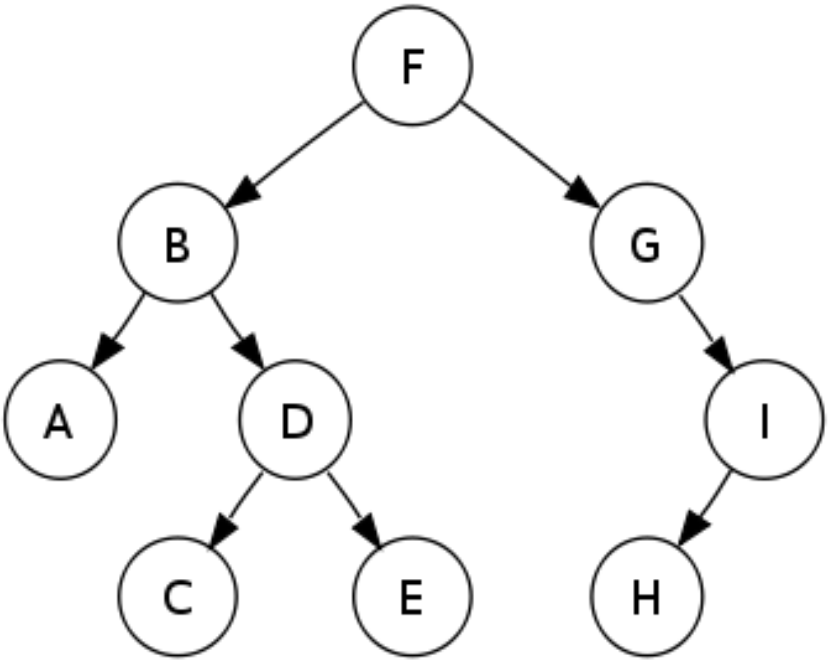
트리

트리(Tree)

- 계층적인 구조를 표현할 때 사용할 수 있는 자료구조
- 비선형 자료구조



트리 관련 용어

- 루트 노드(root node) : 부모가 없는 최상위 노드
- 단말 노드(leaf node) : 자식이 없는 노드
- 크기(size) : 트리에 포함된 모든 노드의 개수
- 깊이(depth) : 루트 노드로부터의 거리
- 높이(height) : 깊이 중 최댓값
- 차수(degree) : 각 노드의 (자식 방향) 간선 개수

- 기본적으로 트리의 크기가 N 일 때, 전체 간선의 개수는 N-1개

트리의 구현

```
function TreeNode(value) {
    this.value = value;
    this.children = [];
}
```

