자바 보고서

이름 : 정예린

전공 : 컴퓨터공학부 소프트웨어

학번 : 20222872

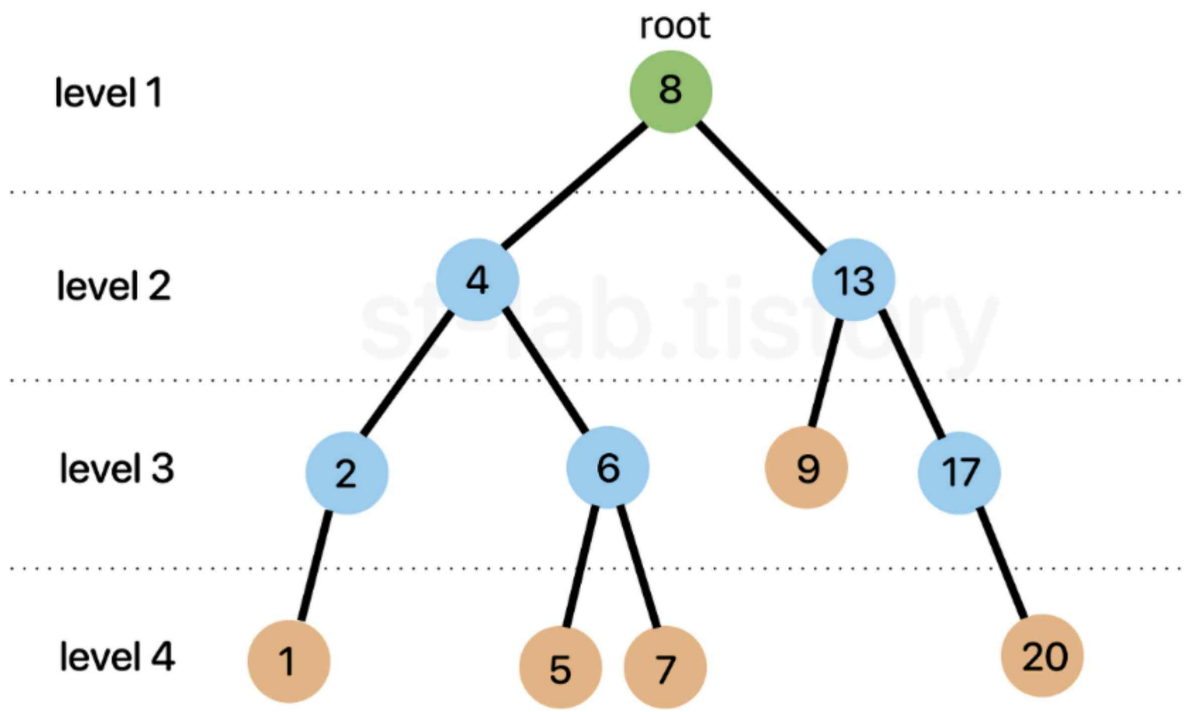
교과 : JAVA프로그래밍

교수님 : 유진호 교수님

A) 이진트리(binary)에 대해서 조사하기

이진트리는 영어로 binary tree로 이분화된 트리 자료구조라는 뜻이다. 각 노드가 최대 두 개의 (0 ~ 2) 자식 노드를 가지는 트리 자료 구조로, 그것을 그린 모습이 나무를 뒤집어 놓은 모습과 비슷하다고 하여 붙인 이름이다.

트리는 데이터를 계층 구조로 저장하기 때문에 파일이나 폴더와 같은 계층 구조를 갖는 곳에 사용되며 보다 효율적인 삽입, 삭제, 검색을 위해서도 사용된다.



여기서 몇 가지 알아야 할 트리 용어들이 있다.

- 트리의 구성요소인 노드

- 가장 상위에 위치한 노드인 루트노드

- 루트노드를 제외한 나머지 로드들을 자식 노드들과 함께 묶은 서브트리

- 1개의 선으로 직접 연결된 부모와 자식 관계

- 부모가 같은 노드인 형제 노드

- 자식 노드가 없는 노드는 단말 노드와 자식 노드가 있는 노드는 비단말 노드

- 노드들을 연결하는 선을 간선

- 노드가 가지고 있는 자식 노드의 수는 차수

- 각 층에 번호를 매기는 것으로 0 혹은 1부터 시작해서 아래로 내려갈수록 커지는 레벨

- 트리의 높이는 해당 트리의 최대 레벨을 의미한다.

이 외에도 다양한 용어들이 더 있다.

이진트리는 형태에 따라 포화 이진트리, 완전 이진트리로 구분할 수 있다.

