동국대학교
dongguk university

실습: Week 9

Data Structures

Contents

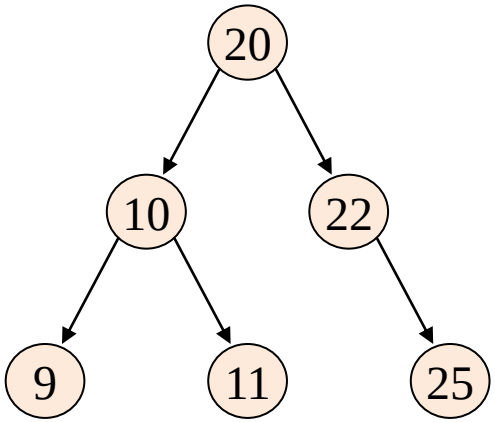
- Tree
  - Binary Search Tree 구현
- 실습
  - 실습 9-1. BST
  - 실습 9-2. BST - dictionary

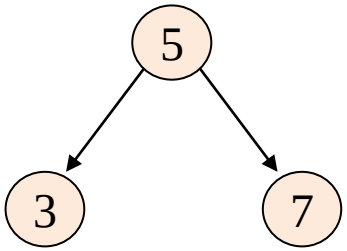
Binary Search Tree

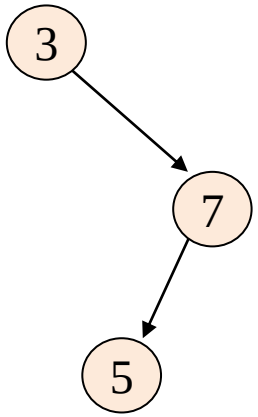
- 리스트 (array or linked list)
  - 원소의 삽입, 삭제, 탐색 → O(n)
- 이진탐색트리
  - 원소의 삽입, 삭제, 탐색 → O(log n) (average)

Binary Search Tree

- 이진탐색트리(BST)
  - 모든 원소는 유일한 키를 가진다.
  - 왼쪽 서브트리에 있는 키들 < 루트의 키
  - 오른쪽 서브트리에 있는 키들 > 루트의 키
  - 왼쪽과 오른쪽 서브트리도 이진탐색트리

