e-learning

2020 학년도 1 학기

컴퓨터 정보과

자료구조(Data Structures)

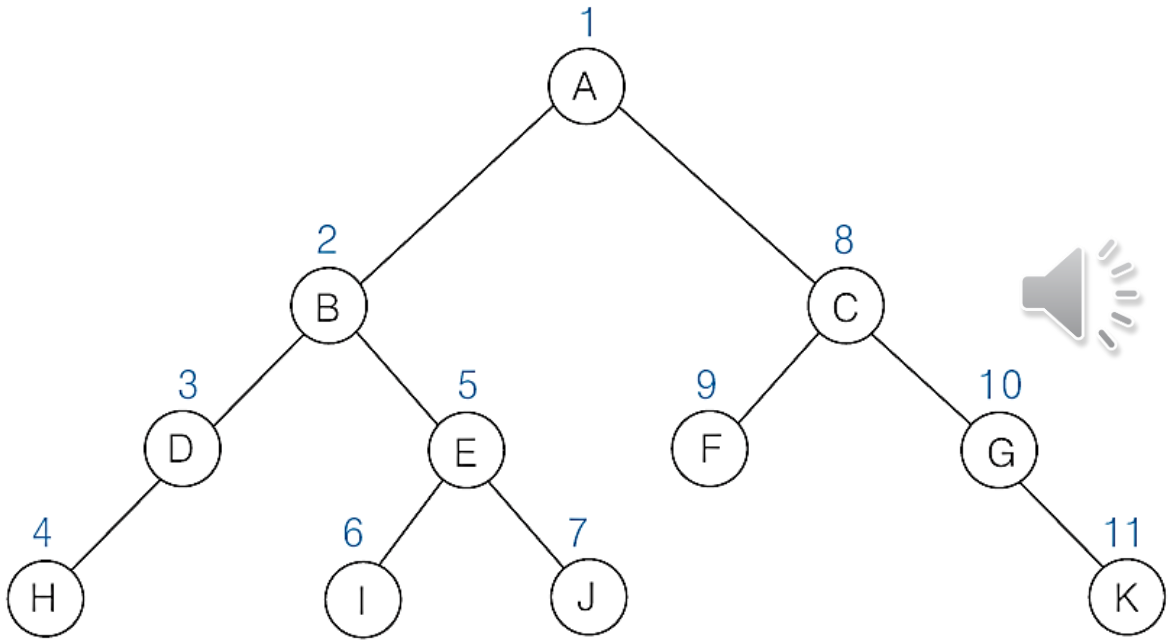
담당교수 : 김주현

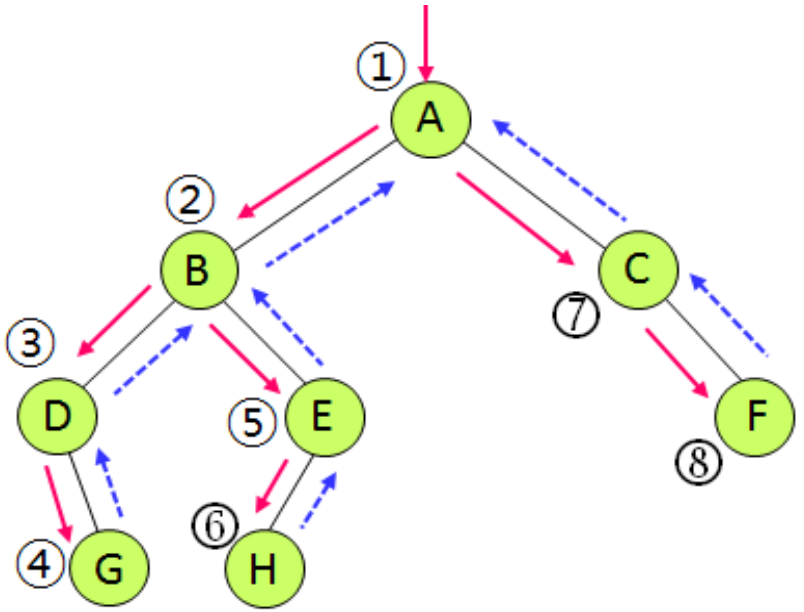
제 5 주차 / 제 3 차시

DIT 동의과학대학교
DONG-EUI INSTITUTE OF TECHNOLOGY

- 전위 순회(preorder traversal)
  - ✓ 수행 방법
