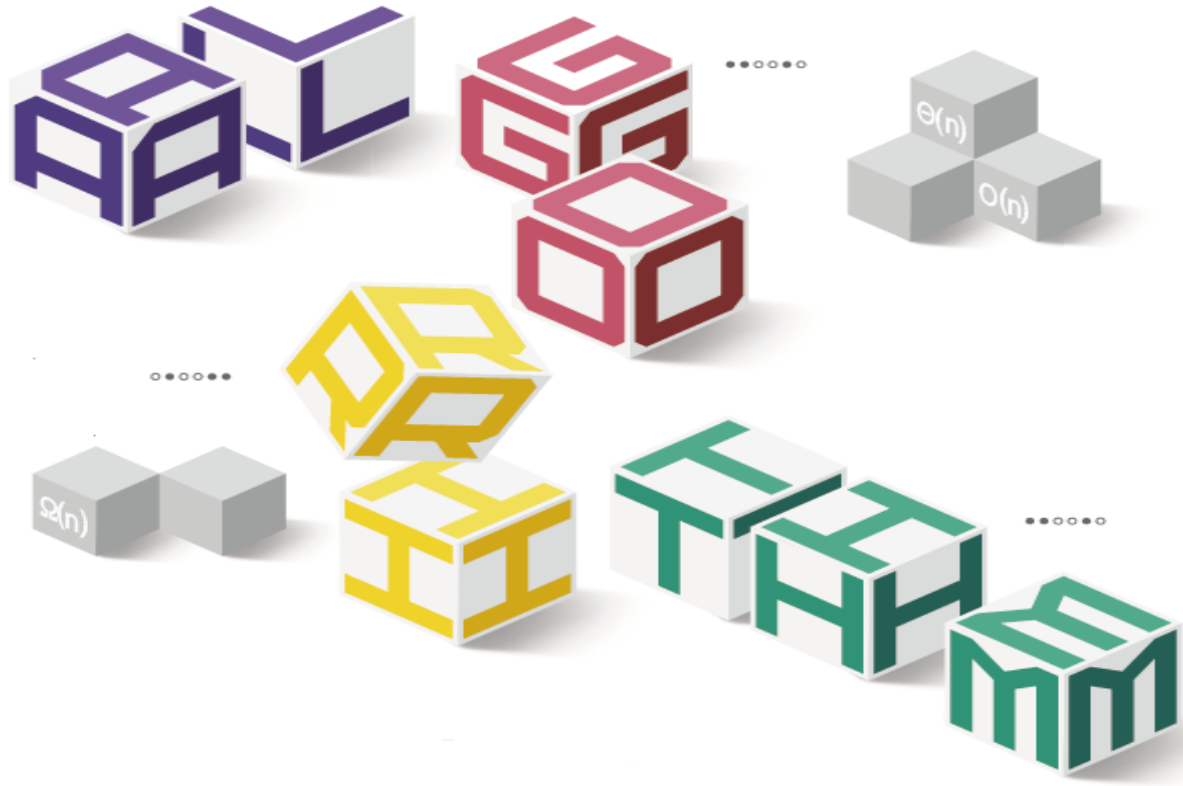


관계 중심의 사고법

쉽게 배우는 알고리즘

12주차
해시테이블

해시테이블


1. 해시테이블-해시테이블과 해시함수
12명
0명
1명
(0)
강의 보기
2. 해시테이블 검색성능, 특징 및 원리
11명
1명
1명
(0)
강의 보기
3. 해시함수 및 프로그램
11명
1명
1명
(0)
강의 보기


NOW
12주차
2020-05-30 ~ 2020-06-06
1.Collision 충돌
4명
0명
9명
(0)
강의 보기
Lab-Hash Function-1
4명
0명
9명
(0)
강의 보기
Lab-Hash Function-2
4명
0명
9명
(0)
강의 보기
Lab Hash Table Set-1
4명
0명
9명
(0)
강의 보기
2. Chaining, 체이닝
9명
0명
10명
(0)
강의 보기
3. Linear Probe, 선형조사
9명
0명
10명
(0)
강의 보기
4. QuadraticProbe-아차원조사
9명
0명
10명
(0)
강의 보기
5. Loadfactor-적재율
9명
0명
10명
(0)
강의 보기

