

# 자료구조응용

## 23. Static Hashing-Random Probing, Chaining (15점)

2022.05.30

- [Random Probing] 다음과 같은 division 해시함수와 Random Probing을 사용하는 해시 테이블에 대해 search, insert 함수를 작성하고 그 결과를 출력하는 프로그램을 작성하라. (8점)

<해싱조건>

입력파일(input.txt) :

5 8 13 7 21 23

해싱함수( $h(k)$ ):  $k \% b$

키 탐색순서 -  $h(k), (h(k)+s(i)) \% b, 1 \leq i \leq b-1$ ,  $s(i)$ 는 유사난수(pseudo random number)

난수생성 :  $s(i)$ 는  $1 \leq i \leq b-1$  시퀀스에 대해 1에서  $b-1$  범위의 난수를 정확하게 한 번씩 생성해야 하며, 매 탐색마다 동일한 seed를 사용하여야 함. 이러한 특징의 난수 생성기를 직접 구현하는 대신, C 언어의 srand, rand함수를 활용함

버킷 수 (b) : 8

슬롯 수 (s) : 1

<예>

Input sequence : 5 8 13 7 21 23  
Random numbers : 5 2 3 7 1 4 6  
Hash table : 8 buckets with 1 slot

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
ht	8	23	13		21	5		7

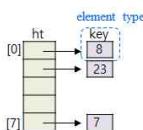
$k=5 : h(k) = 5 \% 8 = 5$   
 $k=8 : h(k) = 8 \% 8 = 0$

$k=13 : h(k) = 13 \% 8 = 5$   
 $(h(k)+s(1)) \% 8 = (5+5) \% 8 = 2$

※ 구현

$k=7 : h(k) = 7 \% 8 = 7$

$k=21 : h(k) = 21 \% 8 = 5$   
 $(h(k)+s(1)) \% 8 = (5+5) \% 8 = 2$   
 $(h(k)+s(2)) \% 8 = (5+2) \% 8 = 7$   
 $(h(k)+s(3)) \% 8 = (5+3) \% 8 = 0$   
 $(h(k)+s(4)) \% 8 = (5+7) \% 8 = 4$



$k=23 : h(k) = 23 \% 8 = 7$   
 $(h(k)+s(1)) \% 8 = (7+5) \% 8 = 4$   
 $(h(k)+s(2)) \% 8 = (7+2) \% 8 = 1$

### <실행순서>

- ① 해싱테이블(ht)을 element\* 타입의 구조체포인터 배열의 전역변수로 선언하고 초기화한다.
    - \* element는 key 요소만으로 구성된 구조체이다.
    - \* 입력되는 key 값은 0보다 큰 정수이다.
  - ② 사용자로부터 seed를 입력받는다.
  - ③ 1에서 b-1 범위의 난수 시퀀스를 중복 없이 생성해서 전역변수인 배열에 저장한다.
  - ④ 입력파일로부터 읽은 key값은 element 타입의 구조체를 동적으로 할당받아 저장하고 그 주소를 해당 버킷에 저장한다.
- \* h, s, insert 함수를 정의하여 사용
- 해싱테이블(ht)에 더 이상 추가할 수 없을 때는 적절한 메시지를 출력하고 종료한다.
  - 중복된 key가 있을 경우에는 적절한 메시지를 출력하고 종료한다.
- ⑤ 해싱테이블을 최종 생성한 후 인덱스 순서대로 key를 출력하라.
  - ⑥ 사용자로부터 키를 입력받아 탐색 후 (key, 비교횟수)를 출력하되 0이 입력될 때 까지 반복한다. \* search 함수를 정의하여 사용

### <실행결과>

```
C:\Windows\system32\cmd.exe
key sequence from file : 5 8 13 7 21 23
input seed >> 1

randNum[1] : 7
randNum[2] : 2
randNum[3] : 6
randNum[4] : 4
randNum[5] : 3
randNum[6] : 1
randNum[7] : 5

key
ht[ 0] : 8
ht[ 1]:
ht[ 2]:
ht[ 3]: 21
ht[ 4]: 13
ht[ 5]: 5
ht[ 6]: 23
ht[ 7]: 7

input 0 to quit
key to search >> 5
key : 5, the number of comparisions : 1

input 0 to quit
key to search >> 8
key : 8, the number of comparisions : 1

input 0 to quit
key to search >> 13
key : 13, the number of comparisions : 2

input 0 to quit
key to search >> 7
key : 7, the number of comparisions : 1

input 0 to quit
key to search >> 21
key : 21, the number of comparisions : 4

input 0 to quit
key to search >> 23
key : 23, the number of comparisions : 2

input 0 to quit
key to search >> 1
it doesn't exist!

input 0 to quit
key to search >> 0
계속하려면 아무 키나 누르십시오 . . .
```

2. [Chaining] 다음과 같이 입력파일로부터 문자열을 입력받아 해싱테이블에 추가하는 프로그램을 작성하라. (7점)

<해싱조건>

입력파일(input.txt) :

```
acos atoi char define exp ceil cos float floor ctime
```

\* 입력문자열의 최대 크기는 10임을 가정한다.