    - ① 현재 노드 n을 방문하여 처리한다. : D
    - ② 현재 노드 n의 왼쪽 서브트리로 이동한다. : L
    - ③ 현재 노드 n의 오른쪽 서브트리로 이동한다. : R

전위 순회의 예



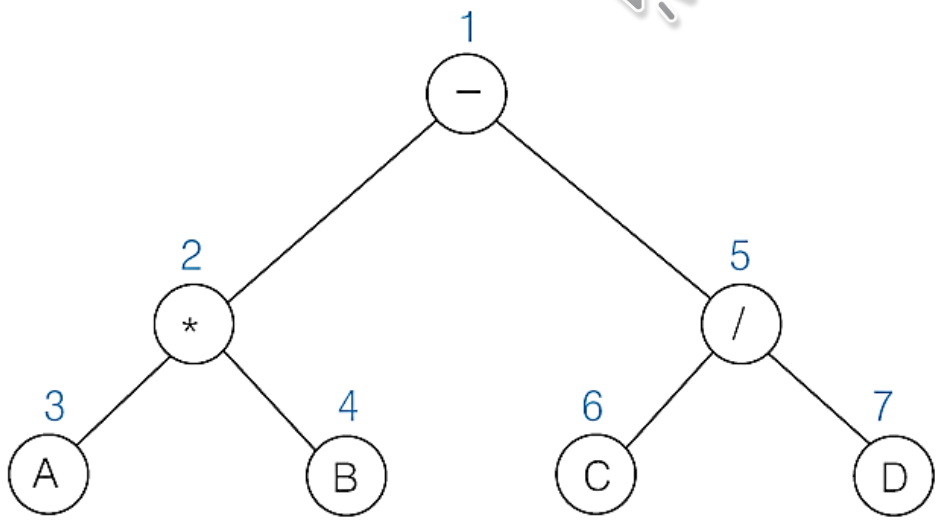


- # 수식 이진 트리에 대한 전위 순회

✓ 수식을 이진 트리로 구성한 수식 이진 트리를 전위 순회하면, 수식에 대한 전위 표기식을 구할 수 있다.