해시 테이블 개요

- 가장 빠른 검색 성능을 제공($\Theta(1)$)
- 용도
  - 키 암호 알고리즘
  - 컴파일러의 심볼테이블
  - 데이터베이스 저장 스킴,
  - 데이터 동기화 검증(rsync, dropbox)
  - 개인 정보 보호 인터넷 주소 암호화(Yahoo 의 경우)

해시함수

```
int OAHT_Hash( KeyType Key, int KeyLength, int TableSize )
{
    int i = 0;
    int HashValue = 0;

    for ( i=0; i<KeyLength; i++ )
    {
        HashValue = (HashValue << 3) + Key[i];
    }

    HashValue = HashValue % TableSize;
    return HashValue;
}
```

```
typedef char* KeyType;
typedef char* ValueType;
```

```
char key[50] = "BK";
int szHashTable = 13;
printf("key = %s, hash = %d₩n", key, OAHT_Hash(key, strlen(key), szHashTable));
```

해시함수

- 다음은 문자열 s 자료를 해쉬테이블에 저장하기 위해 사용하는 해쉬함수 hash_function() 을 보인 것이다.

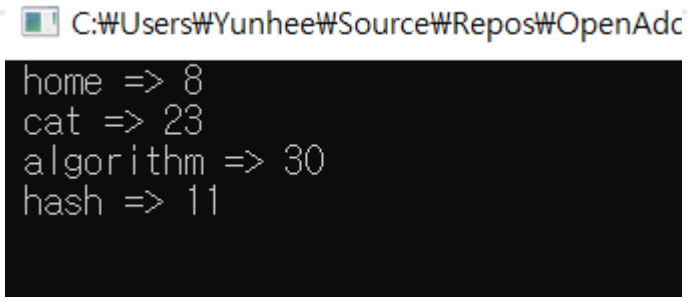
```
# define HASHSIZE 31
int hash_function(char *s)
{
    unsigned hashval;

    for (hashval = 0; *s != '₩0'; s++)
        hashval = *s + 31*hashval;
    return hashval % HASHSIZE;
```

해시함수

- 해쉬테이블의 크기는 HASHSIZE 에 설정한다. 자료 “home”, “cat”, “algorithm”, “hash” 의 해쉬함수의 반환값을 하도록 프로그램을 작성하여 결과를 보이시오.

```
printf(" %s => %d ₩n", "home", hash_function("home"));
printf(" %s => %d ₩n", "cat", hash_function("cat"));
printf(" %s => %d ₩n", "algorithm",  hash_function("algorithm"));
printf(" %s => %d ₩n",  "hash",  hash_function("hash"));
```



Collision

- Hash table의 한 주소를 놓고 두 개 이상의 원소가 자리를 다투는 것
  - Hashing을 해서 삽입하려 하니 이미 다른 원소가 자리를 차지하고 있는 상황
- Collision resolution 방법은 크게 두 가지가 있다
  - 체이닝(Chaining)
  - 개방주소방법(Open addressing)

Hash Function

65가 삽입되기전 구성된 해시테이블을 보이시오

What happens when you try to insert: x = 65 ?

x = 65

f(x) = 5

f(x) = x % 15

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
_	_	47	_	_	35	36	_	_	129	25	2501	_	_	_
					65(?)									

This is called a collision.

