동국대학교
dongguk university

실습: Week 12

Data Structures

Contents

- Hashing
  - Hash dictionary (Linear Probing) 구현
- 실습
  - 실습 12. hash dictionary

The Symbol Table ADT

- BST
  - Average O(log n), worst-case O(n)
- 해싱(hashing)
  - Key 비교 안함
  - 해시 함수를 이용, key → 저장위치 를 바로 계산
    - Ex> Hash function f: id → id mod 11
      Search 731010: f(731010) = 731010 % 11 = 5 → box[5]
  - Very good expected performance: O(1)
  - Worst-case O(n)

Hash Table

- 동의어(synonyms)
  - 두 key i_1 과 i_2 가 $f(i_1) = f(i_2)$ 일때
  - Ex> f(char) = f(ceil) = 2
- 충돌(collision)
  - 두개의 다른 key 가 같은 버켓에 해싱되는 경우
  - Ex> hashing 'char' → hashing 'ceil'
- 오버플로우(overflow)
  - 새로운 key i 를 가득찬 버켓에 해싱시키는 경우
  - Ex> hashing 'char' → hashing 'ceil' → hashing 'clock'
  - Collisions = overflows if bucket size is 1

Hash Function

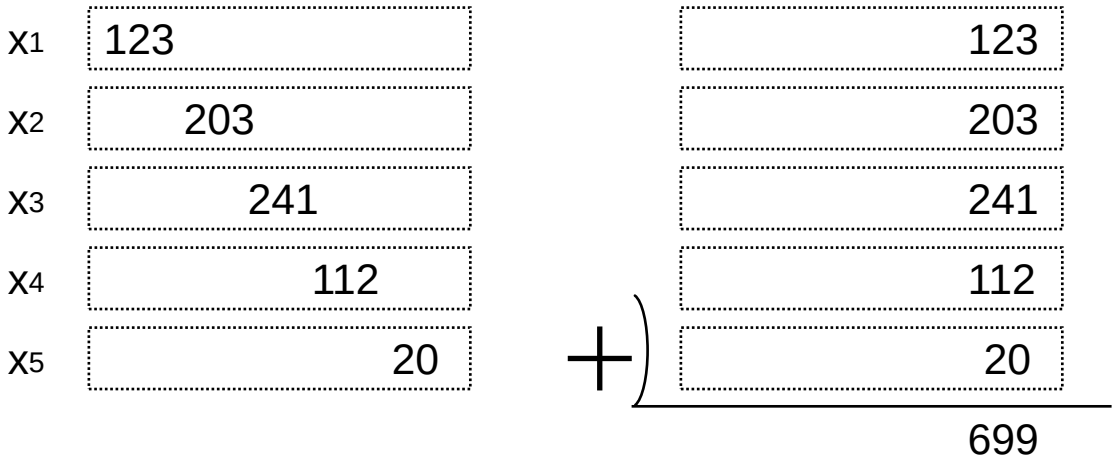
- 제산함수(division)
  - 모드 연산자(%) 사용

    $f_D(x) = x \% M$ (M: table size)
  - Range of bucket address: [0, M-1]
  - M의 선택이 중요
    - Id: 327, 107, 857, 497, 617, … , M = 10

      → all id's are hashed into bucket 7 !
    - Choose M as a prime number
    - Choose M such that it has no prime divisors less than 20

Hash Function

- 접지(folding)
  - key 를 여러 부분으로 나누어 더함
  - Range of bucket address: [0, 2^r-1]
    (각 부분이 r bits 일때)
  - Ex> identifier x = 12320324111220



Linear Probing

- Insert
  - Examine the hash table buckets

    ht[(f(x) + j) % TABLE_SIZE], 0 ≤ j ≤ TABLE_SIZE

    1) 버켓에 x 있음
       → 중복
    2) 버켓에 다른 key 있음
       → 다음 버켓 검사 (++j)
    3) 버켓이 비어있음
       → x 삽입
    4) j = TABLE_SIZE (제자리로 돌아옴)
       → 에러 (hash table full)