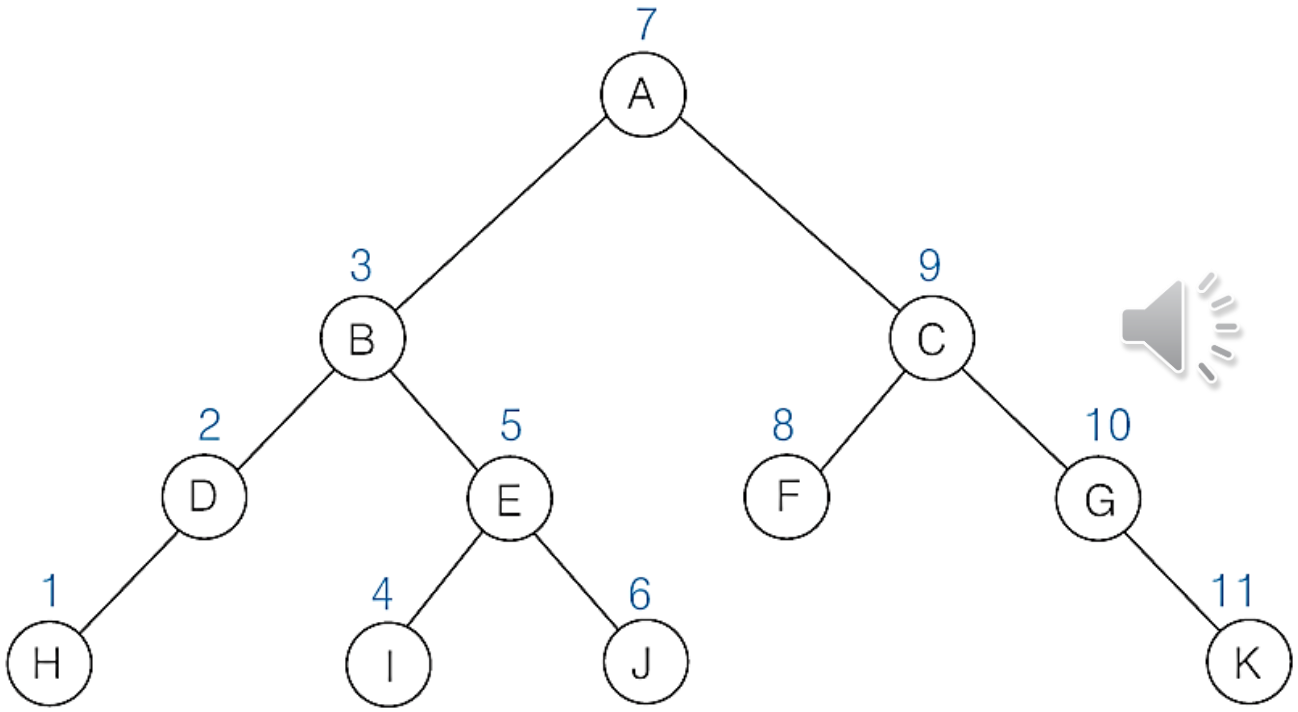
✓ 이진 트리의 전위 순회 경로 예

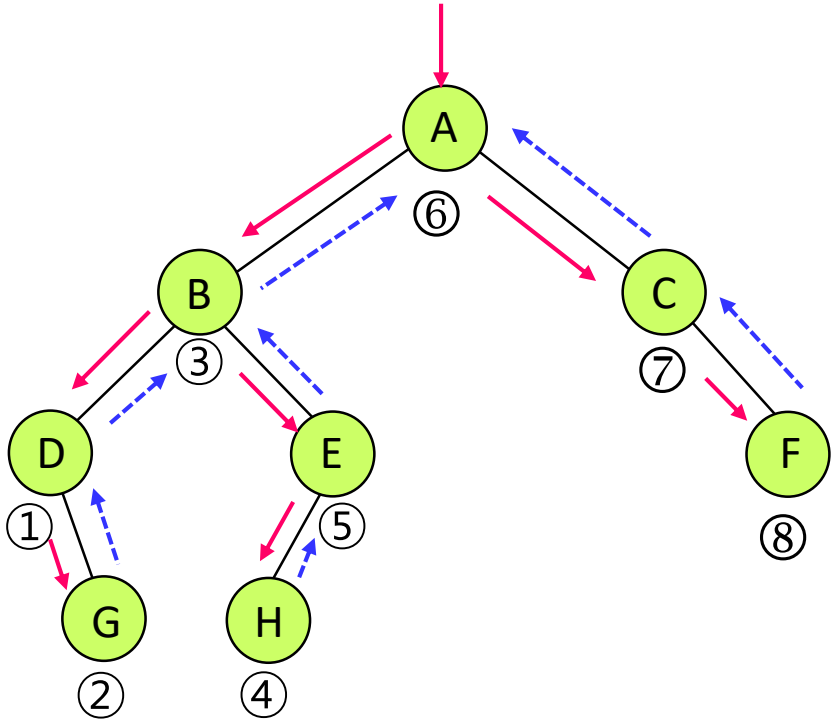
(A*B-C/D) ➔ -*AB/CD



- 중위 순회(inorder traversal)
  - ✓ 수행 방법
    - ① 현재 노드 n의 왼쪽 서브트리로 이동한다. : L
    - ② 현재 노드 n을 방문하여 처리한다. : D
    - ③ 현재 노드 n의 오른쪽 서브트리로 이동한다. : R