Collision의 예

입력: 25, 13, 16, 15, 7

0	13
1	
2	15
3	16
4	
5	
6	
7	7
8	
9	
10	
11	
12	25

h(29) = 29 mod 13 = 3

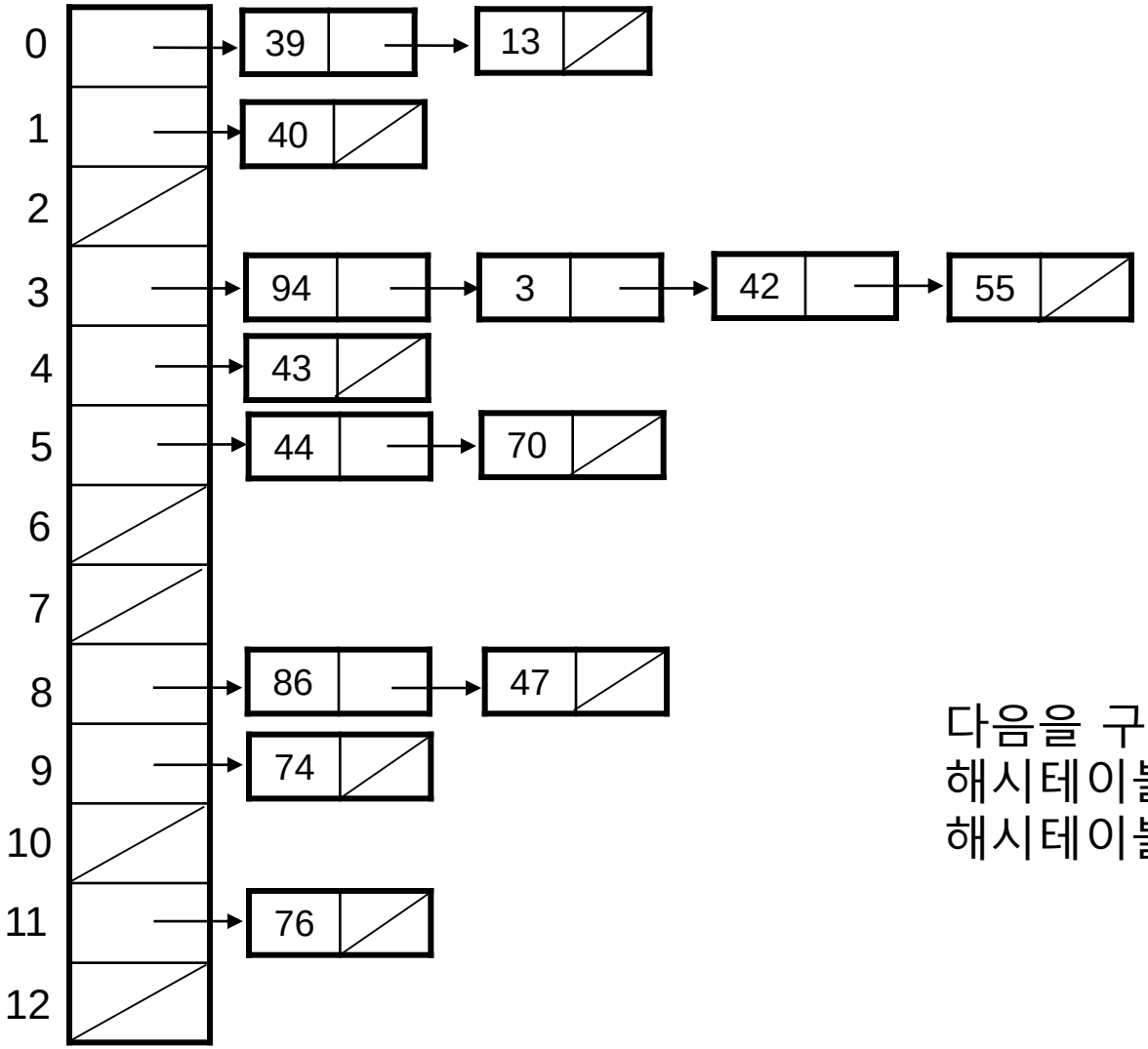
29를 삽입하려 하자 이미
다른 원소가 차지하고 있다!

