

# 자료구조 스터디

## [학생 정보 관리 프로그램]



인하대학교 컴퓨터정보공학과

12141567 양선미

010-8975-5515

[pdsunmii@gmail.com](mailto:pdsunmii@gmail.com)

# 개요

## ■ 설계의 목적

- 이중해싱을 이용하여 학생정보 관리 프로그램을 설계하고 자료구조를 이해한다.

## ■ 요구사항

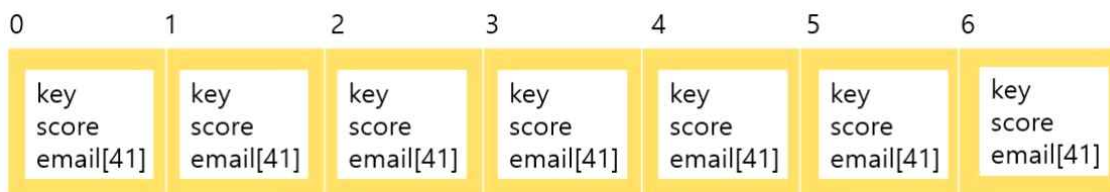
- 파일로부터 데이터 불러와서 회원정보를 해시테이블로 구축
- 표준입력으로 주어진 각 질의에 대한 답을 표준출력으로 출력하는 것
  - 학생 추가 (질의 A S E, 출력 N L)
    - A : 학생 추가 질의를 나타내는 기호
    - S : 학생의 학번
    - E : 학생의 이메일 주소
    - N : 질의에 대한 Probe 횟수
    - L : Load factor (소수 셋째자리에서 반올림)
  - 수강과목 추가 (질의 U S C, 출력 N T)
    - U : 수강과목 추가 질의를 나타내는 기호
    - C : 추가한 수강과목의 학점
    - T : 학생의 변경된 총 학점
    - 학번이 존재하지 않는다면 Not found 출력
    - 추가 시 총 학점이 24점을 초과한다면 Exceeded 출력
  - 이메일 주소 변경 (질의 M S E, 출력 N)
    - M : 이메일 주소 변경 질의를 나타내는 기호
    - 학번이 존재하지 않는다면 Not found 출력
  - 학생정보 출력 (질의 P S, 출력 N S T E)
    - P : 학생정보 출력 질의를 나타내는 기호
    - 학번이 존재하지 않는다면 Not found 출력
  - 프로그램 종료 (질의 Q)
    - Q : 프로그램 종료 질의를 나타내는 기호

## ■ 개발환경

- 운영체제 : Windows 10 Home K
- 컴파일러 : Visual Studio 2013
- 언어 : C++

# 필요한 자료구조 및 기능

## ■ 자료구조



$N = 7$ ,  $q = 3$  인 구조체 배열을 사용한 해시 테이블

$$h1(k) = k \bmod N$$

$$h2(k) = q - k \bmod q$$

$h1(k)$ 로 주소 값을 찾고 이미 다른 데이터가 있다면  $h2(k)$ 만큼 뛰어서  $(h1(k)+h2(k))\%N$ 에 다른 데이터가 있는지 확인하는 과정 반복하여 가능한 주소에 데이터 저장

### - 해시 테이블

- 임의의 데이터에 대한 탐색, 삽입, 삭제 연산의 평균수행시간이  $O(1)$ 으로 빠른 자료구조
- key값을 해시함수에 인자로 집어넣어 배열의 주소 값을 구한 후 해당 주소에 데이터를 저장하는 방식
- 이중해싱은 두 개의 해시함수를 사용하여 저장 가능한 주소에 데이터를 저장하는 방식으로 이용 시 충돌확률이 줄어들기 때문에 성능을 향상 시킬 수 있다
- 암호화, 문자열 검색 등에 사용 된다