Linear Probing

- Search
  - Examine the hash table buckets

    ht[(f(x) + j) % TABLE_SIZE], 0 ≤ j ≤ TABLE_SIZE

    1) 버켓에 x 있음
       → 검색 성공
    2) 버켓에 다른 key 있음
       → 다음 버켓 검사 (++j)
    3) 버켓이 비어있음
       → 검색 실패 (x 없음)
    4) j = TABLE_SIZE (제자리로 돌아옴)
       → 검색 실패 (hash table full)

실습 12. hash dictionary

- Hashing (linear probing) 구현
- 명령어
  - R: Read data
    - 파일에서 <key data>들을 읽어 차례로 해시테이블에 insert
  - S: Search data
    - key 값을 받아서 (영어단어), 해시테이블을 search, data 값을 반환 (국어단어)
  - P: Print hash table
    - 현재 해시테이블의 내용을 출력
  - Q: Quit

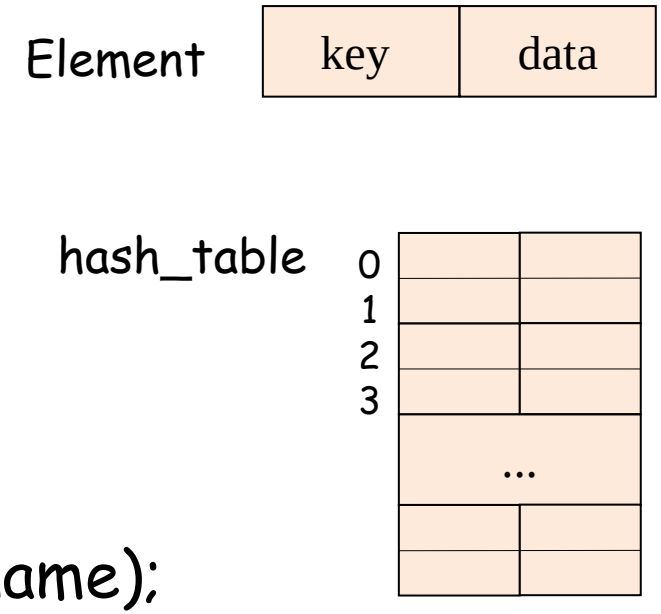
```
one 하나
two 둘
three 셋
...
```

dic.txt

자료구조 및 함수

```
typedef struct {
        char    key[100];
        char    data[100];
} Element;

Element hash_table[TABLE_SIZE];
```



- void build_dictionary(char *fname);
  - 파일에서 단어들을 읽어 해시테이블 구성 (insert)
- void hash_insert(char *key, char *data);
  - 해시테이블에 (key, data) 자료 삽입