Hash function h(x) = x mod 13

Collision Resolution

- 체이닝(Chaining)
  - 같은 주소로 hashing되는 원소를 모두 하나의 linked list로 관리한다
  - 추가적인 linked list 필요
- 개방주소방법(Open addressing)
  - Collision이 일어나더라도 어떻게든 주어진 테이블 공간에서 해결한다
  - 추가적인 공간이 필요하지 않다

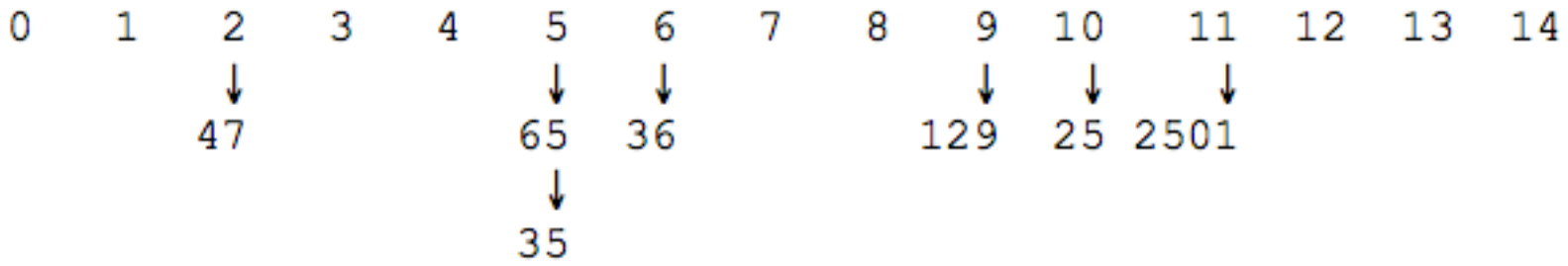
Chaining을 이용한 Collision Resolution의 예



다음을 구하시오.
해시테이블의 크기 : (가.)
해시테이블의 자료수 : (나.)

Lab. Chaining을 이용한 Collision Resolution

Let each array element be the head of a chain.



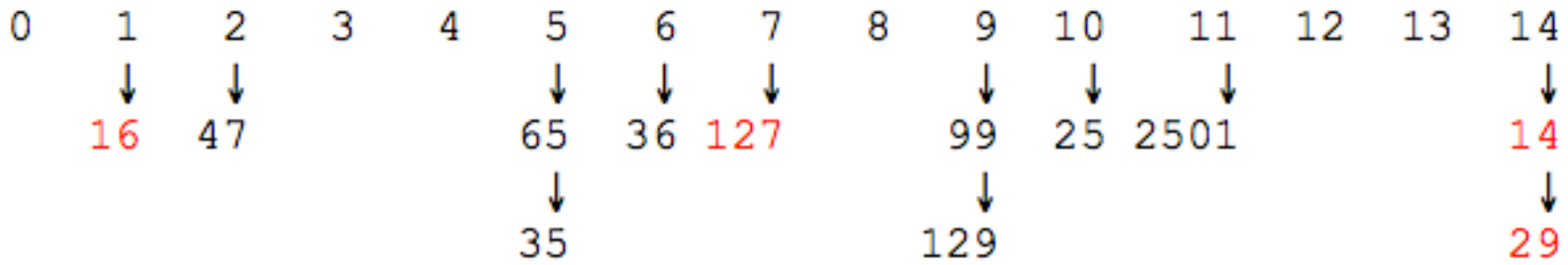
f(x) = x % 15

Where would you store: 29, 16, 14, 99, 127 ?

Solution. Chaining을 이용한 Collision Resolution

Let each array element be the head of a chain:

Where would you store: 29, 16, 14, 99, 127 ?



New keys go at the front of the relevant chain.