트리의 순회 : 재귀로 순회

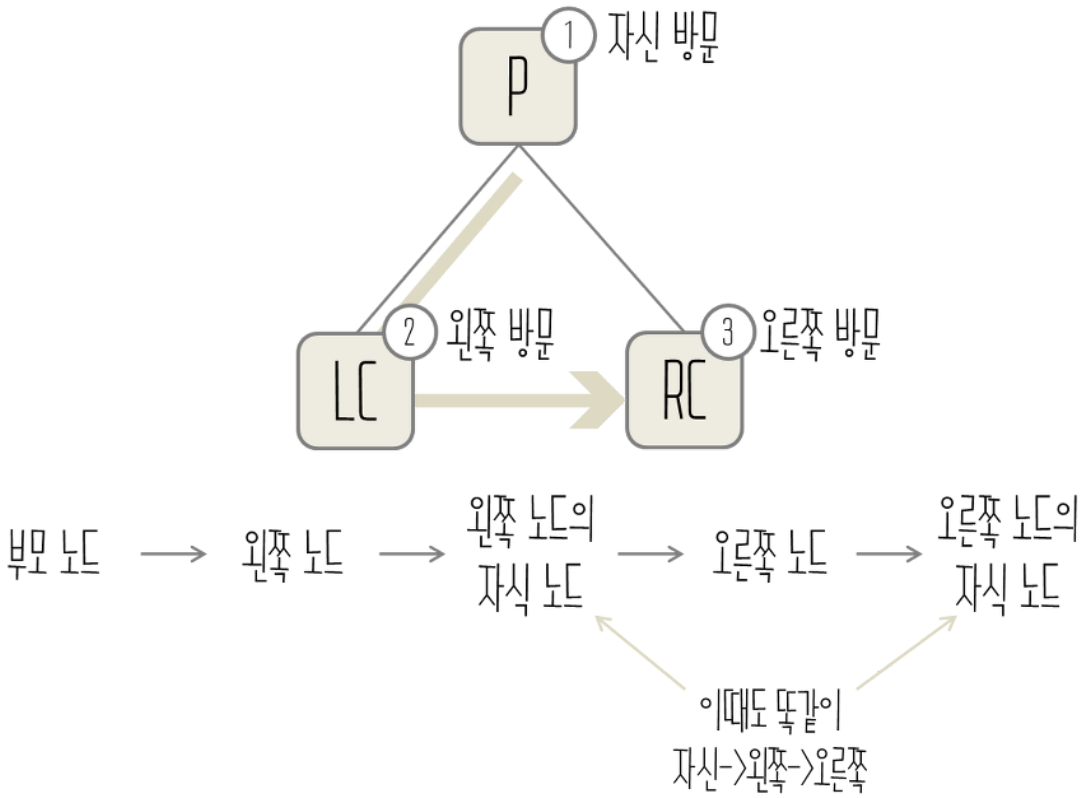
이진 트리

- 각 노드가 최대 두개의 자식을 가지는 트리

```
class Tree {
  constructor(val) {
    this.val = val;
    this.leftNode = null;
    this.rightNode = null;
  }

  setVal(val) {
    this.val = val;
  }

  setLeft(node) {
    this.leftNode = node;
  }

  setRight(node) {
    this.rightNode = node;
  }
}
```

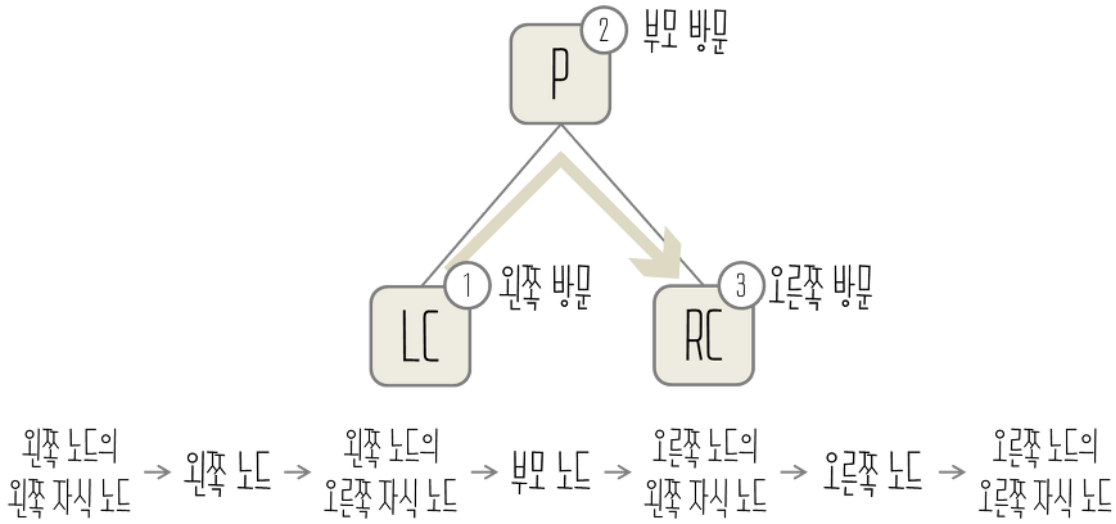
이진 트리의 순회

1. 전위 순회(pre-order traversal)
    - 현재 노드 → 왼쪽 가지 → 오른쪽 가지




```
var recursivePreOrder = function (node) {
  if (!node) {
    return;
  }
  console.log(node.val);
  this.recursivePreOrder(node.leftNode);
  this.recursivePreOrder(node.rightNode);
};
```

