자료 구조
연습 #07
트리

4분반 마감 시간: 5월 8일 오후 11시 59분
5분반 마감 시간: 5월 9일 오후 11시 59분

2019년 1학기

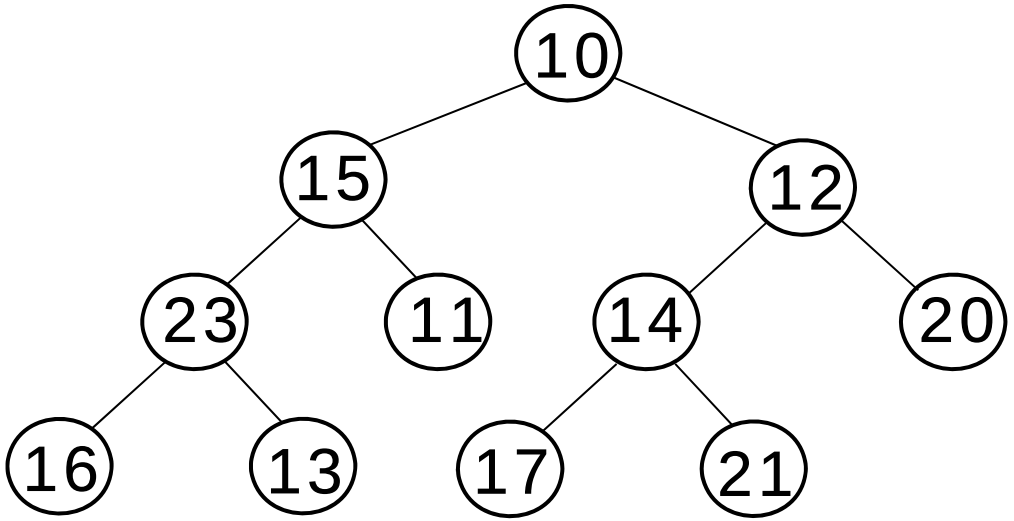
컴퓨터과학과
민경하

내용	**easy**	**moderate**	**hard**
(3) 배열	배열 생성/삽입/삭제	다항식	희소 행렬 합, 곱
(4) 연결리스트	연결리스트 생성	Set/Map 연결리스트 삽입/삭제	
(5) 스택	스택 구현 + 팔린드롬	괄호 매칭	미로찾기 후위 표기
(5) 큐	큐 구현 원형 큐 요제푸스 문제		스케쥴러
(6) 정렬	버블 선택 삽입	합병 쾌속	
(7) 트리	트리 생성 이진 트리 순회	트리 깊이 트리 넓이	동적 트리 조선왕조
(8) 탐색 트리	BST 생성/탐색	BST 제거 Set/Map	AVL tree 2-3 tree Red-black tree B+ tree
(9)우선순위 큐		Heap 생성/제거	
(10) 탐색	이진 탐색 보간 탐색	정적 해쉬	동적 해쉬
(11-13) 그래프	Adjacency matrix Adjacency list DFS BFS	Dijkstra ($O(n^2)$) Floyd Prim/Kruskal Network	Biconnected Strongly connected Dijkstra (O(n log n))

문제

- 다음 텍스트 파일은 5 개의 부모-자식 관계로 구성된 이진 트리를 나타내며, 각 줄에서는 parent, left child, right child의 관계를 나타낸다.

```
10  15  12
15  23  11
16  9  10
12  14  20
23  16  13
11  30  15
14  17  21
15  31  33
```



문제

- 각 줄을 읽어서 해당되는 노드를 트리에 추가하시오. 이 때에 다음의 규칙을 따르시오.
  - Rule 1. Build root node
    - 첫번째 줄의 첫 번째 숫자는 루트 노드임.
  - Rule 2. Avoid forest & duplicate node
    - 각 줄을 읽어서 첫 번째 숫자가 트리의 leaf node에 존재하지 않으면 이 줄은 무시할 것.
  - Rule 3. Avoid cycle
    - 각 줄을 읽어서 두 번째와 세 번째 숫자가 트리에 존재하면 이 줄을 무시할 것.