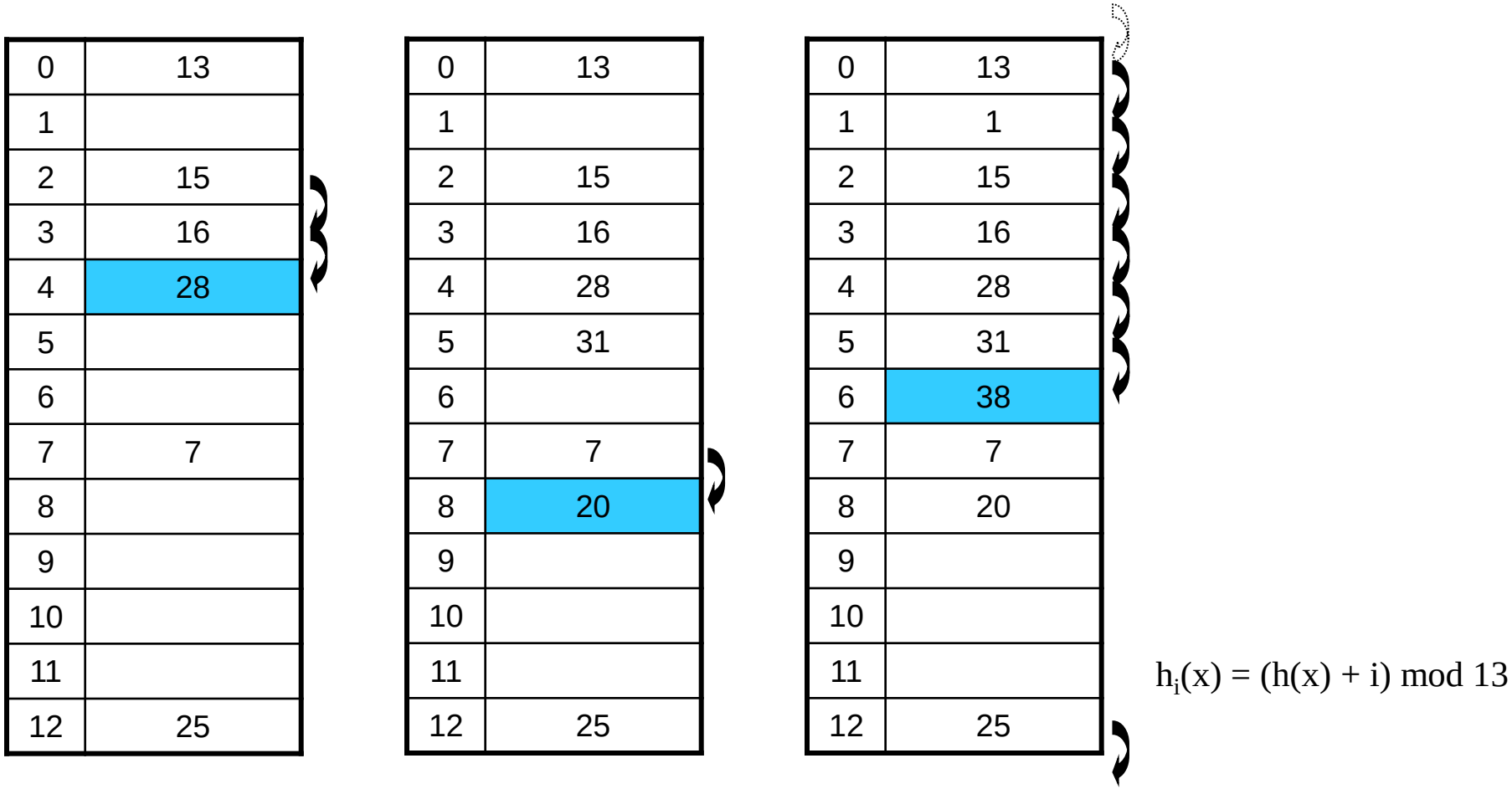
개방주소 방법

- 빈자리가 생길 때까지 해시값을 계속 만들어 낸다
  - $h_0(x)$, $h_1(x)$, $h_2(x)$, $h_3(x)$, ...
- 중요한 세가지 방법
  - 선형 조사
  - 이차원 조사
  - 더블 해싱

