해시 테이블 (Hash Table)

practice 자료구조 · 2024. 9. 26. 13:44

해시 테이블(Hash Table)은 데이터를 **키(key)**와 **값(value)**의 쌍으로 저장하는 자료 구조로, 매우 빠른 검색, 삽입, 삭제를 가능하게 하는 특징이 있습니다. 해시 테이블의 핵심 개념은 **해시 함수(hash function)**입니다. 이 함수는 키를 입력받아 고정된 크기의 해시 값을 반환하며, 이 해시 값을 이용해 값을 저장할 위치(메모리 인덱스)를 결정합니다.

해시 테이블의 주요 특징
1. **빠른 접근 속도**: 해시 테이블은 평균적으로 O(1)의 시간 복잡도로 데이터를 검색할 수 있습니다. 이는 키를 이용해 바로 데이터의 저장 위치를 찾기 때문입니다.
2. **해시 충돌**: 두 개 이상의 키가 동일한 해시 값을 가질 경우 **충돌(collisions)**이 발생합니다. 이 문제를 해결하는 방법에는 **체이닝(chaining)**, **오픈 어드레싱(open addressing)** 등의 방식이 있습니다.
   - **체이닝**: 동일한 해시 값이 여러 개일 경우 연결 리스트나 다른 자료 구조를 사용해 충돌된 요소들을 관리하는 방식입니다.
   - **오픈 어드레싱**: 충돌이 발생하면 다른 빈 슬롯을 찾아 데이터를 저장하는 방식입니다.

해시 테이블의 장단점
- **장점**:
  - 데이터의 빠른 검색과 삽입이 가능.
  - 효율적인 메모리 사용 가능.

- **단점**:
  - 해시 충돌이 많아질 경우 성능 저하 가능.
  - 해시 함수 선택이 성능에 큰 영향을 미침.
  - 데이터의 순서를 유지하지 않음.

이 개념은 **딕셔너리(dictionary)**와 같은 파이썬의 내장 자료 구조에도 사용됩니다. 실제로 파이썬의 딕셔너리는 해시 테이블을 기반으로 구현되어 있어, 매우 빠른 조회 성능을 자랑합니다.

---

detail

1.해시 함수란?

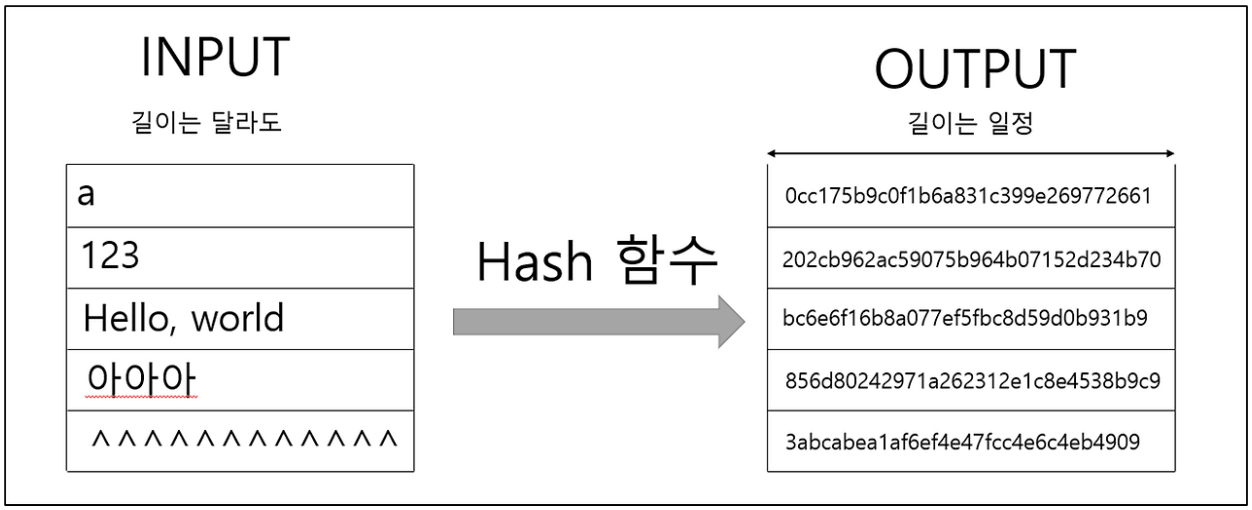
2.해시 테이블이란?
해시 충돌은 왜 생기고, 어떻게 해결하는가?