## ■ 기능

- 파일 정보를 이용한 해시테이블 구축
  - 파일을 읽어서 테이블 크기에 맞게 구조체 배열을 동적 할당
  - 저장된 정보를 구조체 배열에 이중해싱 방법으로 저장
- 학생 정보 추가
  - 이중해싱으로 빈 공간에 새로운 학생의 정보 저장
  - 다른 주소로 갈 때 마다 probe 횟수 증가
  - probe와 loadfactor(회원의 수/테이블의 크기) 출력
- 수강 과목 추가
  - 이중해싱으로 key값에 맞는 정보를 찾음
  - 학점을 추가해주고 요구사항에 맞게 출력
- 이메일 주소 변경
  - 이중해싱으로 key값에 맞는 정보를 찾고 이메일 주소 변경
  - probe와 상황에 따라 요구사항에 맞게 출력
- 학생 정보 출력
  - 이중해싱으로 key값에 맞는 정보를 찾음
  - 요구사항에 맞게 학생 정보 출력

# 기능별 알고리즘 명세

## ■ 파일 정보를 이용한 해시테이블 구축

### - 수도코드

Algorithm HashTable(filename)

read N M q with filename

create Student type array s of size N and initialize

for i <- 0 to M-1

read StudentInfo with filename

if s[h1] is not empty

while s[(h1+h2)%N] is not empty

h1 <- (h1+h2)%N

s[(h1+h2)%N] <- StudentInfo

else

s[h1] <- StudentInfo

시간복잡도.

:  $O(N)$

N 크기의 구조체 배열의 key값을 모두 NULL로 초기화해야 하기 때문에 시간복잡도는  $O(N)$ 이 된다.

## ■ 학생 정보 추가

### - 수도코드

```
Algorithm put(key, email)
    if s[h1] is not empty
        probe++
        while s[(h1+h2)%N] is not empty
            h1 <- (h1+h2)%N
            probe++
        s[(h1+h2)%N] <- key, email, score(0)
    else
        s[h1] <- key, email, score(0)
M++
print probe, loadfactor
```

### - 시간복잡도

:  $O(N)$

최악의 경우 모든 정보와 비교 후에 정보를 저장할 수 있으므로  $O(N)$ 이 된다.

## ■ 수강 과목 추가

### - 수도코드

```
Algorithm addScore(key, score)
    if s[h1] is empty
        print probe Not found and return
    else if s[h1] is not key
        probe++
        while s[(h1+h2)%N] is not key
            h1 <- (h1+h2)%N
            probe++
        if s[(h1+h2)%N] is NULL
            print probe Not found and return
        else
            if (score>24)
                print probe Exceeded and return
            s[(h1+h2)%N] += score
            print probe score
    else
        if(score>24)
            print probe Exceeded
            return
        s[h1] += score
        print score
```

### - 시간복잡도

:  $O(N)$

최악의 경우 모든 정보와 비교 후에 수강과목을 추가한 학생의 정보를 찾을 수 있으므로  $O(N)$ 이 된다.

## ■ 이메일 주소 변경

### - 수도코드

```
Algorithm changeEmail(key, email)
    if s[h1] is empty
        print probe Not found and return
    else if s[h1] is not key
        probe++
        while s[(h1+h2)%N] is not key
            h1 <- (h1+h2)%N
            probe++
        if s[(h1+h2)%N] is NULL
            print probe Not found and return
        else
            s[(h1+h2)%N] <- email
            print probe and return
    else
        s[h1] <- email
        print probe and return
```

### - 시간복잡도

:  $O(N)$

최악의 경우 모든 정보와 비교 후에 이메일 변경해야 하는 학생의 정보를 찾을 수 있으므로  $O(N)$ 이 된다.



## ■ 학생 정보 출력

### - 수도코드

Algorithm print(key)