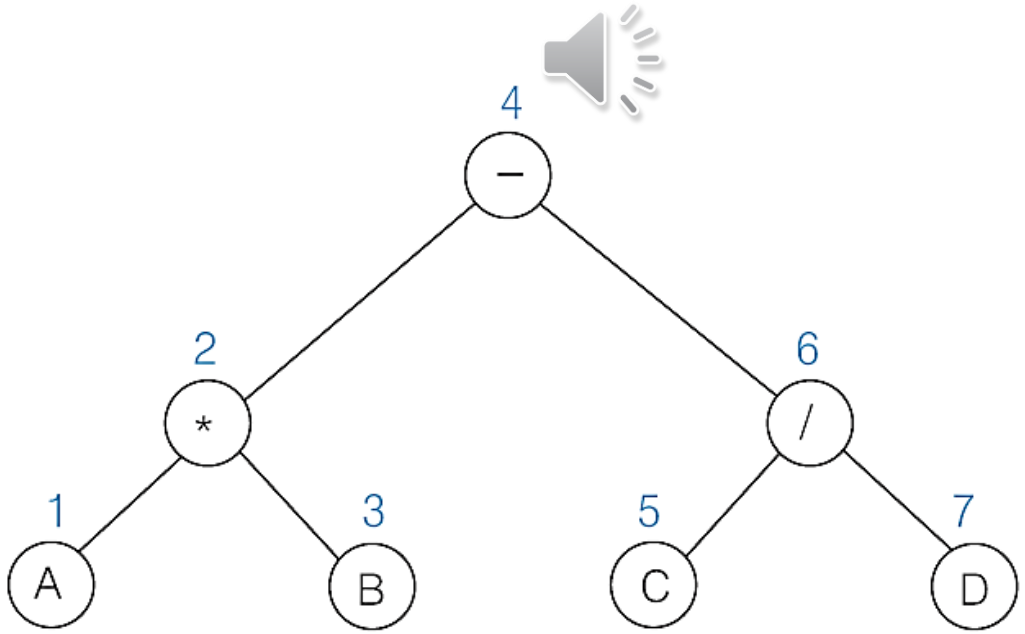
중위 순회의 예





- **수식 이진 트리에 대한 중위 순회**
  - ✓ 수식 이진 트리를 중위 순회하면, 수식에 대한 중위 표기식을 구할 수 있다.
  - ✓ 이진 트리의 중위 순회 경로 예

    (A*B-C/D) ➔ A*B-C/D



- 후위 순회(postorder traversal)

