해시 테이블

2021년 2월 22일 월요일 오후 4:29

262. 해시 테이블

- 다시 보는 해시 테이블의 시간 복잡도

자료구조	평균(average)			최악(worst)		
	검색	삽입	삭제	검색	삽입	삭제
배열	O(n)	O(n)	O(n)	O(n)	O(n)	O(n)
스택	O(n)	O(1)	O(1)	O(n)	O(1)	O(1)
큐	O(n)	O(1)	O(1)	O(n)	O(1)	O(1)
연결 리스트	O(n)	O(1)	O(1)	O(n)	O(1)	O(1)
해시 테이블	O(1)	O(1)	O(1)	O(n)	O(n)	O(n)

  - 해시 테이블은 평균적으로 검색/삽입/삭제가 O(1)
  - 이게 어떻게 가능한 걸까?
  - **O(1)이 가능하려면 어떤 메모리 주소에 어떤 데이터가 저장되어 있는지 한 방에 알 수 있어야 함**

- 무작위로 뽑은 수 10개를 저장해보자

```
703 793 724 441 219 1 81 546 777 747
```

  - 크기가 10인 배열에 차례대로 저장하면 O(N)으로 검색 가능

```
int has_number(int n)
{
    int i = 0;
    for (i = 0; i < LENGTH; ++i) {
        if (s_numbers[i] == n) {
            return TRUE;
        }
    }

    return FALSE;
}
```

s_numbers	703	793	724	441	219	1	81	546	777	747

- 방법 1: 입력값의 최댓값을 배열의 크기로
  - 가장 단순&무식한 방법
    1. 입력값의 최댓값만큼 긴 배열을 만듦
    2. 배열[입력값]에 값이 있으면 1, 없으면 0을 저장
       - 즉, 이 경우에는 입력값이 배열의 색인

```
s_numbers[1] = 1;  /* 입력값: 1 */
s_numbers[81] = 1; /* 입력값: 81 */
```

```
703 793 724 441 219 1 81 546 777 747
```

0	1	...	1	...	0	...	1	...	1
[0]	[1]		[81]		[200]		[546]		[793]

  - 방법 1의 문제
    - 입력값이 엄청 크다면?
      - 예: 무작위로 뽑은 수 중 최댓값이 291948

	...		...		...		...	...	

291949개

        - 배열의 크기를 무한히 늘릴 수도 없음
            - 2^32이면 이미 4기가..
            - 비효율의 극치

263. 해시 테이블의 크기는 2배 이상인 소수로

- 우리가 알고 있는 사실들
    - 입력값은 총 10개 뿐
    - 입력값이 꽤 클 수 있음

```
703 793 724 441 219 1 81 546 777 747
```

예1

```
874 35741059748 6984 2 75 96 81575 178 63214 563
```

예2

- 우리가 만들어야 하는 것
    - 다음과 같은 입력과 출력을 가지는 함수
        - 입력: 10개의 중복이 없는 수
        - 출력: 범위가 [0, 9]인 배열의 색인
    - 즉, 자료(입력) -> 색인(출력)으로 바꾸는 함수
    - 단, 출력값을 찾을 때, 반복문 없이 O(1)이어야 함
        - **가장 간단한 방법: 입력값 % 10**
- 방법 2: 입력값 % 10으로 색인 만들기

- 방법 2의 문제
    - 동일한 색인 위치에 들어가는 애들이 생긴다

    - 배열의 크기를 충분히 키우면 되지 않을까? 한 두 배 정도?
- 방법 3: 배열의 크기를 두배로
    - 겹치는 색인이 아까보다는 덜 생기나 여전히 존재

- 방법 3의 보완: 가장 좋은 방법 -> 소수도 사용
  - 흔히 다음과 같이 하는게 좋다고 말함
    1. 배열의 크기는 최소 2배
    2. 크기에는 소수(prime number)를 사용
  - 소수로 나눠야 동일한 색인이 안 나올 가능성이 제일 높음
    - 소수가 아닌 짝수 24로 나누면
      - 24의 약수가 많아서 색인 중복이 많이 생긴다
        - 24의 모든 약수는 자신의 배수가 곱 중복으로 나타난다
        - 예: 2 3 24 27 48 51 72 75