•임의의 입력값을 받아, 고정된 길이의 해시값을 출력하는 함수

•이때 해시값은 임의의 숫자 혹은 문자열

hash 함수 :

input : 아무거나 (숫자, 문자, 아무거나)
output: 숫자만 또는 문자 또는 섞여서 여러가지 해쉬값
hash함수: hash값으로 바꿔주는 함수 (숫자 , 문자로 바꿔주는 함수)
한방향: input에서 output은 가능, output에서 input은 불가능



•다양한 해시 함수가 존재 (MD5, SHA, 등)

•해시 값을 통해 입력값을 유추하는 것은 (거의) 불가능

•이러한 해시 함수를 통해 효과적으로 데이터를 저장하고 불러오는 것이 가능

•Key와 Value 쌍이 있다고 할때, 주어진 Key에 대한 Value를 효율적으로 찾고 싶음

•예를 들자면, 학생의 이름이 Key, 시험 점수를 Value라고 할때, 주어진 학생의 점수를 빠르게 찾고 싶음

•가장 단순한 방법: 모든 Key-Value 쌍을 저장

•[(김지훈, 60), (김동현, 50), (김현우, 70), …]

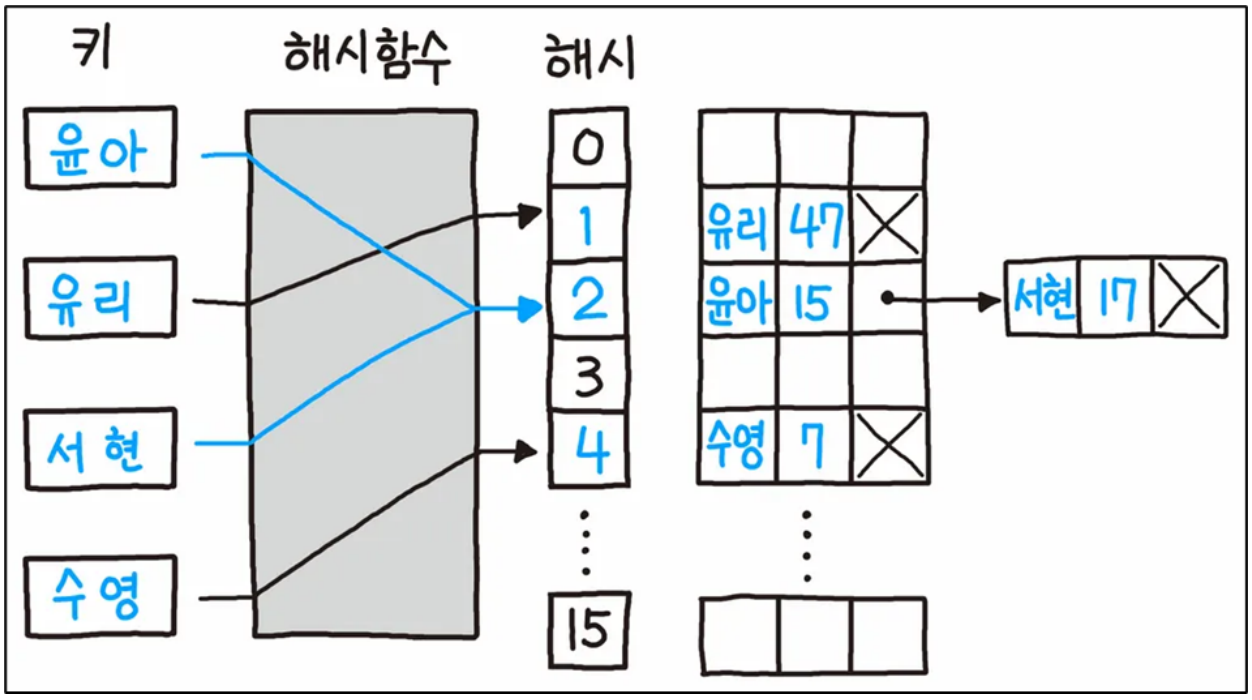
•하지만 이 방법은 O(n)의 시간 복잡도를 가짐 (매우 비효율적)