if s[h1] is empty

print probe Not found and return

else if s[h1] is not key

probe++

while s[(h1+h2)%N] is not key

h1 <- (h1+h2)%N

probe++

if s[(h1+h2)%N] is NULL

print probe Not found and return

else

print probe s[(h1+h2)%N] and return

else

print probe s[h1] and return

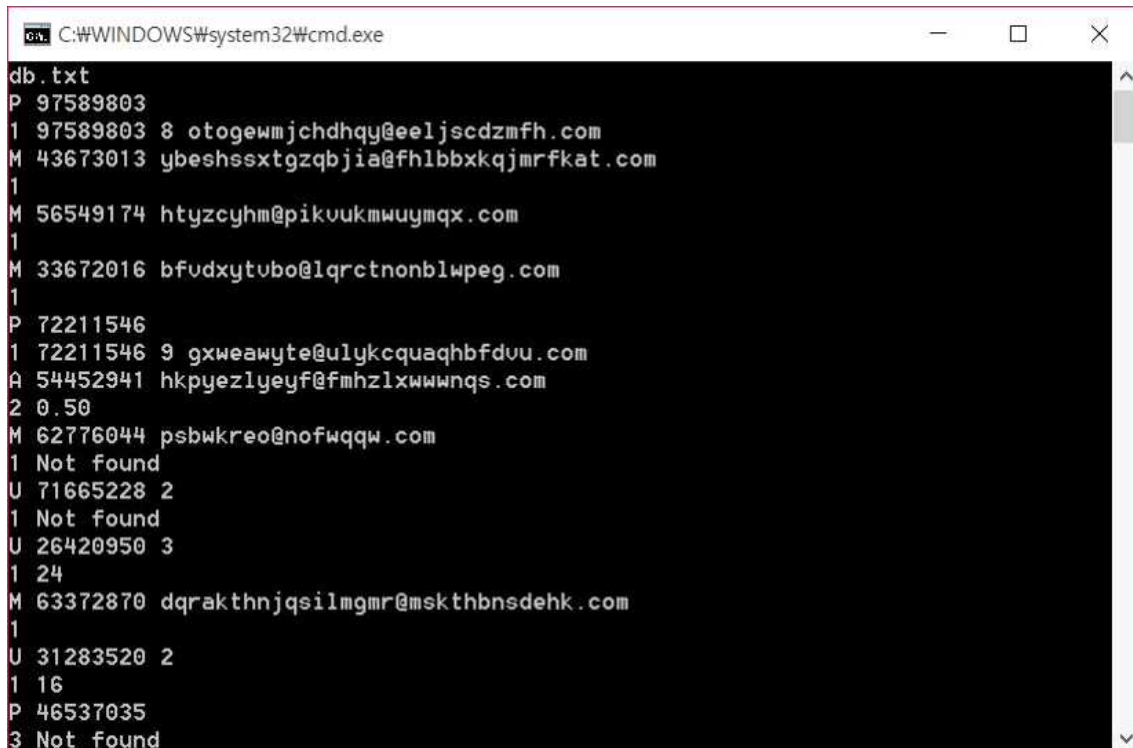
### - 시간복잡도

: O(N)

최악의 경우 모든 정보와 비교 후에 알고자 하는 학생의 정보를 찾을 수 있으므로 O(N)이 된다.

# 인터페이스 및 사용법

## ■ 스크린샷



```
db.txt
P 97589803
1 97589803 8 otogewmjchdhqy@eeljscdzmfh.com
M 43673013 ybeshssxtgzqbja@fh1bbxkqjmrkat.com
1
M 56549174 htyzcyhm@pikuukmwuymqx.com
1
M 33672016 bfvdxytubo@lqrctnonblwpeg.com
1
P 72211546
1 72211546 9 gxweawyte@ulykcquaqbfdvu.com
A 54452941 hkpyezlyeyf@fmhxlwwwnqs.com
2 0.50
M 62776044 psbwkreo@nofwqqw.com
1 Not found
U 71665228 2
1 Not found
U 26420950 3
1 24
M 63372870 dqrakthnjqsilmgmr@mskthbnsdehk.com
1
U 31283520 2
1 16
P 46537035
3 Not found
```

- query.txt의 일부분을 입력했을 때의 결과이다.

## ■ 사용법

- 앞서 요구사항에 설명한대로 질의형식대로 입력을 주면 그에 대한 출력이 나올 것이다.

# 평가 및 개선사항

## ■ 본 결과의 장점 및 단점

- 장점 : 이중해싱을 이용하여 설계했기 때문에 파일정보를 이용한 해시테이블을 구축하는 연산을 제외한 모든 연산의 기대 수행시간이  $O(1)$ 으로 굉장히 빠르다.
- 단점 : 최악수행시간은  $O(N)$ 이기 때문에 해시테이블에 저장된 데이터가 많으면 많을수록, 즉 load factor가 1에 가까울수록 성능이 떨어진다.

## ■ 향후 개선방향

- 해시테이블의 크기를 늘려서 load factor를 작게 하면 성능이 좋은 학생 관리 프로그램이 될 수 있다.