          -> 2 3 0 3 0 3 0 3
    - 소수인 23으로 나누면
      - 23은 약수가 1밖에 없으므로 색인 중복이 덜 생긴다
    - 참고로 자연에서도 마찬가지 이유로 소수가 많이 발견됨
      - 매미가 천적을 피해서 소수년도(13년 17년 등등)마다 지상으로 나오는 이유
  - 그래도 중복이 없을 수는 없음

264. 중복 색인 문제의 해결

- 중복 색인 문제를 해결하는 방법
  - 나머지 연산으로 구한 색인 위치 이후에 빈 공간을 찾아 저장
  - 즉, 색인을 1씩 증가해가며 빈 색인 위치를 찾아 거기에 저장
  - 따라서, 더 이상 불 값(1또는 0)이 아니라 실제 값을 저장해야 함

  - 따라서, 배열에서 비어있는 위치를 알 수 있어야 함
    - 방법 1: 불(bool) 배열을 만들어서 사용 여부를 나타냄

        - 방법 2: 어떤 특정한 값을 저장해서 비어있다는 사실을 표시
            - 보통 유효하지 않는 값(절대 들어올 수 없는 값. 예: INT_MIN)을 사용
        - 우리는 방법 2를 사용

- 입력값을 배열에 넣어보자!
    - 배열에서 비어있는 위치를 알아야 함
        - 중복된 색인값이 있는데 배열이 거의 찬 경우 다 돌아야 해서 O(n)이 될 수도 있다
    - 빈 색인 위치에는 INT_MIN이 들어있음
        - 편의상 아래 그림에서는 공백으로 표시

- 배열을 사용한 해시 테이블 예

    - 중간에 삭제되어서 INT_MIN이 된 것이 없다는 것을 가정한 함수
        - 중간에 값이 삭제되어서 INT_MIN이 있으면 반복문이 돌지 않음

- 이게 바로 해시 테이블
  - 방금 본 게 바로 초간단 해시 테이블
  - 색인 중복이 없으면, 읽는 것도 쓰는 것도 다 O(1)
  - 색인 중복이 있으면, 최악의 경우 O(N)
- 또 다른 방법: 연결 리스트를 사용
  - 배열 안의 각 요소에 연결 리스트를 저장
  - 여전히 색인이 겹치는 경우는 있음
    - 그래도 O(N)이 되는 경우가 앞의 방법보다 적음
      - 중복 될 때 한바퀴 다 돌 경우가 적다
      - 모든 숫자가 같은 색인에 있으면 O(N)
    - O표기상 여전히 O(N)
    - 동적 할당을 하므로 속도상의 문제가 있음
      - 좀더 최적화를 하자면 배열을 사용하는 방법이 있음
        - 겹칠수 있는 최대 요소는 5개로 정하고 배열 5개짜리를 잡아 둠
          배열 5개가 넘어가면 assert

265. 해시란 무엇인가?

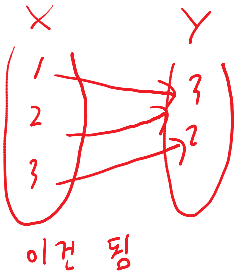
- 왜 해시 테이블이라고 할까?
  - 해시 값을 사용해서
- 그럼, 해시 값은 뭘까?
- 해시 값
  - 어떤 데이터를 **해시 함수**에서 넣어서 나온 결과

  - 그럼 해시 함수는 뭘까?
- 해시 함수
  - 임의의 크기를 가진 데이터를 '고정 크기'의 값에 대응하게 하는 함수

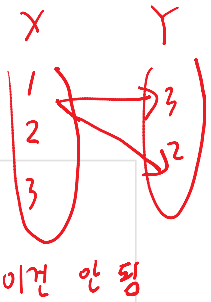
  - 입력값의 범위와 출력값의 범위가 확연히 다른 것
    - 입력값의 범위는 제한이 없음
      - 작을 수도 있고 클 수도 있고
        비트 수로 5000비트여도 상관 없고 64비트여도 상관 없음
    - 해시함수를 거친 출력값은 고정된 크기를 가짐
      - 보통은 32비트 정수형을 쓰거나, 요즘은 64비트 정수형도 씀
        보안쪽 가면 더 큰 수인 256비트 정수형도 쓴다
  - 수학에서 배운 함수와 비슷한 개념
    - 함수는 입력값이 같으면 출력값은 언제나 같다