•보다 나은 방법: 해시 테이블

•List (혹은 Array)에서, 주어진 Index의 값을 찾는 것은 O(1)의 시간 복잡도를 가짐 (매우 효율적)

•또한, 해시 함수를 쓰면 임의의 Key를 숫자로 변환하는 것이 가능

주어진 Key에 대한 Value를 해시 함수로 변환된 숫자 Index에 저장하고 사용하자

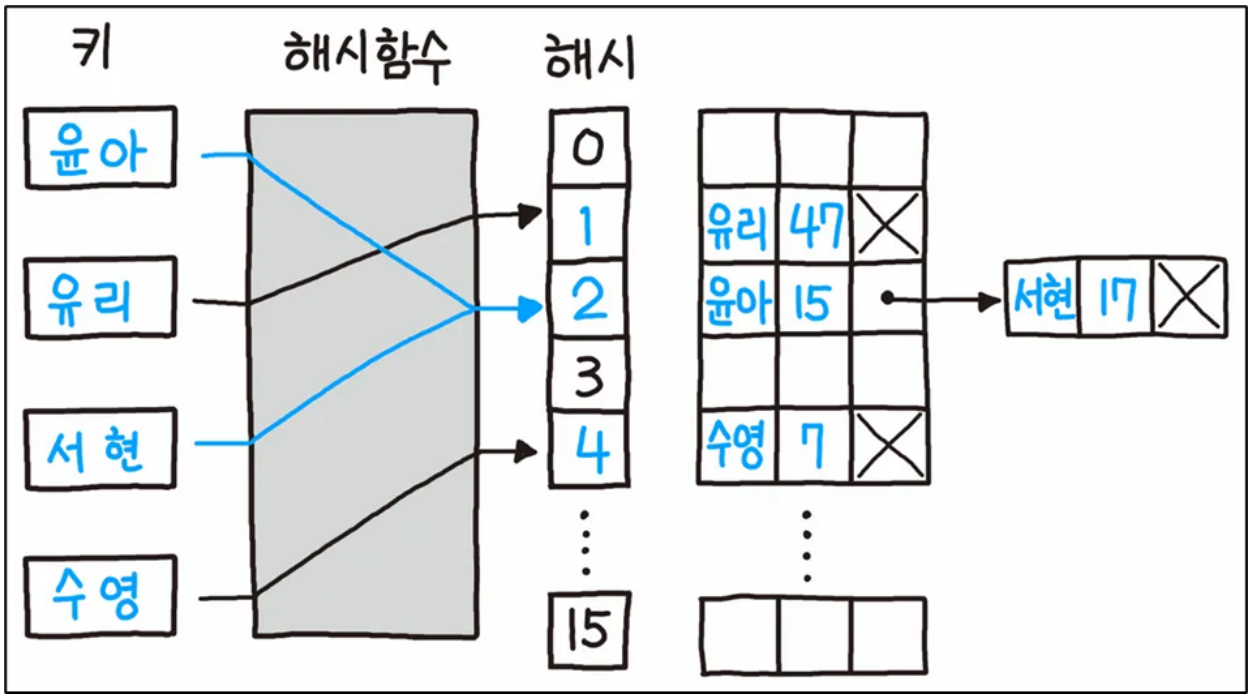


문제 1 – 해시 충돌

•이때 여러 개의 Key가 같은 숫자 (Index)를 가질 수 있음 (충돌)

