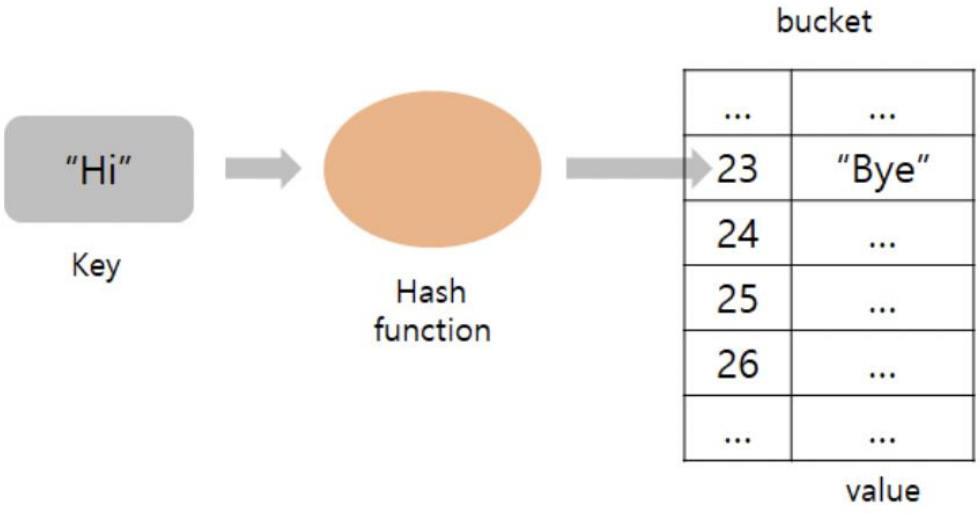
해시란



먼저 Hi 라는 key 값과 Bye 라는 value 값 한 쌍을 만들어진 해시에 삽입한다고 하면 이 key 값이 '해시함수'에 들어가 고정된 bucket 배열 크기 내의 수를 리턴해줍니다.

장점

해시테이블은 key-value 가 1:1 로 매핑되어 있기 때문에 삽입, 삭제, 검색의 과정에서 모두 평균적으로 O(1)의 시간복잡도를 가지고 있다.

단점

해시 충돌이 발생(개방 주소법, 체이닝 과 같은 기법으로 해결해 줘야 한다.)

순서/관계가 있는 배열에는 어울리지 않는다.

공간 효율성이 떨어진다. 데이터가 저장되기 전에 저장공간을 미리 만들어놔야한다.

공간을 만들었지만 공간에 채워지지 않는 경우가 발생한다.

hash function 의 의존도가 높다. 해시함수가 복잡하다면 hash 를 만들어 내는데 오래 걸릴 것이다.

해시가 작동되는 순서

해당 원소(key, value)의 해시 함수를 이용하여 해시값을 얻는다.

해시값을 주소로 하는 위치에 원소를 저장한다.

저장 후 검색 시에도 원소의 해시값으로 계산하여 해당 위치로 이동한다.

해시함수란?

임의의 길이의 데이터를 수학적 연산을 통해 고정된 길이의 데이터로 매핑하는 함수이다.

위같은 과정에서 어떤형태의 key 값을 bucket 크기내의 수를 리턴해주는 함수

해시 함수의 특징

어떤 입력 값에도 항상 고정된 길이의 해시값을 출력한다.
눈사태 효과 : 입력 값의 아주 일부만 변경되어도 전혀 다른 결과 값을 출력한다.
출력된 결과 값을 토대로 입력값을 유추할 수 없다.

만약 hi(아스키코드 h = 104, i = 105)를 입력했는데 해시함수가 각 글자의 아스키 코드를 곱하고 bucket 의 크기로 나누는게 전부라면 104*105%bucket_size 이러한 해시함수가 될 것이고 만약 ih 를 넣었다면 같은 리턴할 것입니다. 그럼 한 공간에 두 데이터를 넣어야 하나?

따라서 좋은 해시함수란 위같은 중복 없이 확률적으로 bucket 에 골고루 나누어 데이터가 저장되는 것으로 볼 수 있다.

충돌이란

해시함수의 입력은 무한함에 비해 bucket 사이즈는 유한하기 때문에 Key 값이 다르더라도 해시함수가 같은 값을 반환 할 가능성이 있다.

충돌을 해결하는 방법 및 충돌이 가장 적게 일어나는 해시함수를 만드는 방법도 있습니다.