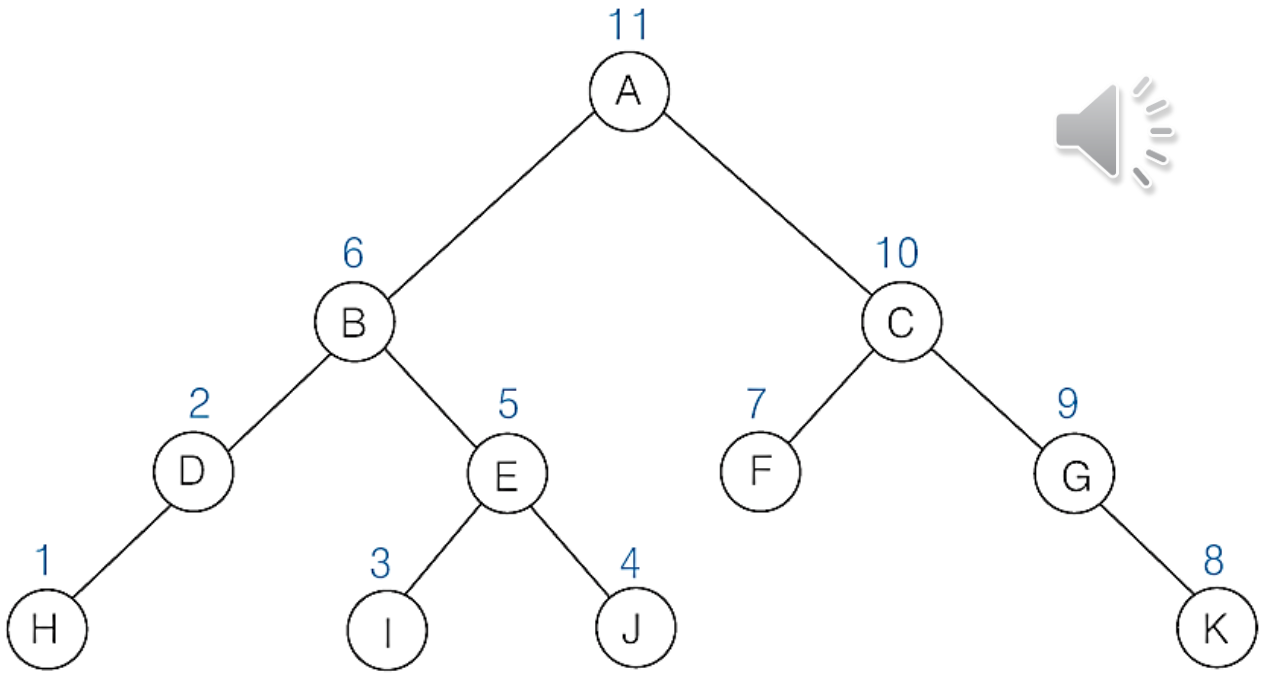
✓ 수행 방법

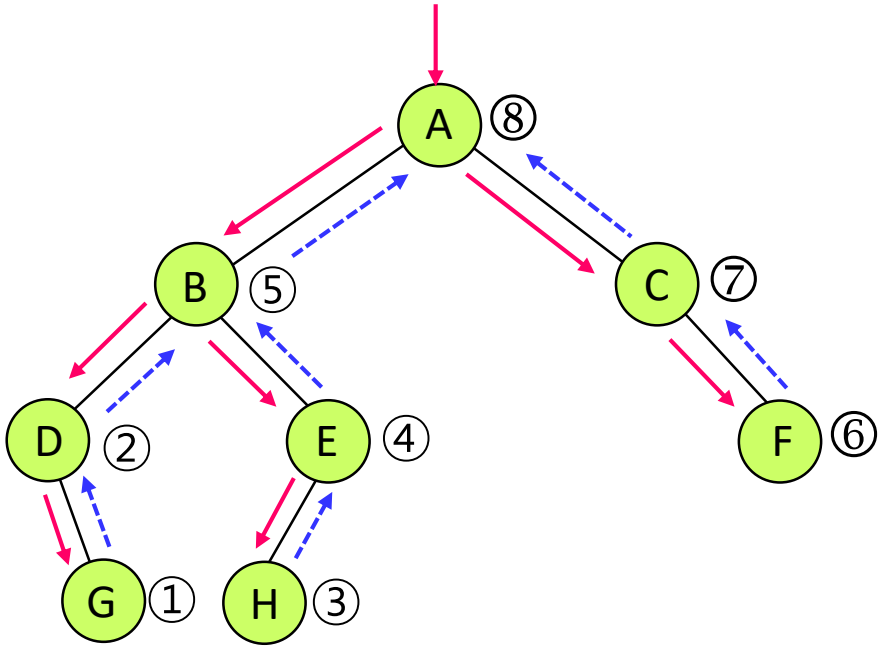
① 현재 노드 n의 왼쪽 서브트리로 이동한다. : L

② 현재 노드 n의 오른쪽 서브트리로 이동한다. : R

③ 현재 노드 n을 방문하여 처리한다. : D

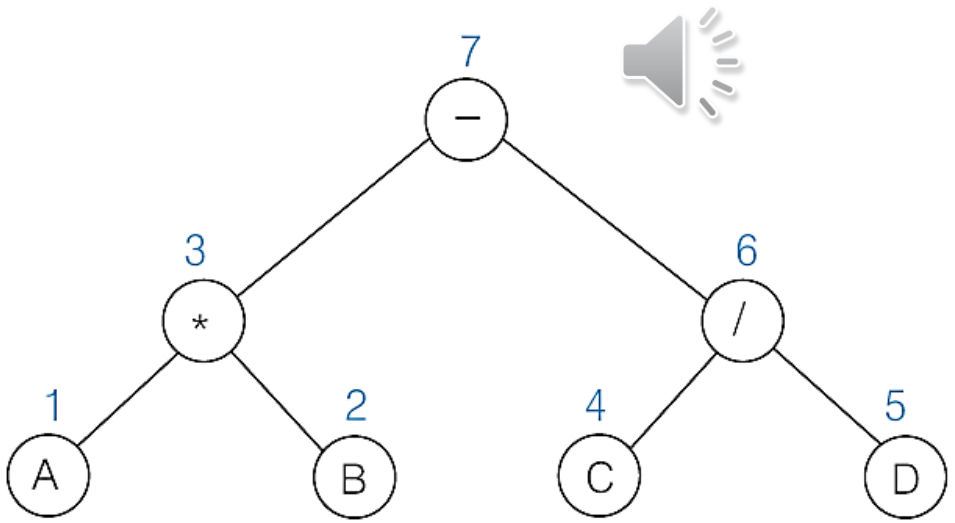
후위 순회의 예





- **수식 이진 트리에 대한 후위 순회**
  - ✓ 수식에 대한 후위 표기식을 구할 수 있음
  - ✓ 이진 트리의 후위 순회 경로 예

    (A*B-C/D) ➔ AB*CD/-



TreeEx_1.java

```
class TreeNode{
        Object data;
        TreeNode left;
        TreeNode right;
}
```

```
class LinkedTree{
        private TreeNode root;

        public TreeNode makeBT(TreeNode bt1, Object data, TreeNode bt2){
                TreeNode root = new TreeNode();
                root.data = data;
                root.left = bt1;
                root.right = bt2;
                return root;
        }
        public void preorder(TreeNode root){
                if(root != null){
                        System.out.printf("%c", root.data);
                        preorder(root.left);
                        preorder(root.right);
                }
        }
```

```
        public void inorder(TreeNode root){
                if(root != null){
                        inorder(root.left);
                        System.out.printf("%c", root.data);

                        inorder(root.right);
                }
        }
        public void postorder(TreeNode root){
                if(root != null){
                        postorder(root.left);

                        postorder(root.right);
                        System.out.printf("%c", root.data);