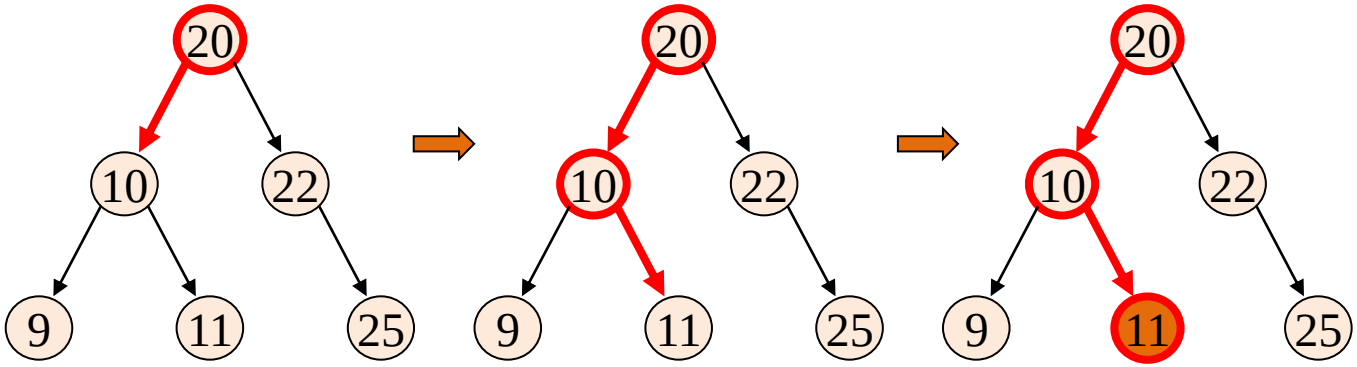






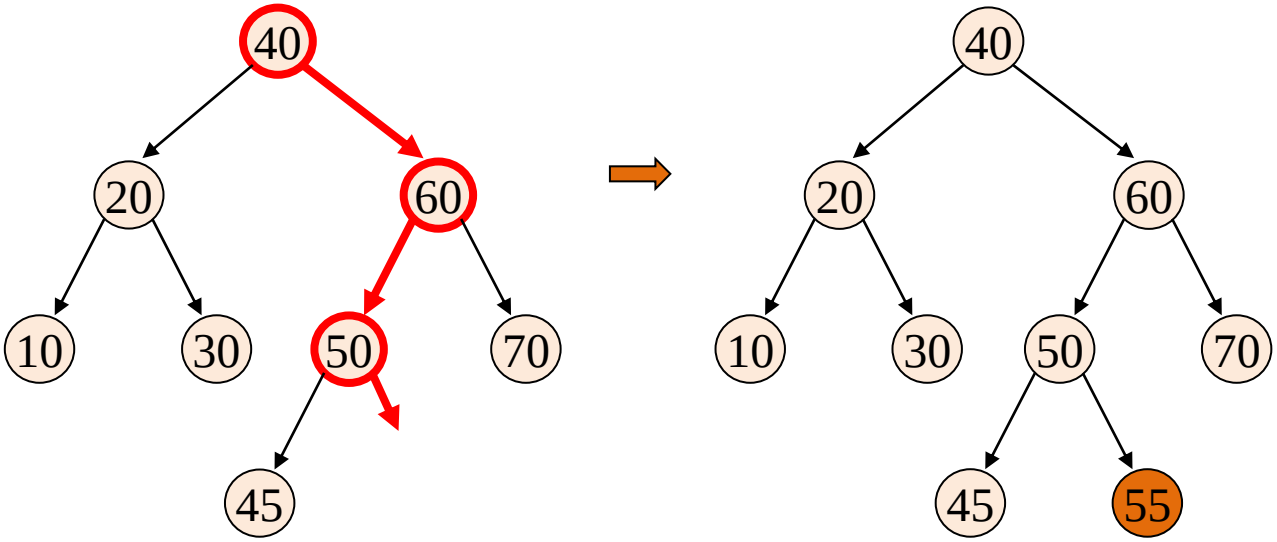
Binary Search Tree

- Searching for 11



Binary Search Tree

- Inserting 55

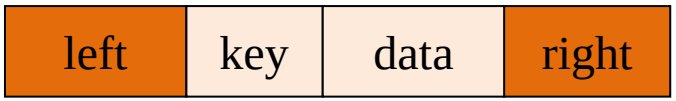


실습 9-1. BST

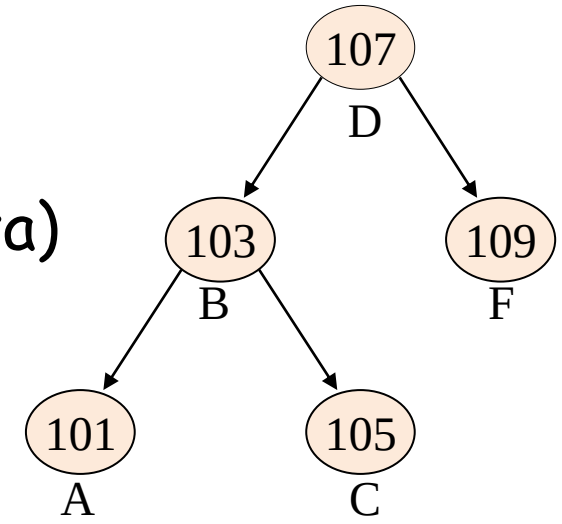
- BST: id를 key로 하는 트리. grade data도 기록
- BST 삽입, 탐색 함수 구현
- 명령어
  - I : Insert data - <id, grade> 삽입 (key: id)
  - S : Search data - id를 탐색, grade를 반환
  - P : Print inorder
  - Q : Quit

자료구조 및 함수

```
typedef struct node *tree_pointer;
typedef struct node{
        int             key;
        char            data;
        tree_pointer    left;
        tree_pointer    right;
};
tree_pointer root;
```