인터뷰 질문으로

1. 해시 맵과 해시 테이블 차이
2. 해시 충돌 해결 방법

여기까지 발표

해시 맵 해시 테이블 차이

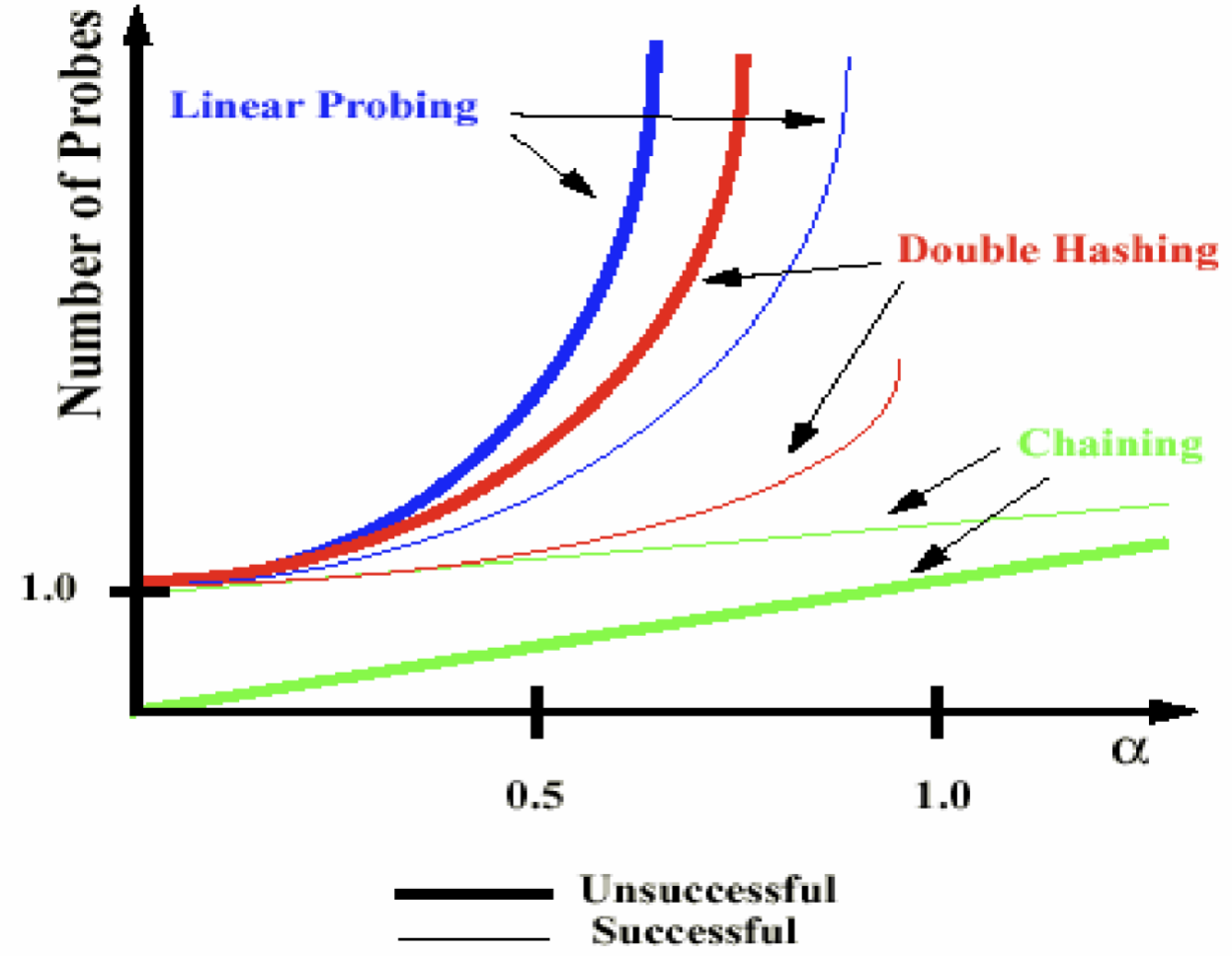
해시 테이블 = 자원의 동기화 고려
이 말은 쓰레드라는 작업 처리 단위가 있는데 여러 개의 쓰레드가 '동시에' 해시테이블에 접근할 때(멀티 쓰레드 환경)에는 접근이 불가능 하다. 이는 해시 맵보다 느린 성능을 가져오는데 이유는 한 쓰레드가 해시테이블을 사용중일 때에는 데이터 접근을 막기 위해 잠금(lock)을 걸어야 하는데 이 잠금을 거는 시간으로 인해 느려집니다. 하지만 데이터의 무결성을 보장합니다.

해시 맵 = 자원의 동기화 고려할 필요 없을 때에 사용
이 말은 멀티 쓰레드 환경이 아니라면 동시에 접근하지 않으므로 안전하고 빠르게 사용할 수 있다.

이들의 장점만으로 자바에서 concurrenthashmap 을 통해 한 쓰레드가 데이터에 접근할때 그 데이터만 wrapping(lock 와 비슷하지만 더 작은 역할)하여 다른 쓰레드는 다른 데이터에도 동시에 접근이 가능하여 데이터의 무결성 또한 보장 된다.

해시 충돌 해결

Load factor (α) 에 따른 시간 복잡도 추이
Load factor 란 저장된 데이터 수를 만들어진 bucket 으로 나눈 값



https://velog.io/@jaeyunn_15/DataStructure-해시-충돌과-해결-방법

해시 사용 용도

무결성 검사
DB 비밀번호 저장
해시 테이블 사용

언어별 해시 충돌 해결 방법

언어	방식
C++	개별 체이닝
Java	개별 체이닝
Go	개별 체이닝
Ruby	오픈 어드레싱
Python	오픈 어드레싱

참고자료

해시 테이블 해시 맵 차이
https://doorisopen.github.io/developers-library/Java/2020-07-08-java-hashmap-and-thread-safe

프로세스 스레드
https://gmlwjd9405.github.io/2018/09/14/process-vs-thread.html