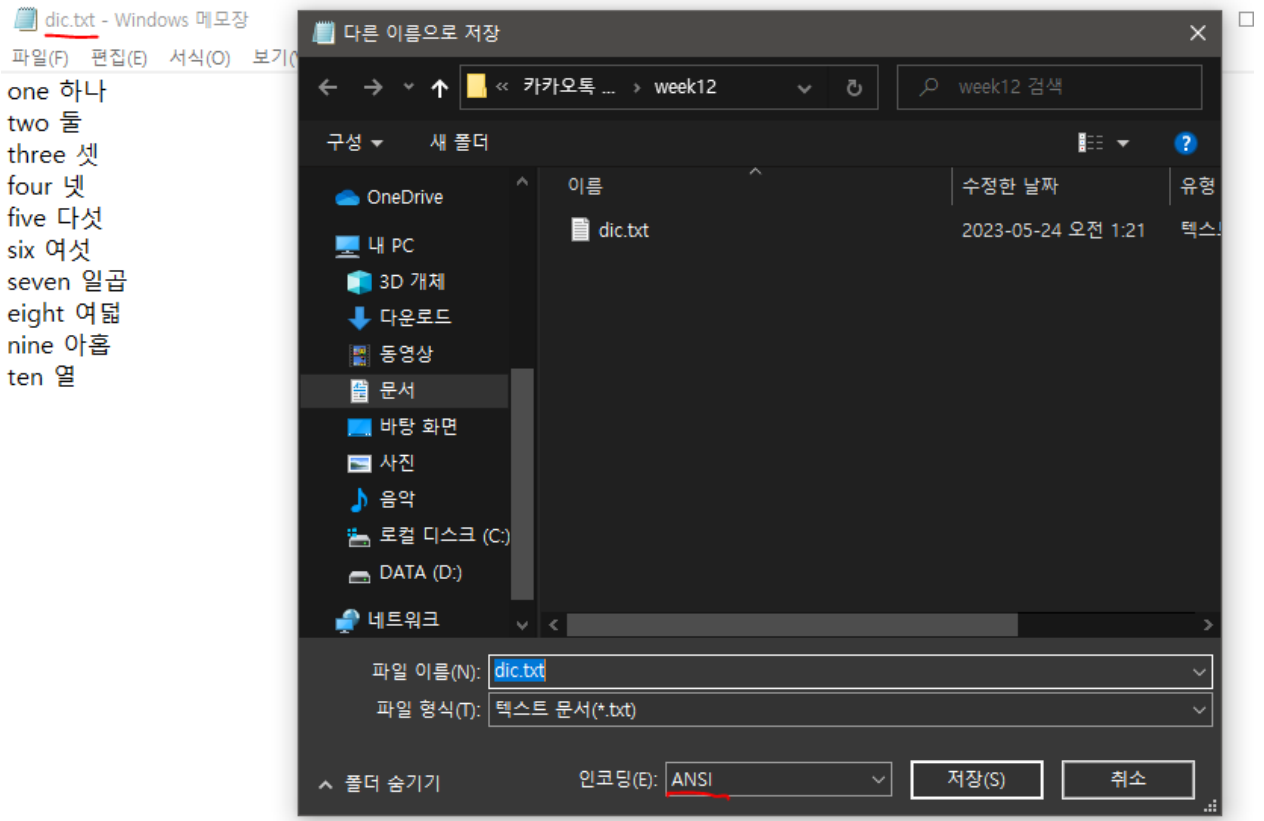
자료구조 및 함수

- char * hash_search(char *key);
  - 해시테이블에서 키값이 key인 자료를 검색, data를 반환
- int hash(char *key);
  - 해시 함수 (folding and division)
- int transform(char *key);
  - folding (key의 각 character 값을 더함)
- void hash_show();
  - 해시테이블의 key들을 차례로 출력

실습 12. 텍스트 파일 생성

one 하나
two 둘
three 셋
four 넷
five 다섯
six 여섯
seven 일곱
eight 여덟
nine 아홉
ten 열

dic.txt



인코딩 : ANSI

실습 12. 실행 예

one 하나
two 둘
three 셋
four 넷
five 다섯
six 여섯
seven 일곱
eight 여덟
nine 아홉
ten 열

dic.txt

```
Command> r
 Dictionary file name: dic.txt
 Total number of words: 10

Command> p
hash_table[0] = <nine , 아홉>
hash_table[1] = < , >
hash_table[2] = <four , 넷>
hash_table[3] = <three , 셋>
hash_table[4] = <six , 여섯>
hash_table[5] = <ten , 열>
hash_table[6] = < , >
hash_table[7] = < , >
hash_table[8] = <two , 둘>
hash_table[9] = <eight , 여덟>
hash_table[10] = <one , 하나>
hash_table[11] = <five , 다섯>
hash_table[12] = <seven , 일곱>
```

```
Command> s
 Word: one
 start_hash_value = 10
 Meaning: 하나
 Tatal number of comparison = 1

Command> s
 Word: five
 start_hash_value = 10
 Meaning: 다섯
 Tatal number of comparison = 2

Command> s
 Word: ten
 start_hash_value = 2
 Meaning: 열
 Tatal number of comparison = 4

Command> s
 Word: zero
 start_hash_value = 6
 No such word !
 Tatal number of comparison = 1

Command> s
 Word: eleven
 start_hash_value = 2
 No such word !
 Tatal number of comparison = 5
```