                }
        }
}
```

```
class TreeEx_1{
        public static void main(String args[]){
                LinkedTree T = new LinkedTree();
                TreeNode n7 = T.makeBT(null, 'D', null);
                TreeNode n6 = T.makeBT(null, 'C', null);
                TreeNode n5 = T.makeBT(null, 'B', null);
                TreeNode n4 = T.makeBT(null, 'A', null);
                TreeNode n3 = T.makeBT(n6, '/', n7);
                TreeNode n2 = T.makeBT(n4, '*', n5);
                TreeNode n1 = T.makeBT(n2, '-', n3);
                System.out.printf("₩n Preorder : ");
                T.preorder(n1);
                System.out.printf("₩n Inorder : ");
                T.inorder(n1);
                System.out.printf("₩n Postorder : ");
                T.postorder(n1);
        }
}
```

BTNode.java

```
public class BTNode<Key extends Comparable<Key>> {
        private  Key       item;
        private  BTNode<Key> left;
        private  BTNode<Key> right;

        public BTNode( Key newItem, BTNode<Key> lt, BTNode<Key> rt ) { // 노드 생성자
                item = newItem; left = lt; right = rt; }

        public Key              getKey( )   { return item; }
        public BTNode<Key> getLeft( )  { return left; }
        public BTNode<Key> getRight( ) { return right;}

        public void setKey(Key  newItem)  { item  = newItem; }
        public void setLeft(BTNode<Key> lt) { left  = lt; }
        public void setRight(BTNode<Key> rt) { right  = rt; }
}
```