문제

- 실행 예)
  - 10 15 12 → Rule 1 적용: root node 생성
  - 15 23 11 → 트리에 추가
  - 16 9 10 → Rule 2 적용: 무시
  - 12 14 20 → 트리에 추가
  - 23 16 13 → 트리에 추가
  - 11 30 15 → Rule 3 적용: 무시
  - 14 17 21 → 트리에 추가
  - 15 31 33 → Rule 2 적용: 무시

문제

- 이 파일을 읽어서 트리를 구성하고 다음 물음에 답하시오.

  (1) 이 트리를 **in-order search**한 결과를 쓰시오.

결과: 16, 23, 13, 15, 11, 10, 17, 14, 21, 12, 20

문제

- 이 파일을 읽어서 트리를 구성하고 다음 물음에 답하시오.

  (2) 이 트리의 height를 계산하시오.

  - 트리의 height는 node의 height의 최대값
  - Node의 height는 다음과 같이 정의한다.
    - root node의 height를 1로 가정
    - child node의 height는 parent node의 height + 1로 계산

결과: 4

문제

- 이 파일을 읽어서 트리를 구성하고 다음 물음에 답하시오.

  (3) 이 트리의 **width**를 계산하시오.

  - 트리의 **width**는 두 **node** 사이의 거리의 최대값
    - **Node**의 거리는 **parent-child** 관계를 **1**로 설정

결과: 6

해결 방법

- 추천 자료구조

```
typedef class node *nptr;

class node {
    int val;
    nptr lchild, rchild;

public:
    void init();
    int insert(int cval, int lval, int rval);
    void inorder_print();
    int height();
    int width();
    int search(int sval);
    int valid(int cval, int lval, int rval);

    int is_leaf_node(int cval);
};
```

해결 방법

- init ()
  - Tree를 초기화하는 함수
  - root node의 val을 -1로, lchild와 rchild를 NULL로 초기화함

해결 방법

- insert (int cal, int lval, int rval)
  - cval을 부모 노드로 하고 lval과 rval을 각각 lchild와 rchild로 하는 노드를 트리에 추가
  - 트리의 leaf node 중에서 그 val의 값이 cval과 같은 값을 갖는 node를 찾을 것
  - 재귀 호출로 구현할 것

해결 방법

- inorder_print ()
  - Inorder 순으로 트리를 방문하면서 **val**값을 출력

해결 방법

- height ()
  - Tree의 height를 계산하는 함수
  - 재귀 호출로 구현할 것
  - 한 node의 height는 max (lchild의 height, rchild의 height) + 1로 계산됨

해결 방법

- width ()
  - Tree의 width를 계산하는 함수
  - 재귀 호출로 구현할 것
  - 한 tree의 width는 max (lchild의 height + rchild의 height, lchild의 width, rchild의 width로 계산됨)

해결 방법

- search (int sval)
  - sval과 같은 값을 갖는 val을 저장하는 노드가 있는지 찾는 함수
  - 있으면 1, 없으면 0을 리턴

해결 방법

- valid (int cval, int lval, int rval)
  - cval을 parent node의 값으로 하고 lval과 rval을 child node의 값으로 하는 노드를 삽입할 때 Rule 2와 3을 위배하지 않는지 점검하는 함수

해결 방법

- is_leaf_node (int cval)
  - cval과 같은 값을 val로 저장하는 node가 leaf node인지 점검하는 함수
  - Leaf node이면 1을, 그렇지 않으면 0을 리턴

해결 방법

- 추천 main 함수

```
int main(int argc, char *argv[])
{
    int i;
    int cval, lval, rval;
    char str[128];
    FILE *fp = fopen("tree2.txt", "r+t");

    node *root = (node *) malloc ( sizeof(node));

    root->init();

    while (fgets(str, 128, fp) != NULL) {
        sscanf(str, "%d %d %d", &cval, &lval, &rval);
        if (!root->valid(cval, lval, rval)) {
            printf("Invalid: %d, %d, %d₩n", cval, lval, rval);
            continue;
        }
        root->insert(cval, lval, rval);
    }
    fclose(fp);

    root->inorder_print();
    printf("₩n");

    printf("Height: %d₩n", root->height());
    printf("Width: %d₩n", root->width());

    system("Pause");
    return 0;
}
```