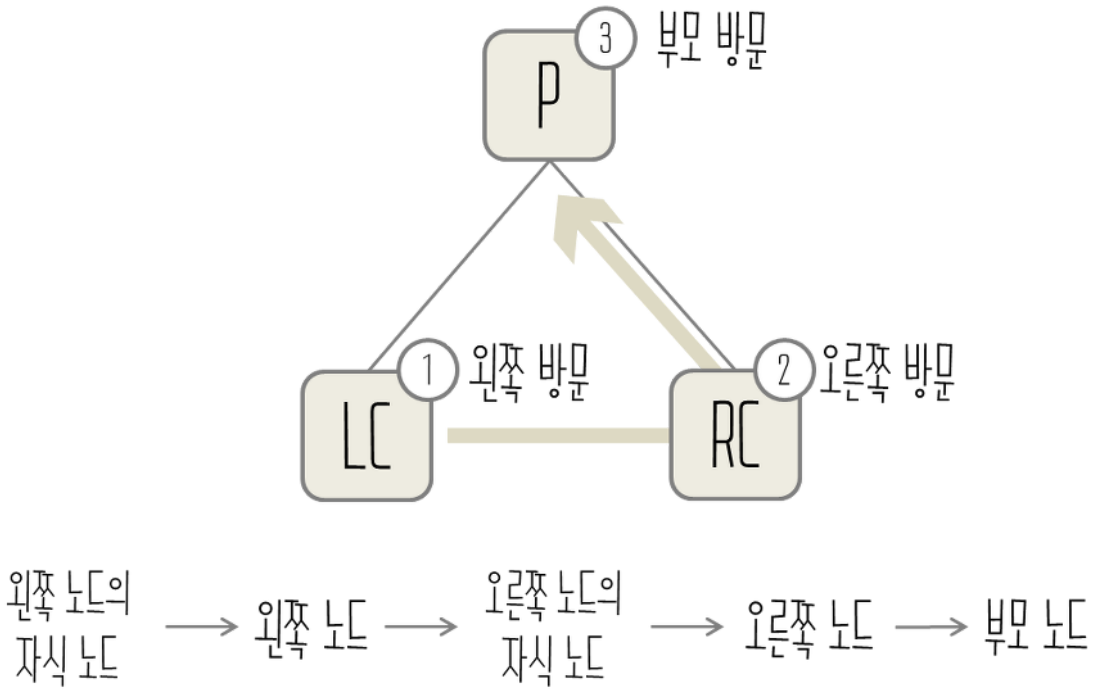
2. 중위 순회(in-order traversal)
    - 왼쪽 가지 → 현재 노드→ 오른쪽 가지




```
var recursiveInOrder = function (node) {
  if (!node) {
    return;
  }
  this.recursiveInOrder(node.leftNode);
  console.log(node.val);
  this.recursiveInOrder(node.rightNode);
};
```

3. 후위 순회(post-order traversal)

- 왼쪽 가지 → 오른쪽 가지 → 현재 노드




```
var recursivePostOrder = function (node) {
  if (!node) {
    return;
  }
  this.recursivePostOrder(node.leftNode);
  this.recursivePostOrder(node.rightNode);
  console.log(node.val);
};
```

사진 : [출처] https://butter-shower.tistory.com/80

이진 탐색 트리(Binary Search Tree)

- '이진 탐색'이 동작할 수 있도록 고안된 효율적인 탐색이 가능한 자료구조
- **이진탐색**

  : 탐색에 소요되는 시간 복잡도는 O(logN), but 삽입, 삭제가 불가능하다.

  이진 탐색이란??

