Data Structure

Week 13

KyuDong SIM

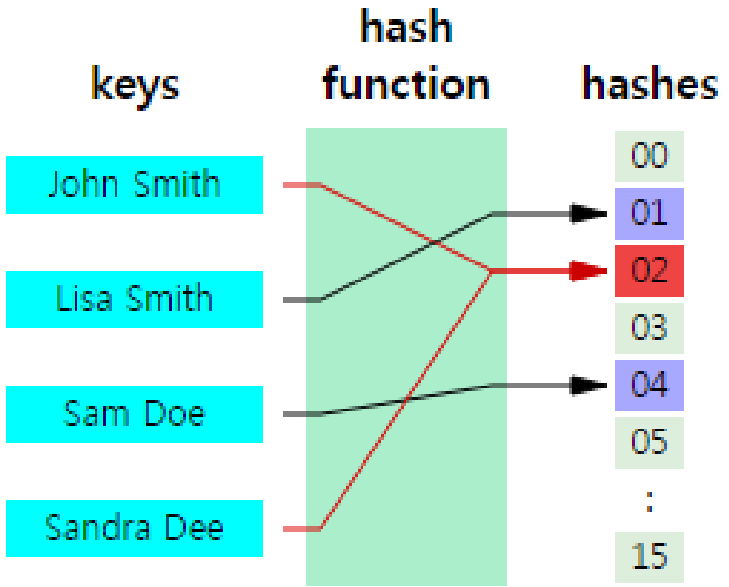
1. 이번 주 실습 내용

- Hash 구현

Hash

- 임의의 데이터를 해시 값으로 매핑하는 알고리즘
- 데이터가 달라도 해시 값은 같을 수 있다.
- 해시 값이 같아도 데이터는 다를 수 있다.

- 용도
  - 데이터베이스 내의 항목들을 색인하고 검색하는데 사용
  - 전자서명을 암호화하고 복호화 하는데 사용



실습에 사용할 Hash

- $\mathrm{H(k)} = \lfloor (m(kA \bmod 1)) \rfloor$

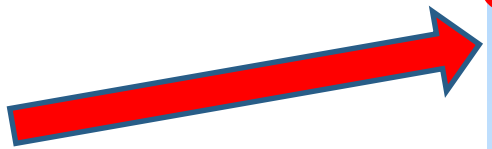
- K : 입력 값
- m : Hash 크기
- A : constant float between 0 and 1
- ⌊ ⌋ : 내림

