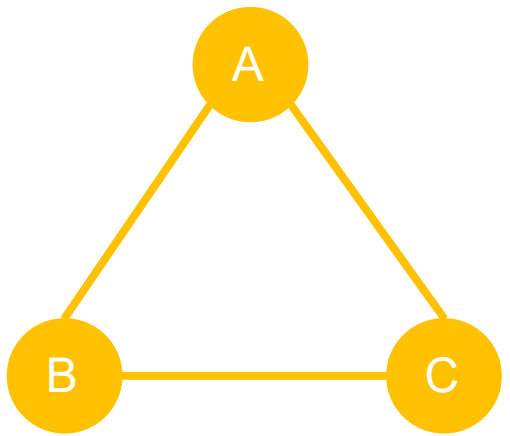
트리

트리란?

**트리는 정점(node)과 간선(edge)으로 연결된 그래프의 특수한 형태이다.
부모-자식 개념을 가지는 비순환적 경로로 연결되어있는 자료구조이다.**

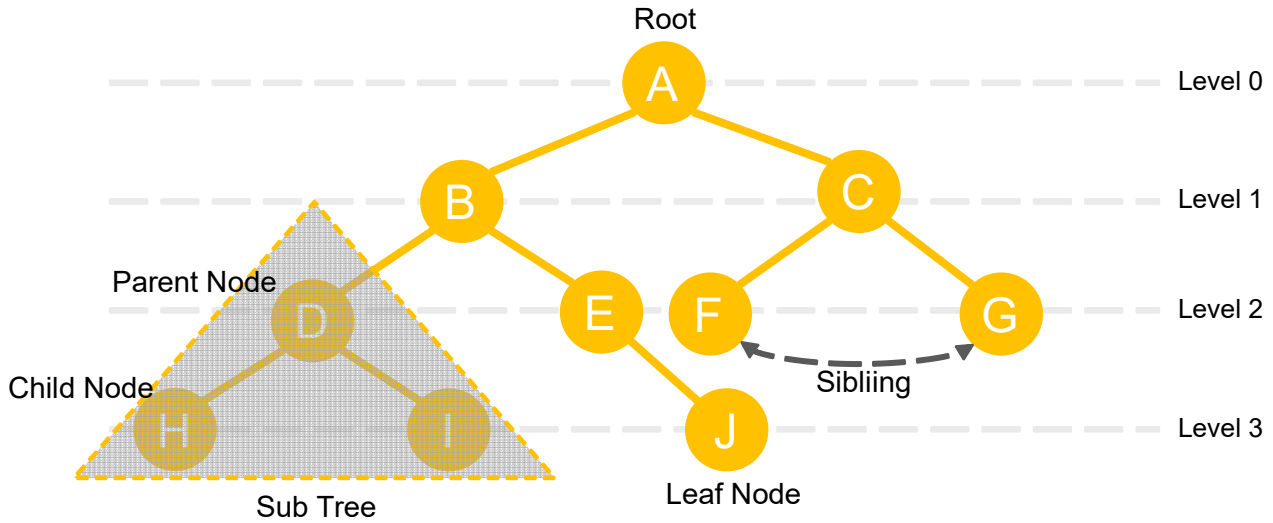
루트노드(A)를 기준으로 하나 이상의 노드와 비순환적 경로로 연결되어 있는 그래프는 트리구조이다.



위 그래프는 순환적 경로를 가지기 때문에 트리가 아니다.