  - 정렬된 리스트의 중앙에 있는 키 값을 조사하며 타색의 범위를 반으로 좁혀가는 방법

- **연결리스트**

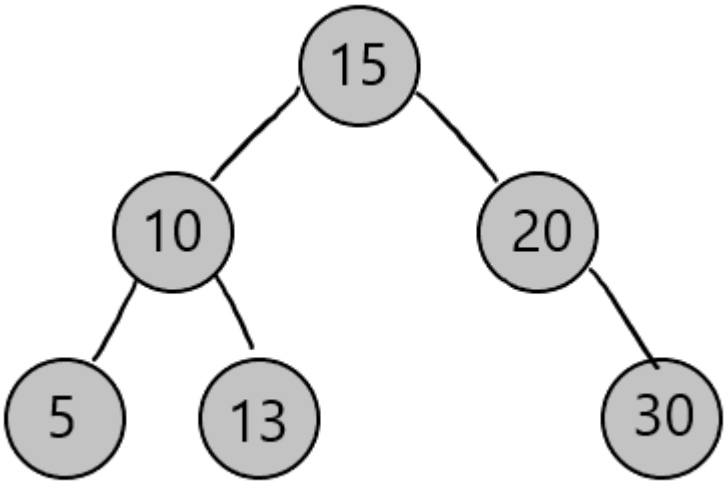
  : 삽입, 삭제의 시간복잡도는 O(1), but 탐색하는 시간복잡도가 O(N)
- 이진탐색과 연결리스트의 장점을 모두 얻는 것이 '**이진 탐색 트리**'이다
- 특징
  - **왼쪽 자식 노드 < 부모 노드 < 오른쪽 자식 노드**
  - 중복이 없어야 한다

이진 탐색 트리에서의 순회

- 준위 순회 방식
- 정렬된 순서를 읽을 수 있음

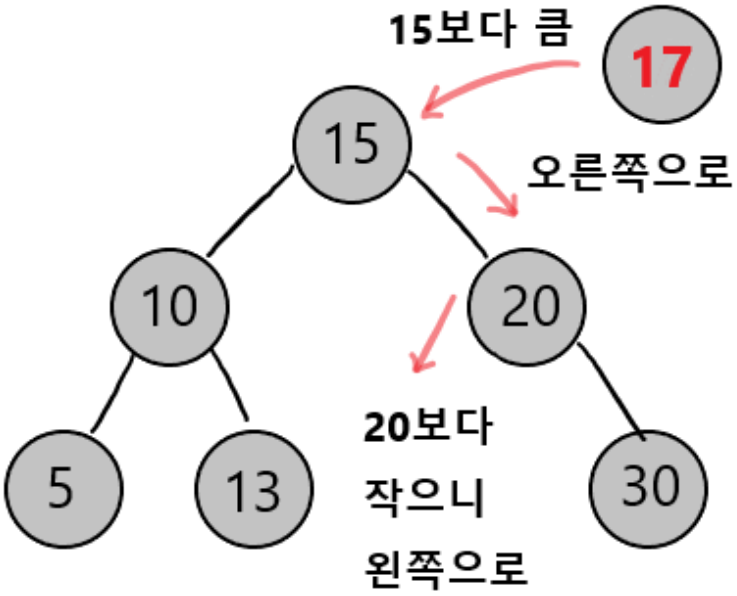
이진 탐색 트리에서의 탐색

- 나보다 작으면 왼쪽, 크면 오른쪽
- ex) 20을 찾고 싶으면 오른쪽으로~



이진 탐색 트리에서의 삽입

- 자신(루트노드)보다 작다면 왼쪽, 크다면 오른쪽으로 보낸다.
- 이렇게 탐색하다가 비교대상이 없다면 그 자리에 삽입한다.



이진 탐색 트리에서의 삭제

1. 자식이 없는 단말 노드 일 때
    - 그냥 삭제
2. 자식이 1개인 노드
    - 지워진 노드에 자식을 올린다.

3. 자식이 2개인 노드
    - 오른쪽 자식 노드에서 가장 작은값 or 왼쪽 자식 노드에서 가장 큰 값을 올린다.

균등 트리

- 노드 개수가 N 개일 때, 시간복잡도 O(logN)

편향 트리(한쪽으로만 뻗는 구조)

- 노드의 개수가 N 개일 때 시간복잡도 O(N)
- 트리를 사용할 이유가 없어짐
    - 그래서 개선된 트리 AVL Tree, RedBlack Tree가 있다,