당 들어가는 엔트리의 수에 제한을 두지않음

모든 자료를 할당, 해당 버킷에 데이터가 존재하면

체인처럼 노드를 추가 (Linked List) ⇒ 체인



시간 복잡도는

탐색, 삽입, 삭제는 평균적으로 $O(1)$

but 충돌이 최악의 경우 $O(N)$

개방 번지화 (open Addressing)
chaining 과 달리 버킷에 들어가는 엔트리는 1개
해시 함수로 얻은 주소가 아닌, 다른 주소에 데이터
저장이 가능 ⇒ open Addressing

빈 공간을 찾는 방법

1) 선형탐사
데이터 조회시 고정값을 옮겨 다음 해시값에 해당하는
버킷에 액세스 (삽입, 삭제, 탐색), 만약 공간이 없을시
고정폭으로 옮겨 액세스

2) 제곱탐사
데이터 조회시, $1^2 \rightarrow 2^2 \rightarrow 3^2 \rightarrow 4^2$ 순으로 해시값에
1 4 9 16
액세스
데이터가 한곳에 몰려 저장되는 현상
⇑
3) 이중해싱
탐사할 해시값의 규칙성을 없애 clustring
2개의 해시 함수를 사용,
1개는 최초 해시값 액세스시, 1개는 충돌발생시
탐사 이동폭을 얻기위해 사용