트리

트리 용어

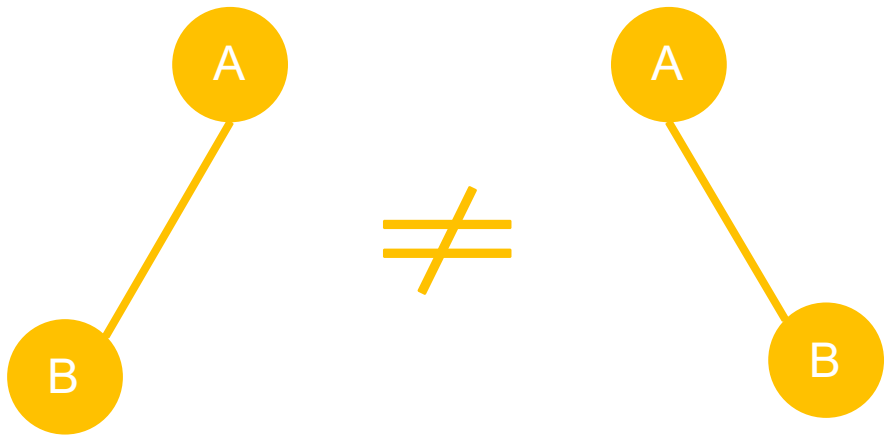


이름(한글)	설명	예시
정점(node)	트리를 구성하는 기본 원소	모든 노드
뿌리 노드(Root Node)	부모 노드가 없는 최상위 노드, 트리는 하나의 루트 노드를 갖는다.	A 노드
잎 노드(Leaf Node)	자식 노드가 없는 노드	H, I, J, F, G 노드
내부 노드(Internal Node)	Leaf node가 아닌 노드	A, B, C, D, E 노드
간선(Edge)	노드를 연결하는 선	-
노드의 차수(Degree)	노드가 가진 간선의 수	A의 Degree는 2
노드의 깊이(Depth)	루트 노드에서 어떤 노드까지 도달하기 위해 거쳐야하는 간선의 수	D의 Depth는 2
부모 노드(Present Node)	H와 I의 부모 노드는 D노드	
자식 노드(Child Node)	D노드의 자식 노드는 H와 I노드	
형제 노드(Sibling Node)	같은 부모 노드를 가지는 노드	F노드와 G노드
서브 트리(Sub Nree)	B노드를 기준으로 왼쪽 D, H, I 노드는 왼쪽 서브 트리	

트리이진 트리

이진 트리란?

트리 중에 각 노드가 최대 2개의 자식노드를 가질 때 이진트리(Binary Tree)라고 한다. (최대 Degree = 2)
같은 루트에 같은 자식노드가 하나 연결되어 있어도 자식노드의 위치가 왼쪽과 오른쪽으로 다르다면 서로 다른 트리이다.

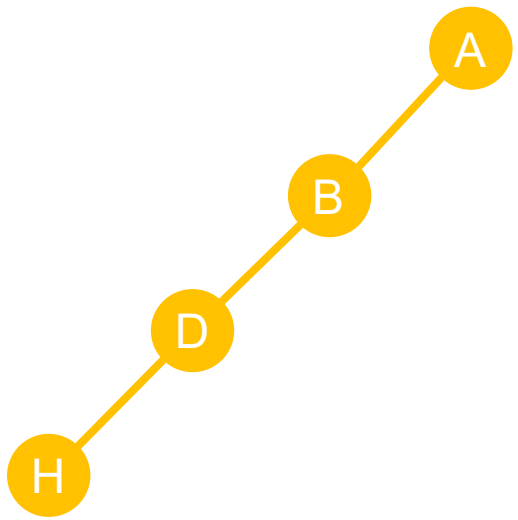


자식 노드의 위치가 다르면 서로 다른 트리다.