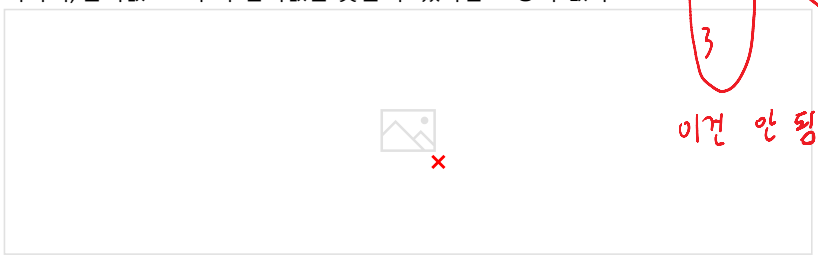
- 입력값이 달라도 출력값이 같을 수 있다

- 해시 충돌 (hash collsion)
  - 방금 본 것처럼 입력값이 다른데 출력값이 같은 경우를 말함
  - 물론, 이런 일은 없을수록 좋음



  - 따라서, 출력값으로부터 입력값을 찾을 수 있다는 보장이 없다





    - 보안 쪽에서는 출력 값으로부터 입력 값을 찾기 어려울 수록 좋음
      예: 해시 함수를 거친 로그인 정보가 털렸을 때 원래 값이 뭔지 알아내기 어려우면 좋다
- 그럼 아까전에 본 % 23이 해시 함수일까?

  - 개념상으로 해시함수긴 한데 반드시 그렇다고 할 수가 없다
- 좀 더 자세히 설명하는 해시 값
  - 어떤 데이터를 대표하는 값 하나를 내놓으라고 함
  - 이렇게 해서 받은 값이 해시 값
    - 마치 대한민국의 주민등록 번호 같은 존재
    - '사람'은 수가 아니지만 주민등록 번호가 그 사람을 대표하는 수
    - 그래서 영어는 ID(identity) 라고 함
    - 해시값은 어떤 데이터를 대표하는 정해진 형식의 숫자
  - 앞의 예에선 입력값 그 자체를 해시 값이라 볼 수도 있음
    - 앞의 예의 입력값은 크기가 고정된(32비트 숫자 int형) 숫자이므로
  - 그 해시 값을 정해진 배열 안에 꾸겨 넣으려고 % 연산을 한 것뿐

- 색인으로 쉽게 변환할 수 없는 데이터는 뭐가 있을까?
  - 문자열 아니면 바이트 배열
    - 크기가 정해지지 않는 데이터

266. 해시 테이블에 문자열 저장하기

- 해시 테이블에 문자열 저장하기
  - 우리가 할 일: 문자열을 배열 색인으로 변환
  - 아까 사용한 연산자 %은 문자열에 사용 불가
  - 문자열을 정수형으로 변환하는 해시 함수가 있어야 함!