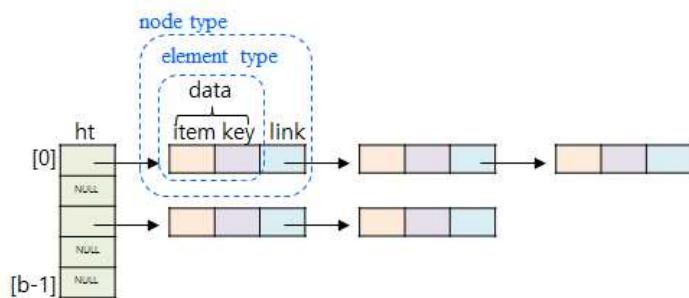
**키 변환함수** : 각 입력문자열을 0 이상의 정수로 바꿈 (Program 8.1)  
반환된 정수를 해싱함수의 입력 k로 사용

**해싱함수( h(k) )** :  $k \% b$  연산 결과를 반환함

**버킷 수 (b)** : 11

\*각 버킷은 체인으로 구성함

<자료구조>



<실행순서>

- ① 전역변수로 해시테이블(ht)을 선언하고, STRING\_MAX 및 BUCKET\_SIZE 기호상수를 각각 11로 정의한다.
- ② 입력파일로부터 읽은 각 문자열과 해당 key값은 `node` 타입의 구조체를 동적으로 할당받아 저장하고 `버킷의 체인에` 추가한다. 체인의 각 노드는 `element type`의 `data`, `nodePointer` 타입의 `link`로 구성되며, `element` 타입은 `key`와 `item(입력받은 문자열)`으로 구성된다. \* `insert` 함수
- ③ 해싱테이블을 최종 생성한 후 인덱스 순서대로 `key`를 출력하라.
- ④ 사용자로부터 키를 입력받아 탐색 후 (`item`, `key`, `비교횟수`)를 출력하되 ^Z가 입력될 때 까지 반복한다. \* `search` 함수(Program 8.4)

### <함수정의>

insert 함수

- search(Program 8.4) 함수를 참고하여 정의
- 노드는 흠 버킷 체인의 마지막에 추가됨
- 중복된 key가 있을 경우에는 적절한 메시지를 출력하고 종료함

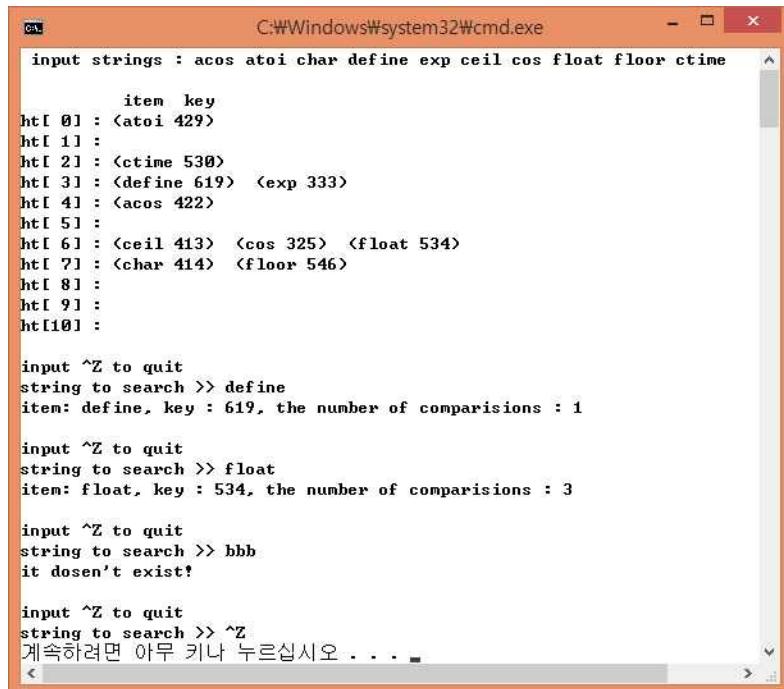
---

```
element* search(int k)
{ /* search the chained hash table ht for k, if a pair with
   this key is found, return a pointer to this pair;
   otherwise, return NULL.
   nodePointer current;
   int homeBucket = h(k);
   /* search the chain ht[homeBucket] */
   for (current = ht[homeBucket]; current;
        current = current->link)
       if (current->data.key == k) return &current->data;
   return NULL;
}
```

---

**Program 8.4:** Chain search

### <실행결과>



```
C:\Windows\system32\cmd.exe
input strings : acos atoi char define exp ceil cos float floor ctime
               item  key
ht[ 0] : <atoi 429>
ht[ 1] :
ht[ 2] : <ctime 530>
ht[ 3] : <define 619>  <exp 333>
ht[ 4] : <acos 422>
ht[ 5] :
ht[ 6] : <ceil 413>  <cos 325>  <float 534>
ht[ 7] : <char 414>  <floor 546>
ht[ 8] :
ht[ 9] :
ht[10] :

input ^Z to quit
string to search >> define
item: define, key : 619, the number of comparisions : 1

input ^Z to quit
string to search >> float
item: float, key : 534, the number of comparisions : 3

input ^Z to quit
string to search >> bbb
it doesn't exist!

input ^Z to quit
string to search >> ^Z
계속하려면 아무 키나 누르십시오 . . .
```

## ■ 제출 형식

- 솔루션 이름 : DS 23
- 프로젝트 이름 : 1, 2
- 각 소스파일에 주석처리  
“학번 이름”  
“본인은 이 소스파일을 다른 사람의 소스를 복사하지 않고 직접 작성하였습니다.”
- 실행화면을 캡쳐한 보고서를 작성 후 pdf 파일로 변환하여 솔루션 폴더에 포함
- 솔루션 정리 메뉴를 수행 후 전체 솔루션을 “학번.zip”으로 압축하여 제출

## ■ 주의

- 소스 복사로는 실력향상을 기대할 수 없습니다!!!
- 1차 마감: 수업 다음날 자정
- 2차 마감: 없음
- 위 마감 이외의 제출은 허용하지 않습니다. ( 이메일 제출 불가! )