BinaryTree.java

```
import java.util.Queue;
import java.util.LinkedList;
public class BinaryTree<Key extends Comparable<Key>>  {
        private BTNode<Integer> root;
        public BinaryTree( ) { root = null; }       // 트리 생성자
        public BTNode<Integer> getRoot( )              { return root; }
        public void setRoot(BTNode<Integer> newRoot) { root = newRoot; }
        public boolean isEmpty( ) { return root == null; }

        public void preorder(BTNode<Integer> n) {     // 전위 순회
                if (n != null) {
                        System.out.print(n.getKey()+" ");  // 노드 n 방문
                        preorder(n.getLeft());  // n의 왼쪽     서브 트리를 순회하기 위해
                        preorder(n.getRight()); // n의 오른쪽 서브 트리를 순회하기 위해
                }
        }
```

```
public void inorder(BTNode<Integer> n){        // 중위 순회
        if (n != null) {
                inorder(n.getLeft());   // n의 왼쪽 서브 트리를 순회하기 위해
                System.out.print(n.getKey()+" ");  // 노드 n 방문
                inorder(n.getRight());  // n의 오른쪽 서브 트리를 순회하기 위해
        }
}

public void postorder(BTNode<Integer> n) {     // 후위 순회
        if (n != null) {
                postorder(n.getLeft());  // n의 왼쪽     서브 트리를 순회하기 위해
                postorder(n.getRight()); // n의 오른쪽 서브 트리를 순회하기 위해
                System.out.print(n.getKey()+" "); // 노드 n 방문
        }
}
```

```
public int size(BTNode<Integer> n)      {  // n를 루트로하는 (서브)트리에 있는 노드 수
        if (n == null)
                return 0;    // null이면 0 리턴
        else
                return (1 + size( n.getLeft() ) + size( n.getRight() ));
}

public int height(BTNode<Integer> n) {  // n를 루트로하는 (서브)트리의 높이
        if (n == null)
                return 0;     // null이면 0 리턴
        else
                return (1 + Math.max(height(n.getLeft()), height(n.getRight())));
}
```

```
public static boolean isEqual(BTNode<Integer> n, BTNode<Integer> m){
// 두 트리의 동일성 검사
        if(n==null || m==null)  // 둘중에 하나라도 null이면
                return n == m;      // 둘다 null이면 true, 아니면 false
        if (n.getKey().compareTo(m.getKey()) != 0) // 둘다 null이 아니면 item 비교
                return false;
        return( isEqual(n.getLeft(), m.getLeft()) &&
                                // item이 같으면 왼쪽/오른쪽 자식으로 재귀 호출
                        isEqual(n.getRight(), m.getRight()) );
}

public BTNode<Integer> copy (BTNode<Integer> n) {
        BTNode<Integer> left, right;
        if (n == null) return null;
        else {
                left  = copy(n.getLeft());
                right = copy(n.getRight());
                return new BTNode<Integer>(n.getKey(),left, right);
        }
}
```

```
        public void levelorder(BTNode<Integer> root) { // 레벨 순회
                Queue<BTNode<Integer>> q = new LinkedList<BTNode<Integer>>();
                // 큐 자료구조 이용
                BTNode<Integer> t;
                q.add(root);  // 루트 노드 큐에 삽입
                while (!q.isEmpty()) {
                        t = q.remove();   //큐에서 가장 앞에 있는 노드 제거
                        System.out.print(t.getKey()+" "); // 제거된 노드 출력(방문)
                        if (t.getLeft() != null)       // 제거된 왼쪽 자식이 null이 아니면
                                q.add(t.getLeft());       // 큐에 왼쪽 자식 삽입
                        if (t.getRight() != null)     // 제거된 오른쪽 자식이 null이 아니면
                                q.add(t.getRight());      // 큐에  오른쪽 자식 삽입
                }
        }
}
```