  - 문자열의 각 요소는 어차피 아스키 코드 = 정수 값
  - 정수 값들이 줄줄이 서 있는걸 잘 합치면 될 것 같음

- 잘 쓰이지 않는 해시 함수 예

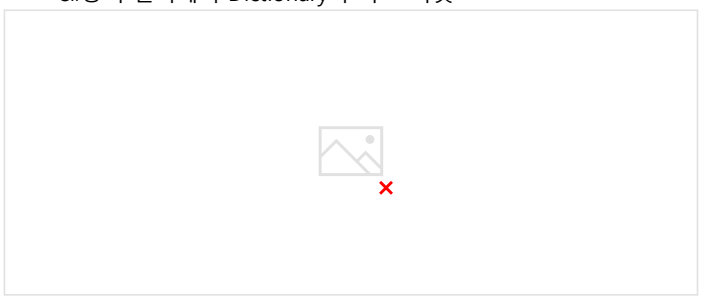
  - 문자열의 아스키 코드를 싸그리 더해서 반환
  - "Teemo"를 매개변수로 넣었을 때
    - 84 + 101 + 101 + 109 + 111 = 506
  - "Hana"의 해시 값
    - 376
  - 해시 충돌이 빈번해 잘 사용하지 않음
- 해시 값이 나왔으니 이제 진짜 저장해보자
  - 해시 값을 찾았으니 색인을 구할 수 있음
    - 색인 구하기는 나머지 연산(%)만 하면 된다
  - 색인을 구하면 테이블에 저장이 가능
    - 해당 색인 위치에 다른 값이 들어있다면 1씩 증가하면서 빈 곳을 찾으면 됨

- 하지만 O(문자열 길이)
  - O(1)이 아님
  - 해시 함수가 문자열 길이만큼 반복문을 도니 O(문자열 길이)
- 정말 O(1)으로 만들고 싶다면?
  - 해시 함수를 한 번만 호출하고 그 결과를 기억해둠
    - int has_id = hash_function(문자열)
  - 필요할 때마다 이 변수로 해시 테이블 함수를 호출
    - add(hash_id, data);
  - 이제 hash_id로부터 저장할 위치의 색인을 O(1)으로 구할 수 있다!

- 지금까지 본 건 엄밀히 말하면 해시 세트
  - 즉, 집합 내 어떤 데이터를 중복 없이 저장한 게 전부
  - 저장할 위치를 찾으려고 해시 함수를 썼을 뿐

267. 해시 맵

- 일반적인 해시 테이블의 형태
  - 어떤 키에 대응하는 어떤 값을 같이 저장
  - 이런 형태를 보통 해시 테이블(엄밀히 말하면 해시 맵)이라 함
    - C#등의 언어에서 Dictionary가 바로 이것



  - 키(key): 데이터의 위치를 의미하는 데이터
    - 꼭 정수가 아니어도 됨. 문자열도 가능. 구조체도 가능
  - 값(value): 실제 저장하는 데이터
    - 키를 가지고 그 위치에 가서 자물쇠를 열면 데이터가 나온다는 의미
- 키와 값을 사용해서 저장하는 예

- Lulu의 점수를 알고 싶을때
  1. 해시값 계산 : 418
  2. 색인 : 418 % 7 = 5
  3. 색인 5를 찾아간다
  4. 값이 Lulu가 아니므로 다음칸 이동
  5. 찾음

268. 해시 충돌과 훌륭한 해시 함수

- 이 해시 테이블에도 충돌이 발생함
  - 한 가지도 아니고 무려 두가지
    1. 해시 충돌: 키가 다른데 같은 해시 값이 나옴
    2. 색인 충돌: 해시 값이 다른데 같은 색인이 나옴
  - 두 번째 문제는 이미 해결한 문제
    - 같은 색인이면 다음칸 이동
- 해시 충돌
  - 지금까지 사용한 해시 함수에 이 두 문자열을 넣으면 해시 충돌

- 해시 충돌 복습시간
  - 함수니까 입력값이 같으면 출력값은 언제나 같다
  - 함수니까 입력값이 달라도 출력값이 같을 수 있다
  - 따라서, 출력값으로부터 입력값을 찾을 수 있다는 보장이 없다

- 해시 충돌 문제를 해결해야 하나요?
  - 해시 충돌 문제는 사실 해결 안 해도 되긴 함
  - 그냥 색인 충돌 문제의 해법으로 같이 풀림
- 만약, 해시 충돌을 방지할 수 있다면?
  - char*를 복사해서 키로 저장할 이유가 없음
  - 해시 값인 정수형(int)만 저장해도 됨
    - 다른 문자열을 넣었을 때 언제나 다른 정수가 나온다면
  - 뭐가 좋냐고? char* 저장하려고 동적 메모리 할당 안 해도 됨
    - 키 문자열은 고정된 크기로 저장할 수 없고 동적할당해서 저장해야 함
    - 동적 메모리 할당은 느림 안 할 수 있다면 이득
  - 이게 가능한가요?
    - 훌륭한 해시 함수가 있다면 가능
- 훌륭한 해시 함수란?
  - [필수] 어떤 경우에도 고정된 크기의 값으로 변환 가능
    1. 어떤 자료형이든
    2. 데이터의 길이에 상관없이
  - 해시 충돌이 거의 없음 (이게 훌륭하냐 아니냐를 결정)
    - but, 용도에 따라 속도가 빠르고 충돌이 많은게 좋을 수도 있다
    - 충돌이 없으면 좋다는 건 지금 우리가 배우는 일반적인 데이터 구조에서 좋다는 것
  - 따라서, 충돌이 덜 나는 해시 함수가 좋음