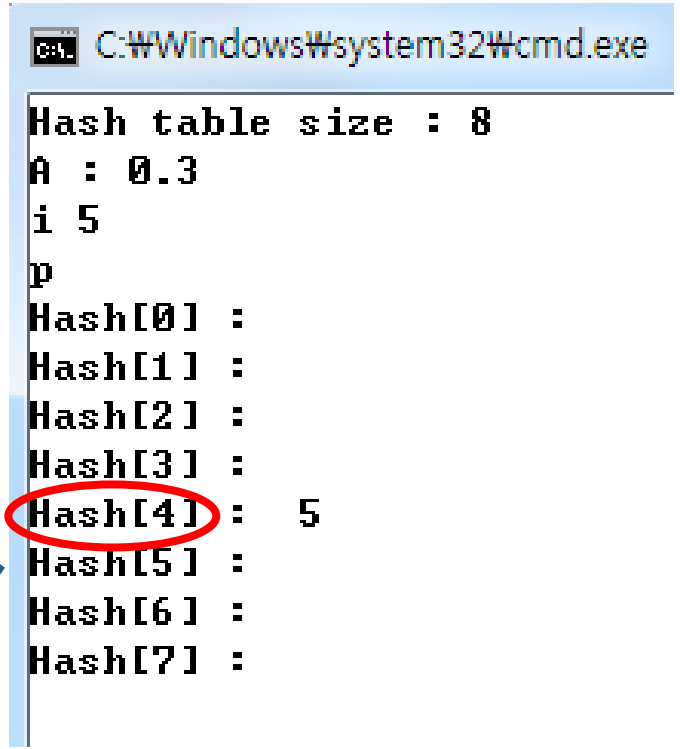
실습에 사용할 Hash

- $H(k) = \lfloor (m(kA \bmod 1)) \rfloor$

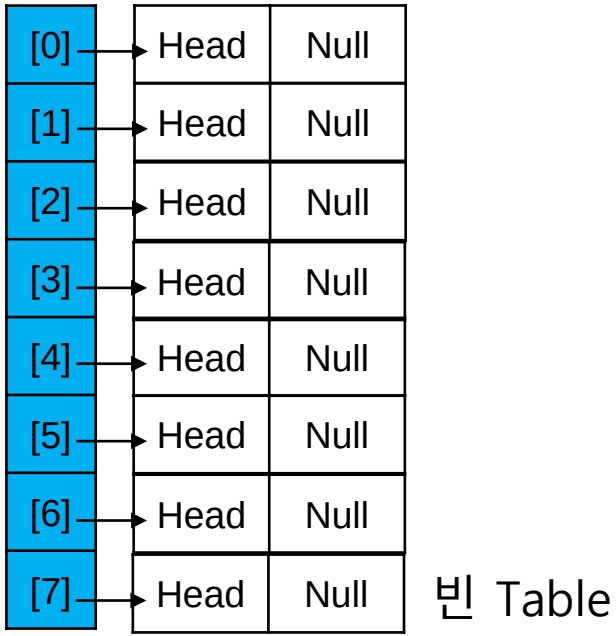
- K : **5** (입력 값)
- m : 8 (Hash 크기)
- A : 0.3 (0-1 float)
- ⌊ ⌋ : 내림

  - kA = 1.5
  - kA mod 1 = 0.5
  - m(kA mod 1) = 4
  - $\lfloor (m(kA \bmod 1)) \rfloor$ = 4

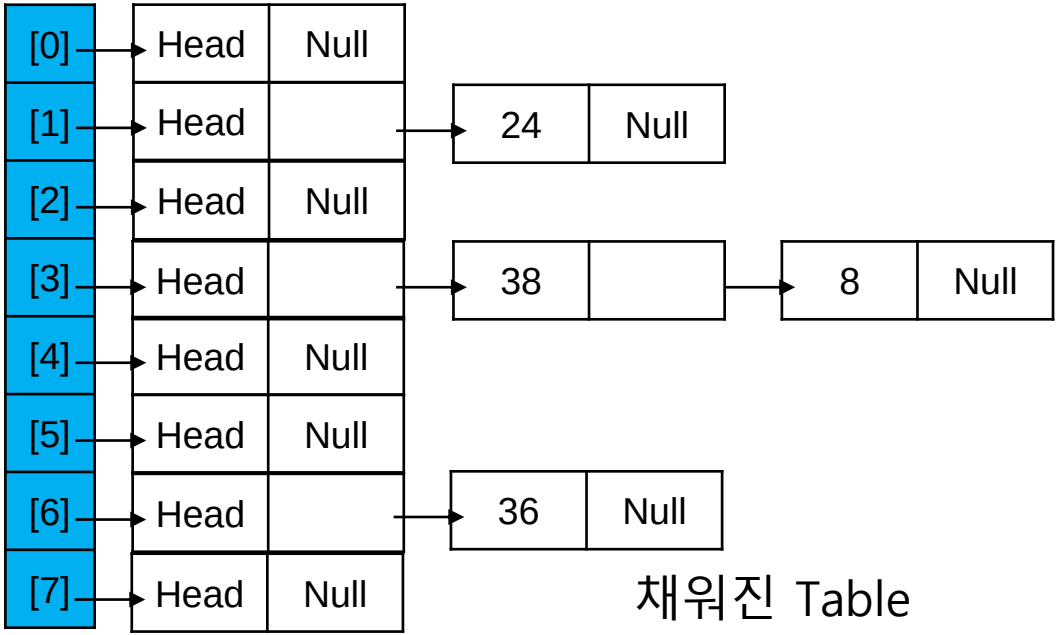




Hash Table (실습)



