[HW#6] 자료구조론 실습

Binary Search Tree

2014037901 컴퓨터공학
나 윤 환

```
class BinaryTree{

    public Node root;

    private class Node{
        Node parent;
        Node lchild;
        Node rchild;
        int key;

        Node(int key){
            this.parent = null;
            this.lchild = null;
            this.rchild = null;
            this.key = key;
        }
    }
```

```
    public BinaryTree(){
        this.root = null;
    }
```

```java
public void insert(int key){
    // root가 비어있을 경우에는 root에 Node를 새로 생성한다.
    if(root == null) root = new Node(key);
    else{
        // root부터 탐색을 시작한다.
        Node curNode = root;

        while(curNode != null){
            // inputKey가 curNode의 key보다 작을 경우
            if(curNode.key > key){
                // curNode의 왼쪽 자식이 비어있다면 새로 생성
                if(curNode.lchild == null){
                    curNode.lchild = new Node(key);
                    curNode.lchild.parent = curNode;
                    return;
                }
                // 왼쪽 자식이 존재한다면 왼쪽노드로 이동한다.
                else{
                    curNode = curNode.lchild;
                    continue;
                }
            }
            // inputKey가 curNode의 key보다 클 경우
            else{
                // curNode의 오른쪽 자식이 비어있다면 새로 생성
                if(curNode.rchild == null){
                    curNode.rchild = new Node(key);
                    curNode.rchild.parent = curNode;
                    return;
                }
                // 오른쪽 자식이 존재한다면 오른쪽노드로 이동
                else{
                    curNode = curNode.rchild;
                    continue;
                }
            }
        }
    }
}
```

```java
public void delete(int key){
    Node selectNode = search(key);
    Node parent = root;

    if(selectNode != root){
        while(!(parent.lchild == selectNode || parent.rchild == selectNode)){
            if(parent.key > key) parent = parent.lchild;
            else parent = parent.rchild;
        }
    }

    Node curNode = null;

    // selectNode에 자식노드가 없을 때 즉 리프노드일 경우
    if(selectNode.lchild == null && selectNode.rchild == null){
        selectNode = null;
        return;
    }


    // selectNode가 노드를 모두 가지고 있을 경우
    if(selectNode.lchild != null && selectNode.rchild != null){
        Node curParent = selectNode;
        curNode = selectNode.lchild;
        while(curNode.rchild != null){
            curParent = curNode;
            curNode = curNode.rchild;
        }

        curParent.rchild = curNode.lchild;
    }

    // selectNode가 왼쪽노드는 가지고 있고, 오른쪽노드는 없는 경우
    else if(selectNode.lchild != null && selectNode.rchild == null){
        Node curParent = selectNode;
        curNode = selectNode.lchild;
        while(curNode.lchild != null){
            curParent = curNode;
            curNode = curNode.lchild;

        }
        curParent.lchild = curNode.rchild;

    }

    // selectNode가 오른쪽노드는 가지고 있고, 왼쪽노드는 없는 경우
    else if(selectNode.lchild == null && selectNode.rchild != null){
        Node curParent = selectNode;
        curNode = selectNode.rchild;
        while(curNode.rchild != null){
            curParent = curNode;
            curNode = curNode.rchild;
        }
        curParent.rchild = curNode.lchild;

    }

    // 만약에 seledtNode가 root일 경우
    if(selectNode == root){
        root = curNode;
    }
    else if(selectNode != root){
        if(parent.lchild == selectNode) parent.lchild = curNode;
        else if(parent.rchild == selectNode) parent.rchild = curNode;
    }


    curNode.lchild = selectNode.lchild;
    curNode.rchild = selectNode.rchild;

}
```

```java
public Node search(int key){
    Node curNode = root;
    while(true){
        if(curNode == null) return curNode;

        // inputKey가 curNode의 key보다 작을 경우에는, 왼쪽노드로 이동
        if(curNode.key > key){
            curNode = curNode.lchild;
            continue;
        }
        // inputKey와 curNode의 key값이 같을 경우 return curNode;
        else if(curNode.key == key) return curNode;
        // inputKey가 curNode의 key값보다 클 경우에는, 오른쪽노드로 이동
        else{
            curNode = curNode.rchild;
            continue;
        }
    }
}
```

```java
public void print_preorder(Node node){
    System.out.print(node.key + " ");
    if(node.lchild != null) print_preorder(node.lchild);
    if(node.rchild != null) print_preorder(node.rchild);
}

public void print_inorder(Node node){
    if(node.lchild != null) print_inorder(node.lchild);
    System.out.print(node.key + " ");
    if(node.rchild != null) print_inorder(node.rchild);
}

public void print_postorder(Node node){
    if(node.lchild != null) print_postorder(node.lchild);
    if(node.rchild != null) print_postorder(node.rchild);
    System.out.print(node.key + " ");
}
```

```java
class Run{
    public static void main(String[] args) {
        BinaryTree tree = new BinaryTree();
        int arr[] = {5, 3, 2, 1, 4, 8, 6, 7, 10, 9};
        for(int item : arr){
            tree.insert(item);
        }


        System.out.println(tree.search(7) == null? "7없음" : "7있음" );

        System.out.println("Preorder");
        tree.print_preorder(tree.root);
        System.out.println("\nInorder");
        tree.print_inorder(tree.root);
        System.out.println("\nPostorder");
        tree.print_postorder(tree.root);
        System.out.println();
        tree.delete(8);
        tree.delete(5);
        tree.delete(2);

        System.out.println(tree.search(8) == null? "8없음" : "8있음" );

        System.out.println("Preorder");
        tree.print_preorder(tree.root);
        System.out.println();


    }
}
```

```
ACC8722F:src nayunhwan$ java Run
7있음
Preorder
5 3 2 1 4 8 6 7 10 9
Inorder
1 2 3 4 5 6 7 8 9 10
Postorder
1 2 4 3 7 6 9 10 8 5
8없음
Preorder
4 3 1 7 6 10 9
```