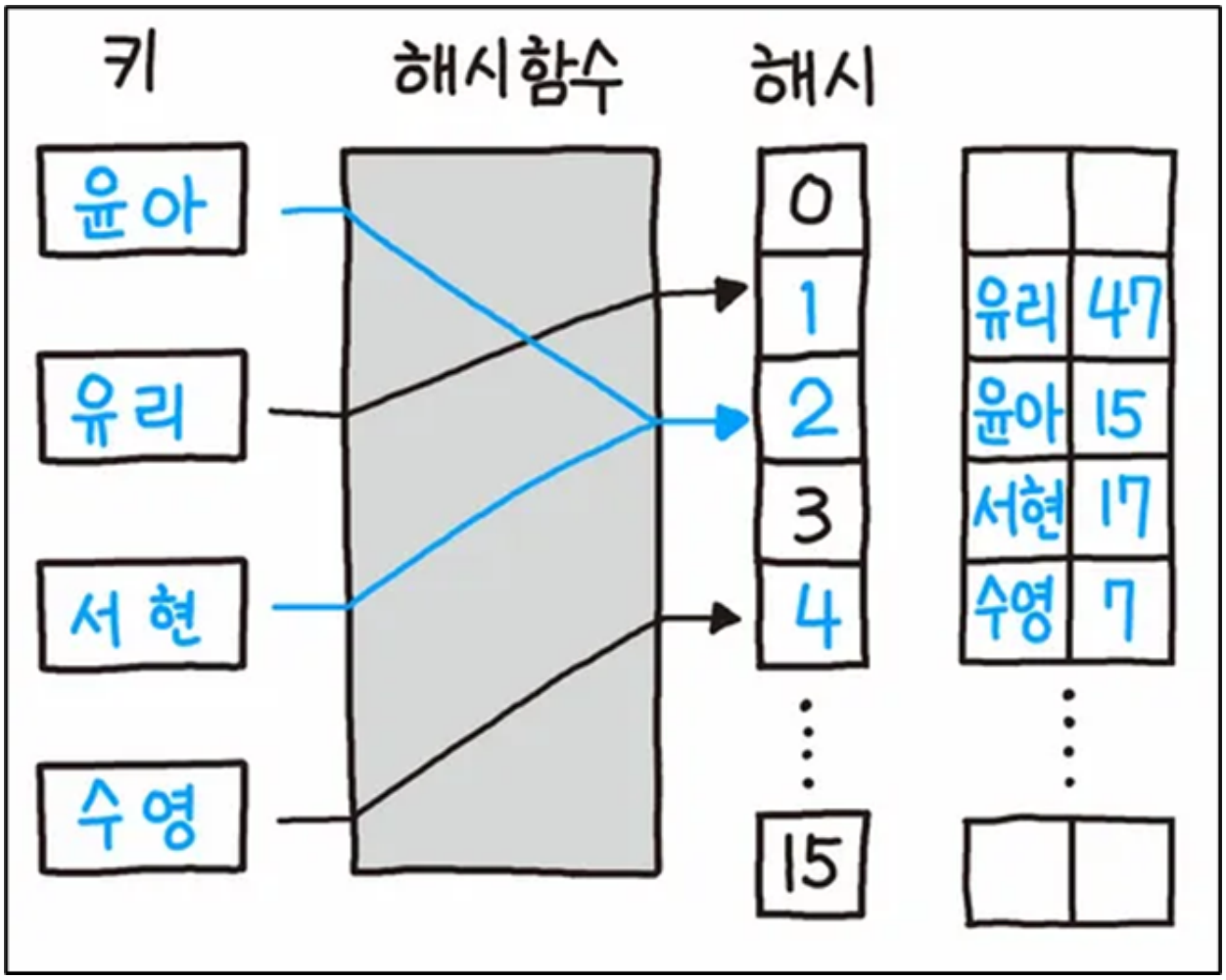
해결방법

첫번째 방법: 체이닝 (Chaining)
•값이 충돌하면, 단순히 Linked List로 새 Value를 저장하는 것



•두번째 방법: Open Addressing

•충돌이 발생할 경우, 주변에 있는 빈 슬롯을 찾아 Value를 저장하는 것



문제 1 – 모범 정답

•해시 테이블은 키를 해시 함수로 변환하여 값을 저장하는 방법입니다.

•해결 방법에는 주로 체이닝(Chaining)과 개방 주소법(Open Addressing)이 있습니다.

•체이닝은 충돌이 발생한 키를 링크드 리스트로 저장하는 방법이며,
Open Addressing
은 빈 슬롯을 찾아서 저장하는 방법입니다

.

문제 1 – 추가 문제

