Hash

Hash vs Hash Table vs Hash function

1. 해시 테이블이란

2. 해시 테이블의 작동 원리

1. 해시 테이블이란 = 사전

Key: Value로 데이터를 저장

사전에서 단어(Key)를 찾아서
설명(Value)을 읽는다.

배열 VS 해시 테이블: 피자의 가격을 알고 싶을 때

```
menu = [
  { name: "coffee", price: 10 },
  { name: "burger", price: 15 },
  { name: "tea", price: 5 },
  { name: "pizza", price: 10 },
  { name: "juice", price: 5 },
];
```

선형 탐색 -> O(n)

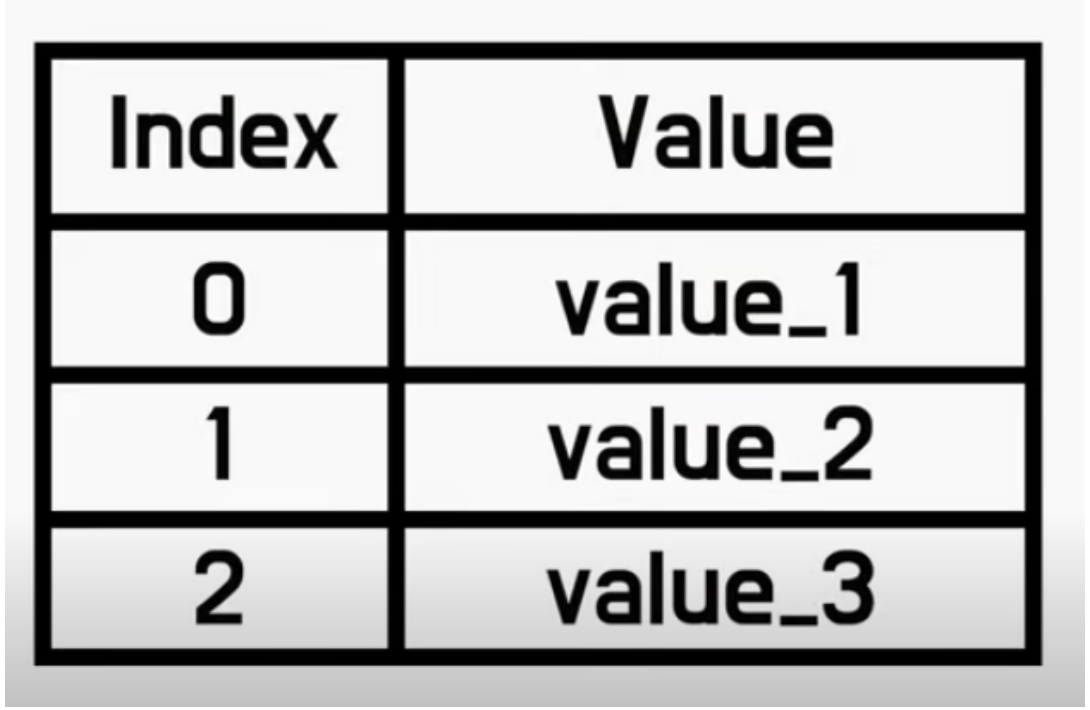
```
menu = {
  coffee: 10,
  burger: 15,
  tea: 5,
  pizza: 10,
  juice: 5,
};
```

O(1)