BinTree.java

```
public class BinTree {

        public static void main(String[] args) {

                BTNode<Integer> n1 = new BTNode<Integer>(100,null,null);
                BTNode<Integer> n2 = new BTNode<Integer>(200,null,null);
                BTNode<Integer> n3 = new BTNode<Integer>(300,null,null);
                BTNode<Integer> n4 = new BTNode<Integer>(400,null,null);
                BTNode<Integer> n5 = new BTNode<Integer>(500,null,null);
                BTNode<Integer> n6 = new BTNode<Integer>(600,null,null);
                BTNode<Integer> n7 = new BTNode<Integer>(700,null,null);
                BTNode<Integer> n8 = new BTNode<Integer>(800,null,null);

                n1.setLeft(n2);    n1.setRight(n3);  // n1의 왼쪽 자식-> n2, n1의 오른쪽 자식-> n3
                n2.setLeft(n4);   n2.setRight(n5);  // n2의 왼쪽 자식-> n4, n2의 오른쪽 자식-> n5
                n3.setLeft(n6);   n3.setRight(n7);  // n3의 왼쪽 자식-> n6, n3의 오른쪽 자식-> n7
                n4.setLeft(n8);
```

```
BinaryTree<Integer> t = new BinaryTree<Integer>();  // 이진 트리 객체 t 생성
t.setRoot(n1);                                      // t의 루트 노드를 n1으로

System.out.print("트리 노드 수  = "+ t.size(t.getRoot())+"₩n트리 높이      = "+
t.height(t.getRoot()));
System.out.printf("₩n전위 순회:  ");
t.preorder(t.getRoot());
System.out.printf("₩n중위 순회:  ");
t.inorder(t.getRoot());
System.out.printf("₩n후위 순회:  ");
t.postorder(t.getRoot());
System.out.printf("₩n레벨 순회:  ");
t.levelorder(t.getRoot());
System.out.println();
```

```
// 두번째 이진 트리를 만들어 isEqual() 테스트하기 위해
BTNode<Integer> n10 = new BTNode<Integer>(100,null,null);
BTNode<Integer> n20 = new BTNode<Integer>(200,null,null);
BTNode<Integer> n30 = new BTNode<Integer>(300,null,null);
BTNode<Integer> n40 = new BTNode<Integer>(400,null,null);
BTNode<Integer> n50 = new BTNode<Integer>(500,null,null);
BTNode<Integer> n60 = new BTNode<Integer>(600,null,null);
BTNode<Integer> n70 = new BTNode<Integer>(700,null,null);
BTNode<Integer> n80 = new BTNode<Integer>(800,null,null);

n10.setLeft(n20); n10.setRight(n30);
n20.setLeft(n40); n20.setRight(n50);
n30.setLeft(n60); n30.setRight(n70);
n40.setLeft(n80);
```

```
                BinaryTree<Integer> t2 = new BinaryTree<Integer>();
                t2.setRoot(n10);

                System.out.printf("동일성 검사:   "+BinaryTree.isEqual(t.getRoot(), t2.getRoot()));
                System.out.println();

                BinaryTree<Integer> t3 = new BinaryTree<Integer>();
                t3.setRoot(t3.copy(t.getRoot()));
                System.out.printf("copy 테스트: "+BinaryTree.isEqual(t.getRoot(), t3.getRoot()));
                System.out.println();

        }
}
```

Report

TreeEx_1.java, BTNode.java, BinaryTree.java, BinTree.java
파일을 구현하고 테스트한 후에 Github에 업로드하세요

Reference

- https://ko.wikipedia.org/wiki/%ED%8A%B8%EB%A6%AC_%EA%B5%AC%EC%A1%B0https://jsieun73.tistory.com/26
- https://coding-factory.tistory.com/231
- https://gmlwjd9405.github.io/2018/08/12/data-structure-tree.html
- 자바로 배우는 쉬운 자료구조, 이지영, 한빛아카데미
- 자바와 함께하는 자료구조의 이해, 양성봉, 생능출판

언제 어디서나 즐공, 열공, 진공하세요.

감사합니다

DIT 동의과학대학교
DONG-EUI INSTITUTE OF TECHNOLOGY

DIT 동의과학대학교
DONG-EUI INSTITUTE OF TECHNOLOGY