선형 조사Linear Probing

$h_i(x) = (h(x) + i) \bmod m$

예: 입력 순서 25, 13, 16, 15, 7, 28, 31, 20, 1, 38



선형 조사는 1차군집에 취약하다

1차군집: 특정 영역에 원소가 몰리는 현상

0	
1	
2	15
3	16
4	28
5	31
6	44
7	
8	
9	
10	
11	37
12	

1차군집의 예

이차원 조사Quadratic Probing

$$h_i(x) = (h(x) + c_1 i^2 + c_2 i) \bmod m$$

예: 입력 순서 15, 18, 43, 37, 45, 30

0	
1	
2	15
3	
4	43
5	18
6	45
7	
8	30
9	
10	
11	37
12	

$$h_i(x) = (h(x) + i^2) \bmod 13$$

이차원 조사는 2차군집에 취약하다

2차군집: 여러 개의 원소가 동일한
초기 해시 함수값을 갖는 현상

0	
1	
2	15
3	28
4	
5	54
6	41
7	
8	21
9	
10	
11	67
12	

← 2차군집의 예

더블 해싱 Double Hashing

$$h_i(x) = (h(x) + i\,f(x)) \bmod m$$

예: 입력 순서 15, 19, 28, 41, 67

0	
1	
2	15
3	67
4	
5	
6	19
7	
8	28
9	
10	41
11	
12	

$h_0(15) = h_0(28) = h_0(41) = h_0(67) = 2$

$h_1(67) = 3$

$h_1(28) = 8$

$h_1(41) = 10$

$h(x) = x \bmod 13$

$f(x) = x \bmod 11$

$h_i(x) = (h(x)+i\,f(x)) \bmod 13$

삭제시 조심할 것

개방주소 선형 조사시

0	13
1	1
2	15
3	16
4	28
5	31
6	38
7	7
8	20
9	
10	
11	
12	25

(a) 원소 1 삭제

H(25) =
H(38) =
12

0	13
1	
2	15
3	16
4	28
5	31
6	38
7	7
8	20
9	
10	
11	
12	25

(b) 38 검색, 선형조사 수행
문제발생
결과없음을 반환

0	13
1	DELETED
2	15
3	16
4	28
5	31
6	38
7	7
8	20
9	
10	
11	
12	25

(c) 표식을 해두면 문제없다

문제풀이

(pp. 224) 해시테이블에 원소가 차 있는 비율은 해시테이블의 성능에 중요한 영향을 미친다. 이를 적재율(load factor)라 한다.
다음의 해시테이블의 적재율을 계산하시오.

(pp.225) 해시테이블의 주소계산에 사용되는 해시함수가 갖는 두가지 성질 중 다음의 ( ) 내용을 채우시오.
- ( )
- 계산이 간단해야 한다

(pp.228-229) 충돌해결 방법 중 체이닝(chaining)을 사용하여 자료가 해시테이블에 저장될 때 적재율과 충돌횟수를 각각 작성하시오.

(pp.230-232) 선형조사와 이차원 조사의 문제점을 각각 작성하시오.

(pp. 233-236) 해시테이블에서 자료 삭제시 발생할 수 있는 문제점과 해결방법을 각각 작성하시오.

(pp. 237) 체이닝 방법을 사용할 때 적재율이 α일때, 실패하는 검색에 조사횟수의 기대치는 α이다.

과제

Test_Chaining.c 의 내용을 참고하여 자료를 해시테이블에 추가할 때 다음의 답하시오. 단, 해시테이블의 크기는 10, 13, 15에 대해 각각 결과를 구하시오.

1) 새롭게 추가된 자료의 해시값를 출력하시오.

2) 해시테이블의 적재율(Load factor)를 구하시오

3) 해시테이블의 충돌(collision)의 발생 횟수와 관련 키를 구하시오.

제공된 프로그램을 사용하여 과제를 수행하고 해당 과제내용별로 출력내용을 보이시오.