2. 해시 테이블 작동 원리

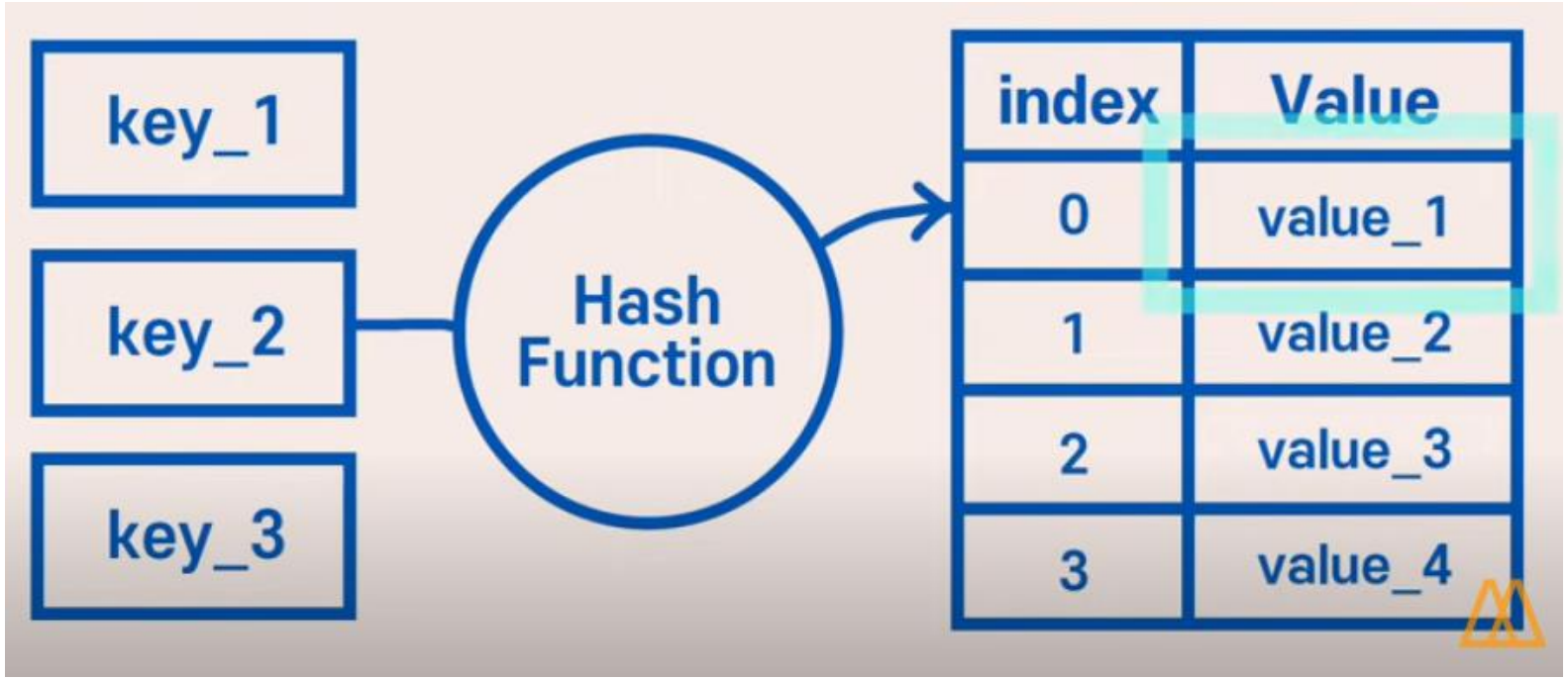
해시 테이블 내부 구조 = 배열

왜 빠른지?



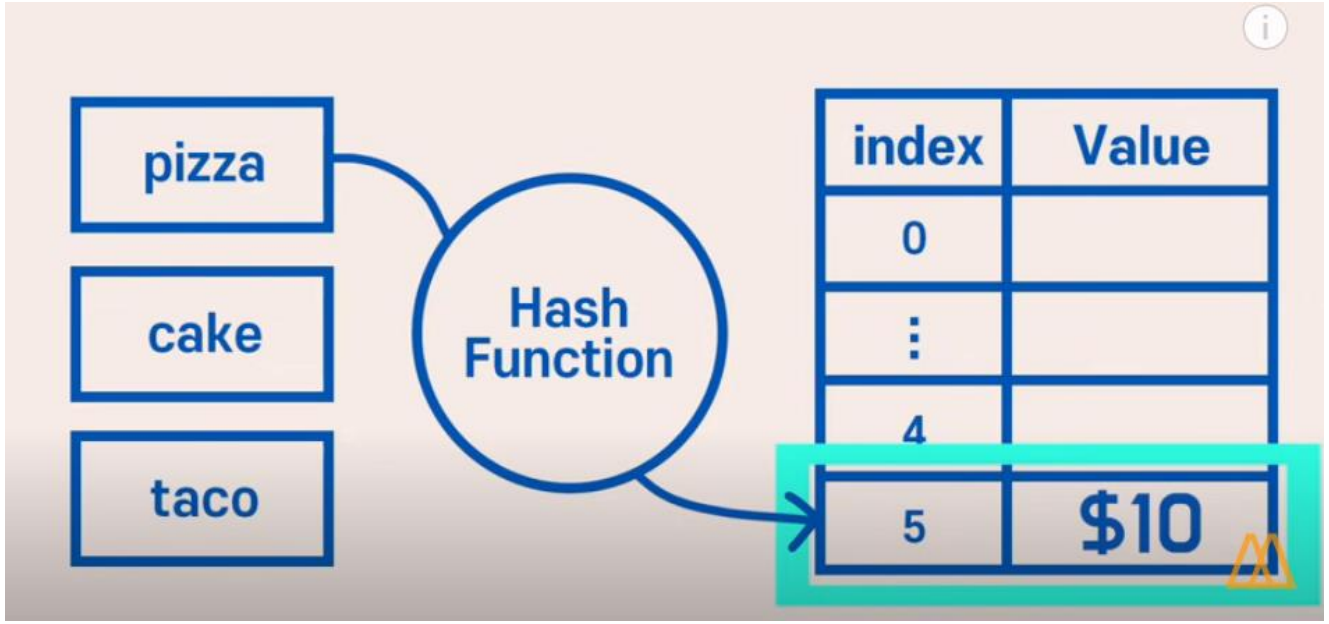
Index	Value
0	value_1
1	value_2
2	value_3