269. 해시 충돌 방지와 성능 향상

- 정말 충돌이 안 나는 해시 함수가 있나요>
  - 아주 덜 나는 함수는 있음
  - 강사가 20년 넘는 경력동안 본 해시 충돌은 딱 2번
    - 단, 문자열이 아닌 0이 굉장히 많이 들어있는 이진 데이터에서
    - 그것도 32비트 해시였음
  - 64비트 해시에서는 아직 한 번도 못 봄
- 해시 함수 테스트
  - 어떤 사람이 해시 함수를 테스트함
  - 테스트에 사용한 키들
    1. 소문자 영어단어 216,553개
    2. 정수(1~216,553)
    3. 무작위로 뽑은 216,543개의 GUID

- 간단하고 나쁘지 않은 해시 함수: 65599

- 완전히 충돌을 없앨 수 있을까?
  - 기술적으로는 안 됨
  - 허나 특정 조건 하에서는 해시 충돌을 확실히 방지 가능
  - 그런 조건이 생각보다 많음
  - 이게 가능하면 키 배열에 문자열 대신 정수만 저장해도 됨
    - char* 동적 메모리 할당이 사라짐
    - 해시 테이블에 추가하는 함수를 호출할 때 키 대신 해시 값을 바로 사용
- 해시 충돌을 완벽히 방지할 수 있는 조건이란?
  - 고성능을 요하는 업계에서 많이 쓰는 방법
    - 예: 게임 개발
  1. 실행 중에 해시 테이블에 저장될 수 있는 데이터를 모두 알고 있음
     - 예: 미리 만들어 놓은 데이터 파일 읽기
     - 유저로부터 받는 입력이 아니라는 이야기
       - 유저로부터 입력받는 데이터는 뭐가 들어올 지 예측 불가능 함
  2. 개발 도중에 해시 충돌이 없다는 걸 확인하고 보장할 수 있음
  - **키 저장 단계를 없애도 된다**
- 충돌까지 고려한 해시 맵 예

- 충돌이 없을 때 해시 맵 예

- 다른 자료형은요?
  - 그냥거저 먹으면 됨
  - 문자열은 char*
  - 어떤 데이터도 char 배열로 표현 가능
    - 바이트 개념으로 보면 어떤 데이터든 적용
    - 즉, 가장 범용적인 데이터
  - 아래와 같은 함수 하나만 있으면 끝

  - 단 저장하려는 데이터에 걸맞는 해시 함수를 찾아 사용할 것
- 베스트 프랙티스: 언제 어떤 자료구조를 쓸까
  - 기본적으로 배열
    - 가장 간단함
    - 캐시 메모리 덕분에 O표기법에 상관없이 성능이 가장 빠른 경우가 많음
      - 메모리 캐쉬라던가 CPU최적화가 많이 되어 있음
        그래서 엄청나게 요소수가 많은 데이터가 아니라면
        배열에 있는 데이터는 O표기법과 상관없이 실제로 빠른 경우가 많음

- 빈번한 데이터 삽입 또는 삭제를 해야 한다면?
  - 연결 리스트
- 다음과 같은 경우는 해시 기반 자료 구조들
  - 데이터 양이 많은데 검색을 자주 해야 함
  - 배열에 넣기 힘든 데이터
    - 예: 연속적, 규칙적인 색인이 나올 수 없는 경우
      - 해쉬 함수로 정수화,색인화 시킴