실습 12. 실행 예

one 하나
two 둘
three 셋
four 넷
five 다섯
six 여섯
seven 일곱
eight 여덟
nine 아홉
ten 열
eleven 열하나
twelve 열둘
thirteen 열셋
fourteen 열넷

dic.txt

```
Command> r
 Dictionary file name: dic.txt
 hash table is full
 Total number of words: 13

Command> p
hash_table[0] = <nine , 아홉>
hash_table[1] = <twelve , 열둘>
hash_table[2] = <four , 넷>
hash_table[3] = <three , 셋>
hash_table[4] = <six , 여섯>
hash_table[5] = <ten , 열>
hash_table[6] = <eleven , 열하나>
hash_table[7] = <thirteen , 열셋>
hash_table[8] = <two , 둘>
hash_table[9] = <eight , 여덟>
hash_table[10] = <one , 하나>
hash_table[11] = <five , 다섯>
hash_table[12] = <seven , 일곱>
```

```
Command> s
 Word: one
 start_hash_value = 10
 Meaning: 하나
 Tatal number of comparison = 1

Command> s
 Word: five
 start_hash_value = 10
 Meaning: 다섯
 Tatal number of comparison = 2

Command> s
 Word: ten
 start_hash_value = 2
 Meaning: 열
 Tatal number of comparison = 4

Command> s
 Word: zero
 start_hash_value = 6
 No such word !
 Tatal number of comparison = 13

Command> s
 Word: eleven
 start_hash_value = 2
 Meaning: 열하나
 Tatal number of comparison = 5
```

hash_dictionary.h

```
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define   TABLE_SIZE        13
#define   boolean    int
#define   true       1
#define   false      0

// Hash table
typedef struct {
            char       key[100];
            char       data[100];
} Element;
Element hash_table[TABLE_SIZE];

// For caomparison count
int num_comparison;
// For word count
int wcount;
```

hash_dictionary.h

```
// 파일에서 단어들을 읽어 해시테이블 구성
void build_dictionary(char *fname);

// 해시테이블에 (key, data) 자료 삽입
void hash_insert(char *key, char *data);
// 해시테이블에서 키값이 key인 자료를 검색, data를 반환
char * hash_search(char *key);
// 해시테이블의 key들을 차례로 출력
void hash_show();

// 해시 함수 (folding + division (TABLE_SIZE로 나눈 나머지))
int hash(char *key);
// folding (key의 각 character 값을 더함)
int transform(char *key);
```

hash_dictionary.c – main()

```
#include "hash_dictionary.h"

void main()
{
        char    c, fname[20];
        char    key[100], *data;
        printf("************* Command ************\n");
        printf("R: Read data, S: Search data    \n");
        printf("P: Print hash table, Q: Quit    \n");
        printf("**********************************\n");
        while (1) {
                printf("\nCommand> ");
                c = _getche();
                c = toupper(c);
                switch (c) {
                case 'R':
                        printf("\n Dictionary file name: ");
                        scanf("%s", fname);
                        build_dictionary(fname);
                        printf(" Total number of words: %d \n", wcount);
                        break;
```

hash_dictionary.c – main()

```
case 'S':
        printf("\n Word: ");
        scanf("%s", key);
        num_comparison = 1;
        data = hash_search(key);
        if (data)
                printf(" Meaning: %s \n", data);
        else
                printf(" No such word ! \n");
        printf(" Tatal number of comparison = %d \n", num_comparison);
        break;
case 'P':
        printf("\n");
        hash_show();
        break;
case 'Q':
        printf("\n");
        exit(1);
default:
        break;                  }}}
```

hash_dictionary.c – build_dictionary()

```
void build_dictionary(char *fname)
{
        char    key[100], data[200];
        FILE    *ifp;

        if((ifp = fopen(fname, "r")) == NULL) {
                printf("No such file ! \n");
                exit(1);
        }

        // 파일에서 (key data)를 읽어 해시테이블에 삽입
        while(fscanf(ifp, "%s %s", key, data) == 2) {

                // 구현 부분
                // hash_insert() 사용
        }
        fclose(ifp);
}
```