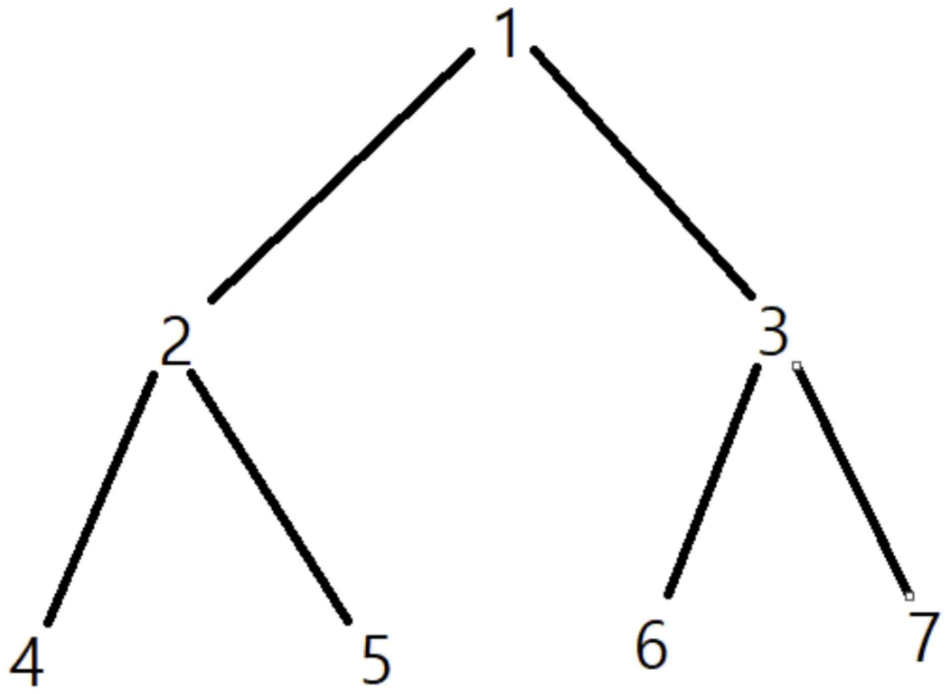
- 포화 이진트리 : 각 레벨에 노드가 꽉 차있는 이진트리로 노드를 더 추가하기 위해서는 레벨을 늘려야 한다.

- 완전 이진트리 : 트리의 높이가 k라면 k-1레벨까지는 노드가 꽉 채워져있고, 마지막 레벨에서는 노드가 꽉 차있지 않아도 왼쪽에서 오른쪽 순서대로 차있는 이진트리

그래서 포화 이진트리 < 완전 이진트리 < 일반 이진트리의 관계를 가지고 있다.

트리 중에서는 탐색에 특화된 트리도 있다. 그것을 이진 탐색 트리라고 하는데 이를 위해서는 , 부모 노드를 기준으로 왼쪽 자식들은 부모 노드보다 값이 작으며, 오른쪽 자식 노드들은 부모 노드보다 값이 크다라는 조건을 붙여야 한다. 그렇게 구성한다면 특정한 값을 탐색하려고 할 때 기준이 되는 노드와 찾으려는 값의 대소 관계를 비교해서 작으면 왼쪽 노드로 가고, 크면 오른쪽 노드로 가는 것을 통해 기준이 되는 노드와 같은 값을 가지고 있는 노드를 쉽게 알아낼 수 있다.

트리 구조에서 각 노드를 한 번씩 방문하는 과정을 트리 순회라고 하는데, 이 트리 순회 방법에는 전위 탐색, 중위 탐색, 후위 탐색이 있다. 이는 노드 방문을 언제 하느냐에 따라 순회 방법이 나뉘어진다. 이 그림을 이용하여 더 쉽게 이야기하자면,



1. 전위 순회 (pre-order)

: 루트 노드를 먼저 순회한 후 왼쪽 하위 -> 오른쪽 하위 순으로 순회하는 방법으로 위 그림에서는 1 - 2 - 4 - 5 - 3 - 6 - 7 순서이다.

2. 중위 순회 (in-order)

: 왼쪽 가장 하위 노드를 먼저 순회한 후 바로 상위 노드 -> 오른쪽 하위 순으로 순회하는 방법으로 위 그림에서는 4 - 2 - 5 - 1 - 6 - 3 - 7 순서이다.

3. 후위 순회 (pre-order)

: 왼쪽 가장 하위 노드를 먼저 순회한 후 오른쪽 하위 노드 -> 바로 상위 노드 순으로 순회하는 방법으로 위 그림에서는 4 - 5 - 2 - 6 - 7 - 3 - 1 순서이다.

B) Class로 트리노드(node) 구현하기

노드 선언

treeNode.java

```
package Tree;

public class treeNode {
    public char data;
    public treeNode leftChild;
    public treeNode rightSibling;
}
```

노드 생성, 연결, 출력

treeMethod.java

```
package Tree;

public class treeMethod {

    public treeNode createNode(char data) {
        treeNode newNode = new treeNode();
        newNode.data = data;
        newNode.leftChild = null;
        newNode.rightSibling = null;

        return newNode;
    }

    public void addChildNode(treeNode parent, treeNode child) {
        if (parent.leftChild == null) {
            parent.leftChild = child;
            return;
        }
        treeNode currentNode = parent.leftChild;
        while (currentNode.rightSibling != null) currentNode =
currentNode.rightSibling;
        currentNode.rightSibling = child;
    }

    public void printTree(treeNode node, int depth) {

        for (int i = 0; i < depth; i++) System.out.print(" ");

        System.out.println(node.data);
```

```
        if (node.leftChild != null) printTree(node.leftChild, depth + 1);
        if (node.rightSibling != null) printTree(node.rightSibling, depth);
    }
}
```

테스트

treeTest.java

```
package Tree;

public class treeTest {

    public static void main(String[] args) {
        treeMethod tm = new treeMethod();
        treeNode a = tm.createNode('a');
        treeNode b = tm.createNode('b');
        treeNode c = tm.createNode('c');
        treeNode d = tm.createNode('d');
        treeNode e = tm.createNode('e');
        tm.addChildNode(a, b);
        tm.addChildNode(a, c);
        tm.addChildNode(b, d);
        tm.addChildNode(b, e);
        tm.printTree(a, 0);
    }
}
```

결과 :

```
a
 b
  d
  e
 c
```

C) 이진트리 삽입 구현하기 (X)