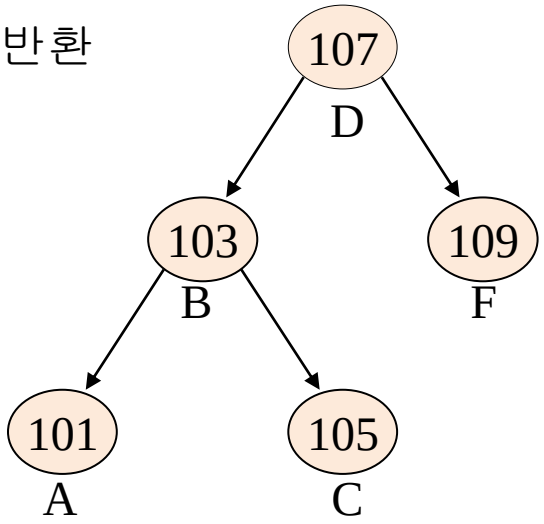


- void bst_insert(int key, char data)
  - key 값에 따라 BST에 (key, data)를 삽입



자료구조 및 함수

- char bst_search(int key);
  - 트리에서 키값이 key인 자료를 검색, data를 반환
  - Ex> bst_search(105) → 'C'



- void bst_show_inorder(tree_pointer ptr);
  - 트리의 자료들을 inorder로 출력
  - Left tree → root → right tree

실습 9-1. 실행 예

```
************* Command ************
I: Insert data, S: Search data
P: Print inorder, Q: Quit
**********************************

Command> i
 id and grade: 107 D

Command> i
 id and grade: 103 B

Command> i
 id and grade: 109 F

Command> i
 id and grade: 105 C

Command> i
 id and grade: 101 A

Command> p
101 A
103 B
105 C
107 D
109 F

Command> s
 id: 105
 Grade of 105: C
```

bst.h

```
#define _CRT_SECURE_NO_WARNINGS

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

// Binary search tree node
typedef struct tree_node* tree_pointer;
typedef struct tree_node {
        int         key;
        char        data;
        tree_pointer        left;
        tree_pointer        right;
} tree_node;
tree_pointer root;   // BST의 root를 가리키는 포인터 (golabl variable)

// 트리에 (key, data) 자료 삽입
void bst_insert(int key, char data);
// 트리에서 키값이 key인 자료를 검색, data를 반환
char bst_search(int key);
// 트리의 자료들을 inorder로 출력
void bst_show_inorder(tree_pointer ptr);
```

bst.c - main()함수

```
#include "bst.h"

void main()
{
        char c, grade;
        int     id;

        printf("************* Command ************\n");
        printf("I: Insert data, S: Search data    \n");
        printf("P: Print inorder, Q: Quit         \n");
        printf("**********************************\n");

        while (1) {
                printf("\nCommand> ");
                c = _getche();
                c = toupper(c);
                switch (c) {
                case 'I':
                        printf("\n id and grade: ");
                        scanf("%d %c", &id, &grade);
                        bst_insert(id, grade);
                        break;
```

bst.c - main()함수

```
case 'S':
                        printf("\n id: ");
                        scanf("%d", &id);
                        grade = bst_search(id);
                        if (grade) printf(" Grade of %d: %c \n", id, grade);
                        else printf(" No such id ! \n");
                        break;
                case 'P':
                        printf("\n");
                        bst_show_inorder(root);
                        break;
                case 'Q':
                        printf("\n");
                        exit(1);
                default: break;
                }
        }
}
```

실습 9-2. BST 사전 구현

- BST: 영어단어를 key로 하는 트리. 국어단어도 기록
- 명령어
  - R : Read data – dic.txt 파일을 읽어 영어단어를 key로 BST를 구성
  - S : Search data – 영어단어를 탐색, 국어단어를 반환
  - P : Print inorder
  - Q : Quit

```
one 하나
two 둘
three 셋
...
```