왜 빠른지? = **해시 함수** 덕분!



1. 해시 함수가 Key를 숫자로 만든다
2. 숫자는 index가 되고 해당 index를 가진 배열에 Value를 저장한다.

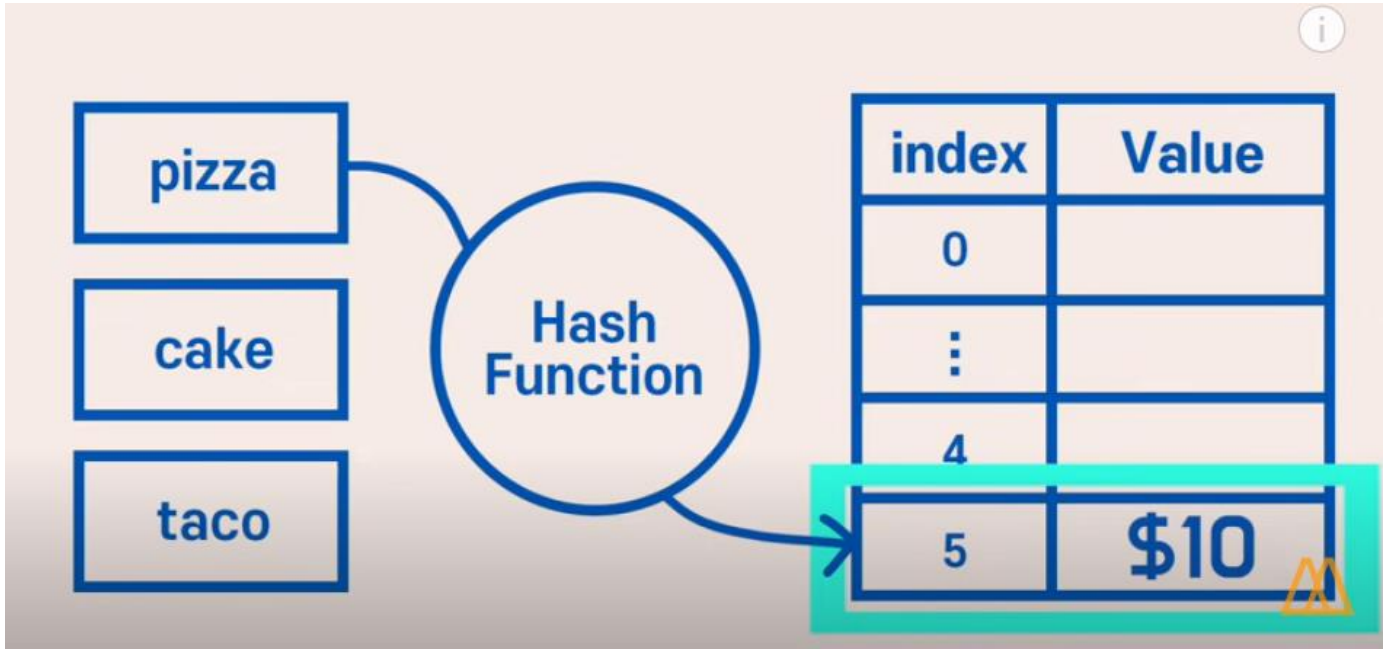
예시: 글자 수를 인덱스로 지정하는 해시 함수



저장:
1. Pizza는 5글자이다.
2. 해시 함수를 통해 index는 5가 된다.
3. 5번 인덱스 배열에 Value를 저장한다.