•Open Addressing 방식은 테이블에 이미 많은 Value들이 저장되어 있을 경우, 처리 시간이 크게 늘어납니다. 왜 이런 현상이 발생할까요?

답변

•Open Addressing에서, 해시 충돌이 발생하면, 주변에 있는 빈 슬롯을 찾아 Value를 저장하게 됩니다.

•하지만 만약 테이블에 이미 많은 값들이 저장되어 있을 경우, 빈 슬롯을 찾는 것이 매우 오래 걸리게 됩니다.

•따라서 해시 테이블에 빈 공간이 별로 없는 경우, 큰 테이블로 옮기는 것이 (일반적으로) 필요합니다.

연습

① Hash table (key, value)

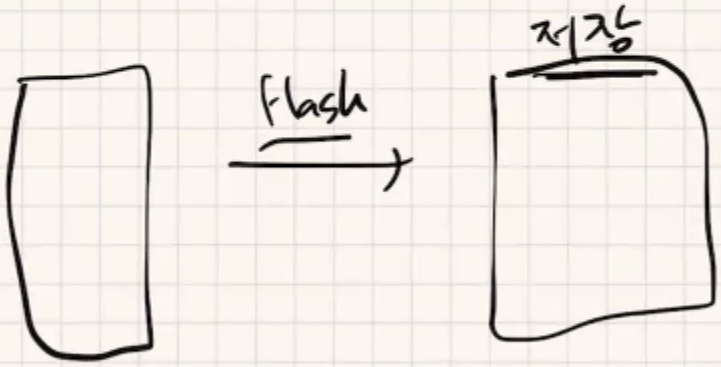
일반적으로 딕셔너리 구현에 사용됨.
(key-value)