빈 Table

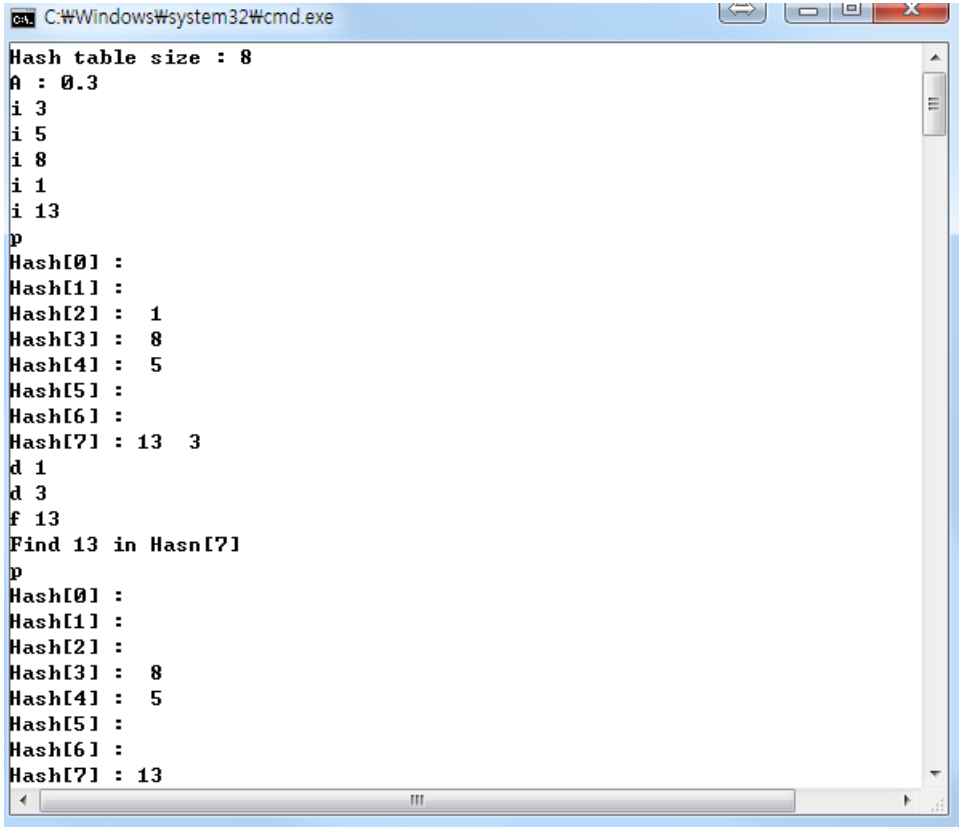


채워진 Table

실습 Command

- i x : x 값을 hash table 에 넣는다. x가 이미 존재하면 작동하지 않음.
- d x : hash table 에서 x값을 지운다. x가 존재하지 않으면 작동하지 않음.
- f x : hash table에서 x값을 찾는다. x값을 찾은 hash table 의 index를 출력한다.
- p : 모든 hash table을 출력한다.
- q : 프로그램을 종료한다.

실습 결과 예


C:₩Windows₩system32₩cmd.exe
Hash table size : 8
A : 0.3
i 3
i 5
i 8
i 1
i 13
p
Hash[0] :
Hash[1] :
Hash[2] : 1
Hash[3] : 8
Hash[4] : 5
Hash[5] :
Hash[6] :
Hash[7] : 13 3
d 1
d 3
f 13
Find 13 in Hasn[7]
p
Hash[0] :
Hash[1] :
Hash[2] :
Hash[3] : 8
Hash[4] : 5
Hash[5] :
Hash[6] :
Hash[7] : 13

Data structure

```
typedef struct ListNode *position;
typedef position List;
typedef struct HashTbl *HashTable;

struct ListNode
{
    int Element;
    position Next;
};

struct HashTbl
{
    int TableSize;
    float A;
    List *TheLists;
};
```

ListNode는 각 key 값이 저장된 Node

HashTbl(HashTable)은 hash의 크기, 상수 A 값과 테이블 Lists 값을 가짐

createHash (예시)

```
HashTable createHash(int size, float A)
{
    HashTable H;
    H = (struct HashTbl*) malloc(sizeof(struct HashTbl));
    H->TableSize = size;
    H->A = A;

    H->TheLists = (position*) malloc(sizeof(position)*size);

    for(int i=0; i<size ;i++)
    {
        H->TheLists[i] = (struct ListNode*) malloc(sizeof(struct ListNode));
        H->TheLists[i]->Next=NULL;
    }

    return H;
}
```

Hash 생성 및 초기화

생성된 Hash의 크기, 상수 A 값 설정

Hash 구조체의 메모리할당

리스트(배열)의 메모리 할당

Node의 메모리할당

Find

```
position Find(int key, HashTable H)
{
    position P;
    List L;

    L = H->TheLists[hx(key,H)];
    P = L->Next;

    while(P !=NULL && P->Element !=key)
        P = P->Next;

    return P;
}
```

Key 값을 찾아 해당 Node의 포인터를 반환하는 함수

hx : index값을 찾는 H(k) 함수

P = L->Next : 첫 node는 헤더라 다음 node부터 검색

Key값을 찾거나 node의 link가 끝날때까지 loop

Insert

```
void Insert(int key, HashTable H)
{
    position Pos, newCell;
    List L;

    Pos = Find(key, H);

    if(Pos == NULL)
    {
        newCell = (struct ListNode *)malloc(sizeof(struct ListNode));

        L = H->TheLists[hx(key,H)];
        newCell->Next = L->Next;
        newCell->Element = key;
        L->Next = newCell;

    }
}
```

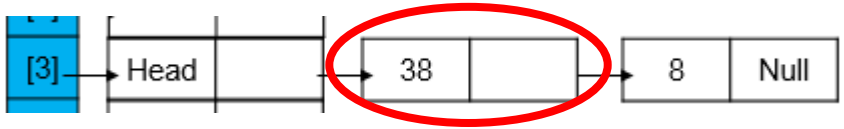
Key 값이 hash에 없을 경우 key를 입력하는 함수

Node의 메모리 할당

Index를 찾아 node의 포인터를 연결해줌

Header다음으로 삽입됨

(8 삽입 후 38을 삽입)



제출 및 알림

수업 중 확인 or 메일제출 (학번 써주세요)

메일 제출 :

주소 : (89kdsim@naver.com)

기한 : ~2016-06-01