dic.txt

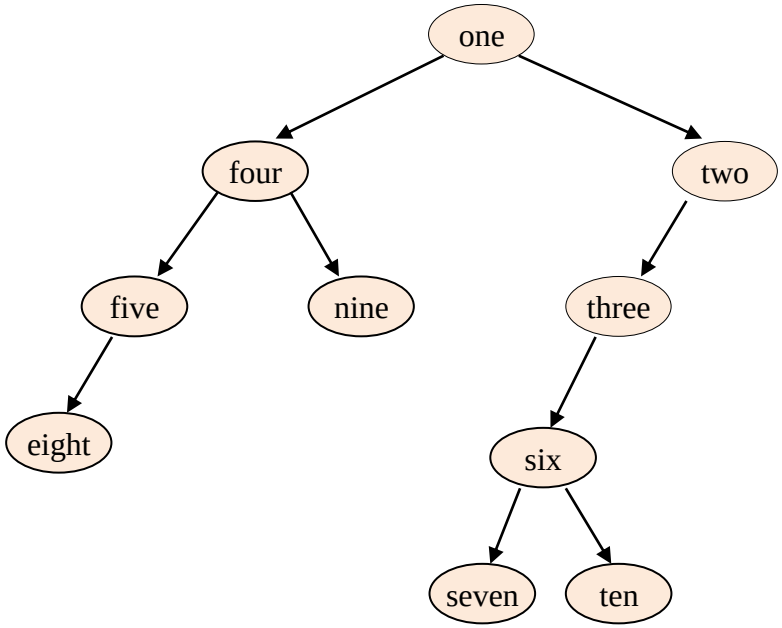
자료구조 및 함수

```
typedef struct node *tree_pointer;
typedef struct node{
        char            w1[100];
        char            w2[100];
        tree_pointer    left;
        tree_pointer    right;
};
tree_pointer root;
```

- int build_dictionary(char *fname);
  - 파일에서 단어들을 읽어 bst_insert를 이용, 이진탐색트리 구성
- void bst_show_inorder(tree_pointer ptr);
  - 트리의 단어들을 inorder로 출력

자료구조 및 함수

- void bst_insert(char *w1, char *w2);
  - 트리에 (w1, w2) 자료 삽입 (key: w1)
- char * bst_search(char *w1);
  - 트리에서 키값이 w1인 자료를 검색, w2를 반환



실습 9-2. 실행 예

```
************* Command ************
R: Read data, S: Search data
P: Print inorder, Q: Quit
**********************************

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

Command> s
 Word: one
 Meaning: 하나

Command> s
 Word: two
 Meaning: 둘

Command> s
 Word: three
 Meaning: 셋

Command> s
 Word: four
 Meaning: 넷

Command> s
 Word: five
 Meaning: 다섯
```

```
Command> s
 Word: six
 Meaning: 여섯

Command> s
 Word: seven
 Meaning: 일곱

Command> s
 Word: eight
 Meaning: 여덟

Command> s
 Word: nine
 Meaning: 아홉

Command> s
 Word: ten
 Meaning: 열

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

Command>
```

bst_dictionary.h

```
#define _CRT_SECURE_NO_WARNINGS

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


// Binary search tree node
typedef struct tree_node* tree_pointer;
typedef struct tree_node {
            char        w1[100];
            char        w2[100];
            tree_pointer          left;
            tree_pointer          right;
} tree_node;
tree_pointer root;   // BST의 root를 가리키는 포인터 (전역변수)

// 파일에서 단어들을 읽어 이진탐색트리 구성
int build_dictionary(char* fname);
// 트리에 (w1, w2) 자료 삽입
void bst_insert(char* w1, char* w2);
// 트리에서 키값이 w1인 자료를 검색, w2를 반환
char* bst_search(char* w1);
// 트리의 단어들을 inorder로 출력
void bst_show_inorder(tree_pointer ptr);
```

bst_dictionary.c - main()함수

```
#include "bst_dictionary.h"
void main()
{
    char  c, fname[20];
    char  w1[100], * w2;
    int    wcount;

    printf("************* Command ************\n");
    printf("R: Read data, S: Search data    \n");
    printf("P: Print inorder, 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);
                wcount = build_dictionary(fname);
                printf(" Total number of words: %d \n", wcount);
                break;
```

bst_dictionary.c - main()함수

```
                case 'S':
                        printf("\n Word: ");
                        scanf("%s", w1);
                        w2 = bst_search(w1);
                        if (w2) printf(" Meaning: %s \n", w2);
                        else printf(" No such word ! \n");
                        break;
                case 'P':
                        printf("\n");
                        bst_show_inorder(root);
                        break;
                case 'Q':
                        printf("\n");
                        exit(1);
                default: break;
                }
        }
}
```