트리 이진 트리

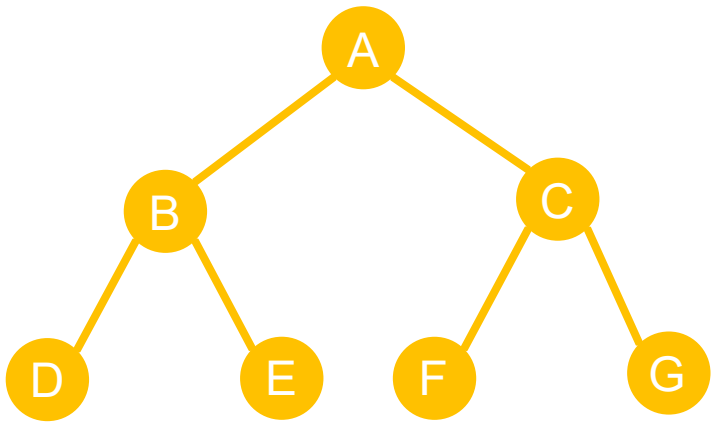
이진 트리 종류

편향 이진 트리



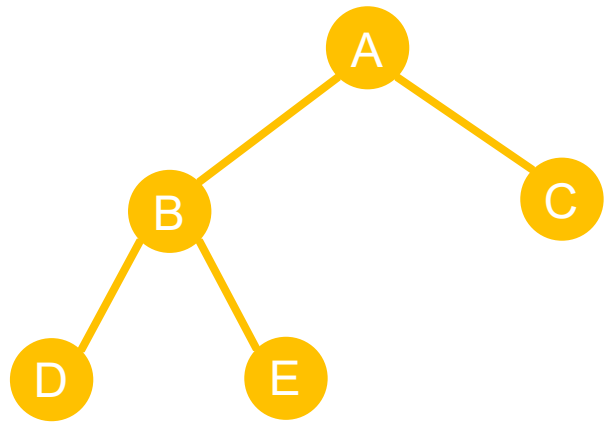
노드들이 한쪽으로 편향되어 생성된 이진트리

포화 이진 트리



트리의 높이가 모두 일정하며 리프 노드가 꽉찬 트리
(모든 리프 노드가 같은 level에 위치)

완전 이진 트리



마지막 레벨을 제외하고 노드들이 완전히 채워져 있고,
마지막 레벨은 왼쪽부터 채워진 트리

트리 트리순회

트리 순회란?

트리의 모든 노드를 방문하는 과정을 트리 순회(Tree Traversal)라고 한다.

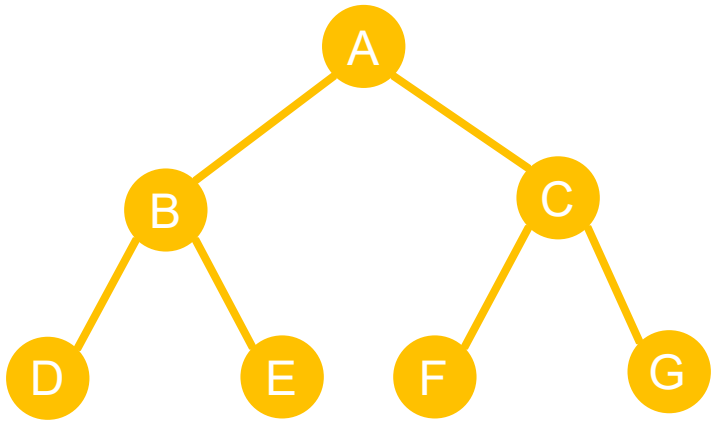
트리 순회 종류

- 전위 순회(Preorder) : 트리를 복사하거나, 전위 표기법을 구하는데 주로 사용됨
  1. Root 노드를 방문
  2. 왼쪽 서브 트리를 전위 순회
  3. 오른쪽 서브 트리를 전위 순회

- 중위 순회(Inorder) : 이진 탐색트리(BST)에서 오름차순 혹은 내림차순으로 값을 가져올 때 주로 사용됨
  1. 왼쪽 서브 트리를 중위 순회
  2. Root 노드를 방문
  3. 오른쪽 서브 트리를 중위 순회

- 후위 순회(Postorder) : 트리를 삭제할 때 주로 사용됨
  1. 왼쪽 서브 트리를 후위 순회
  2. 오른쪽 서브 트리를 후위 순회
  3. Root 노드를 방문

트리 트리순회




```
public void preOrder(Node node) {
    if(node != null) {
        System.out.print(node.data + " ");
        if(node.left != null) preOrder(node.left);
        if(node.right != null) preOrder(node.right);
    }
} // 루트 -> 왼쪽 -> 오른쪽
```

- 결과 : A B D E C F G

```
public void inOrder(Node node) {
    if(node != null) {
        if(node.left != null) inOrder(node.left);
        System.out.print(node.data + " ");
        if(node.right != null) inOrder(node.right);
    }
} // 왼쪽 -> 루트 -> 오른쪽
```

- 결과 : D B E A F C G

```
public void postOrder(Node node) {
    if(node != null) {
        if(node.left != null) postOrder(node.left);
        if(node.right != null) postOrder(node.right);
        System.out.print(node.data + " ");
    }
} // 왼쪽 -> 오른쪽 -> 루트
```

- 결과 : D E B F G C A