- 기본적인 방법

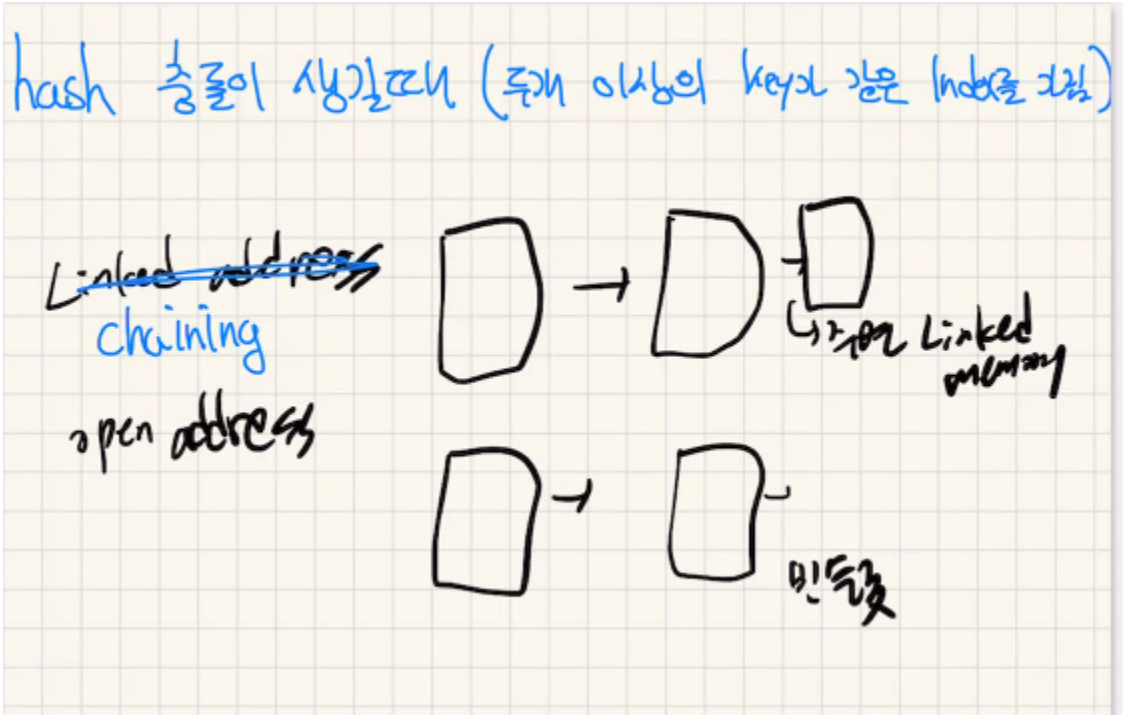
[(key₁, value₁), (key₂, value₂) ····]

→ 주어진 key를 전부 찾아봐야 함 O(n)

data → hash → 저장



- 주어진 key를 hash 함수 사용해서 index로 변환
- 해당하는 value를 index 위치에 저장

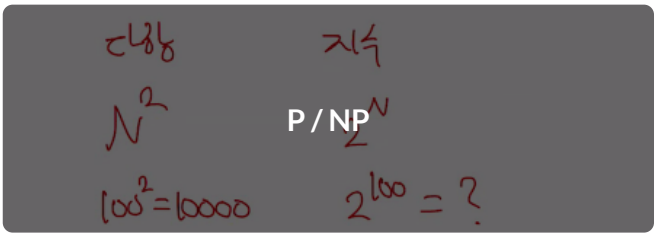


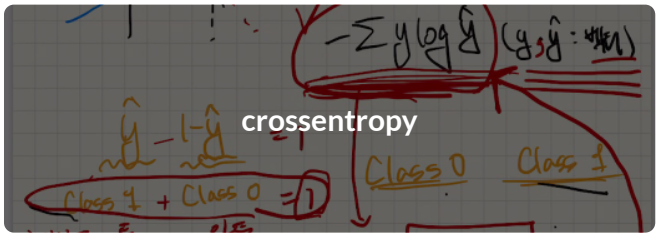
'practice_자료구조' 카테고리의 다른 글

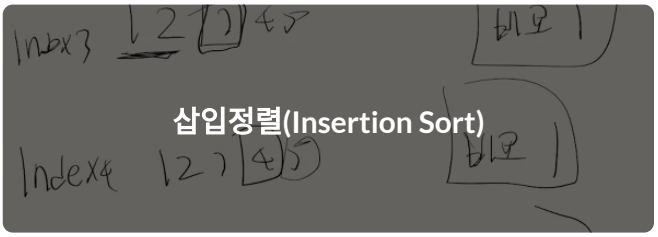
P / NP (0)	2024.09.30
crossentropy (1)	2024.09.20
삽입정렬(Insertion Sort) (0)	2024.09.15

관련글

관련글 더보기







자연어(NLP)

네이쳐2024 님의 블로그입니다.

구독하기 +

댓글 0

이름

비밀번호

내용을 입력하세요.

등록