

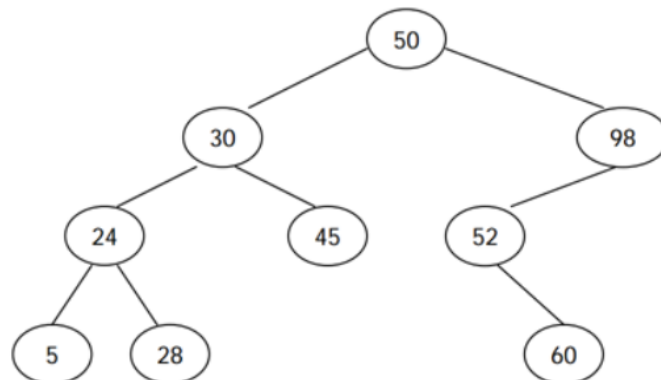


# 5639 - 이진 검색 트리

## 문제

이진 검색 트리는 다음과 같은 세 가지 조건을 만족하는 이진 트리이다.

- 노드의 왼쪽 서브트리에 있는 모든 노드의 키는 노드의 키보다 작다.
- 노드의 오른쪽 서브트리에 있는 모든 노드의 키는 노드의 키보다 크다.
- 왼쪽, 오른쪽 서브트리도 이진 검색 트리이다.



전위 순회 (루트-왼쪽-오른쪽)은 루트를 방문하고, 왼쪽 서브트리, 오른쪽 서브 트리를 순서대로 방문하면서 노드의 키를 출력한다. 후위 순회 (왼쪽-오른쪽-루트)는 왼쪽 서브트리, 오른쪽 서브트리, 루트 노드 순서대로 키를 출력한다. 예를 들어, 위의 이진 검색 트리의 전위 순회 결과는 50 30 24 5 28 45 98 52 60 이고, 후위 순회 결과는 5 28 24 45 30 60 52 98 50 이다.

이진 검색 트리를 전위 순회한 결과가 주어졌을 때, 이 트리를 후위 순회한 결과를 구하는 프로그램을 작성하시오.

## 입력

트리를 전위 순회한 결과가 주어진다. 노드에 들어있는 키의 값은  $10^6$ 보다 작은 양의 정수이다. 모든 값은 한 줄에 하나씩 주어지며, 노드의 수는 10,000개 이하이다. 같은 키를 가지는 노드는 없다.

## 출력

입력으로 주어진 이진 검색 트리를 후위 순회한 결과를 한 줄에 하나씩 출력한다.

### 예제 입력 1 복사

```
50
30
24
5
28
45
98
52
60
```

### 예제 출력 1 복사

```
5
28
24
45
30
60
52
98
50
```

## 풀이과정

주어진 입력값을 바탕으로 이진 검색 트리를 구현하고 후위순회로 출력하는 비교적 간단한 문제였다. 이진 검색 트리를 활용하기 앞서 기본적인 내용을 다루고 넘어가고 싶어 이 문제를 골랐다.

```

static class Node {
    int num;
    Node left;
    Node right;

    Node(int num){
        this.num = num;
    }

    void setLeft(Node leftNode){
        this.left = leftNode;
    }

    void setRight(Node rightNode){
        this.right = rightNode;
    }
}

```

배열로 이진트리를 정렬해볼까 하다가 귀찮아서 **따로 클래스를 만들어서 이진 검색 트리를 구현하는 편이 깔끔할 것 같아** 간단하게 **Node 클래스**를 구현해보았다.

```

Node root;

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
root = new Node(Integer.parseInt(br.readLine()));

String n;

while(true){
    n = br.readLine();
    if(n == null) break;
    Node nextNode = new Node(Integer.parseInt(n));

    // 배열에 노드 배치
    putNode(nextNode, root);
}

postOrder(root);

```

전위 순회는 특성상 루트를 제일 먼저 돌기 때문에 첫번째로 나오는 값을 루트로 만들어주고 시작했다. 그리고 그 다음부터 입력되는 값들은 putNode() 메서드를 통해 이진 검색 트리의 알맞은 곳에 배치해주었다.

다음은 putNode() 코드이다.

```

static void putNode(Node newNode, Node compNode) {

    if(newNode.num < compNode.num){
        if(compNode.left == null){
            compNode.setLeft(newNode);
        }
        else putNode(newNode, compNode.left);
    }
    else{
        if(compNode.right == null){
            compNode.setRight(newNode);
        }
        else putNode(newNode, compNode.right);
    }
}

```

위 함수는 **재귀**로 구현하였다. 대략적인 구조를 설명하자면 현재 새로 삽입하려는 노드 (newNode)의 값이 현재 보고 있는 노드(compNode)의 값보다 작을 경우 왼쪽 서브트리로, 클 경우 오른쪽 서브트리로 가도록 재귀로 넘기는 함수이다. 단 compNode의 왼쪽 자식 노드가 존재하지 않을 경우 newNode를 왼쪽 노드로 설정하고, compNode의 오른쪽 자식 노드가 존재하지 않을 경우 newNode를 오른쪽 노드로 설정해준다.

여기까지는 들어온 입력값들을 바탕으로 이진 검색 트리를 만들었다.

이제 후위 순회를 하며 출력만 해주면 끝이다.

후위 순회 함수도 구현이 간단했다.


```

static void postOrder(Node node) {
    if(node.left != null) postOrder(node.left);
    if(node.right != null) postOrder(node.right);
    System.out.println(node.num);
}

```

왼쪽 → 오른쪽 순으로 검사해주고 돌아온 뒤 현재 자기 노드의 값을 출력해주면 완성 😊

이렇게 후위 순회로 출력해주는 것까지 끝이 났다 🙌🙌🙌

제출 번호	아이디	문제	결과	메모리	시간	언어	코드 길이
35280400	lyuashley	 5639	맞았습니다!!	19356 KB	672 ms	Java 11 / 수정	1565 B