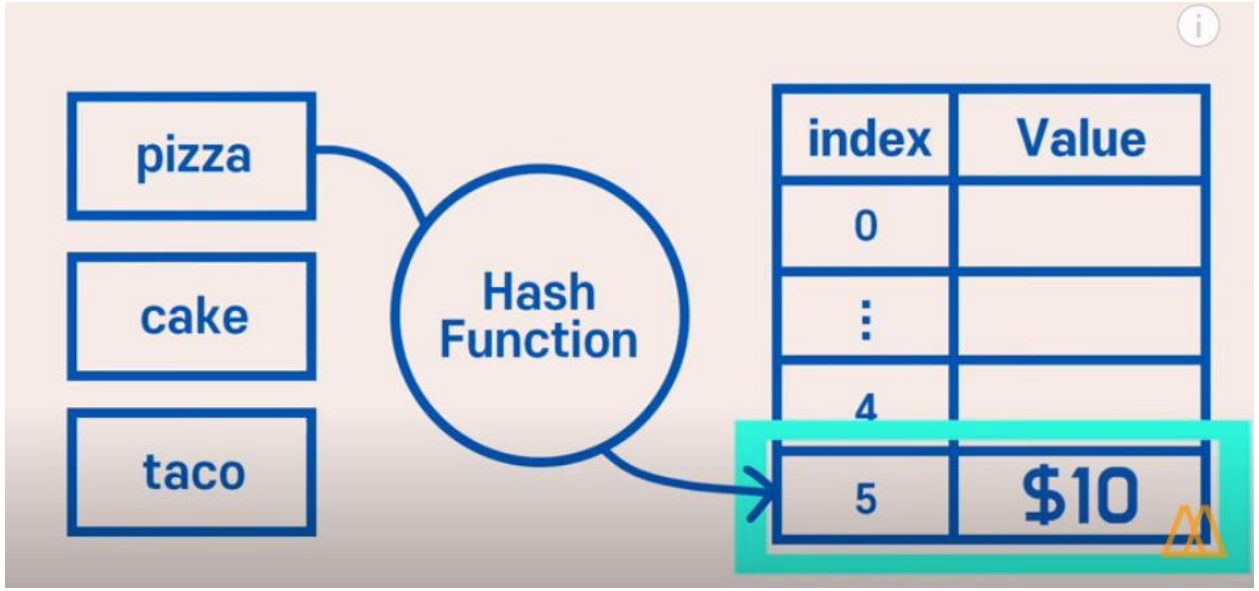
탐색:
1. 해시 함수에 Pizza를 넣는다.
2. 해시 함수는 글자수인 5를 리턴한다.
3. 5번 인덱스에서 Value를 얻는다.

예시: 글자 수를 인덱스로 지정하는 해시 함수



해시: **인덱스**
해시 함수: **Key를 Hash로** 만들어주는 함수
해시 테이블: Hash를 주소 삼아서 **Key와 Value를 저장하는 자료구조**

문제: 해시 충돌 [언제나 O(1)를 보장하지 않는다]



1. Cake는 글자수가 4이므로 4번 인덱스에 저장됨
2. Taco도 글자수가 4이다.
3. 해시 충돌!

해결:
1. 한 배열 안에 또 다른 배열을 넣는다.

문제:
배열 안에서 선형 탐색을 해야한다.
O(n)

정리

해시: **인덱스**
해시 함수: **Key를 Hash로** 만들어주는 함수
해시 테이블: Hash를 주소 삼아서 **Key와 Value를 저장하는 자료구조**

장점: key-value 1:1 매핑으로 **검색, 삽입, 삭제가 평균적 O(1)**

단점: 해시 충돌 해결 필요

순서 or 관계가 있는 배열에는 맞지 않음

공간 효율성 떨어짐(데이터 저장전에 공간을 미리 만들어야 함)

해시 함수 의존도 높음(해시 함수가 복잡하면 오래걸림)