땅울림
자구 스터디

Made by 규정

5주차 - Tree

1 Tree 개념

2 이진트리

3 트리의 순회

4 트리의 구현

1. 트리의 개념

1 Tree

트리의 영어 'tree'은 나무를 뜻합니다. 나무에는 뿌리가 있고 위로 올라가면 수 많은 잎사귀와 열매로 나뉘게 됩니다. 이처럼 자료가 가지처럼 나뉘어서 계속 뻗어 나가는 계층구조를 트리 구조라고 부릅니다. 나무는 뿌리가 밑에 있고 위로 올라갈 수록 잎사귀로 나뉘어지지만 나무 구조는 그 반대입니다. 맨 위에 뿌리가 있고 밑으로 갈수록 자료가 나뉘게 되죠.

이와 같은 트리 구조는 일상 생활에서도 많이 쓰입니다. 스포츠 경기 대진표를 한 번 생각해 볼까요? 월드컵 경기 대진표를 생각해 보세요. 맨위에는 결승전이 있습니다. 그 밑으로는 두 개의 준결승전이 나열될 것입니다. 그 밑으로는 네 개의 8강전으로 나뉘게 되죠. 트리구조의 대표적인 예라고 할 수 있습니다.

컴퓨터에서도 이런 구조를 사용합니다. 저장하는 공간과 폴더를 통해서 드러나는데요. C드라이브를 맨 위의 뿌리라고 했을 때, 그 밑으로는 수 많은 폴더들이 존재합니다. 특정 폴더 안에 들어가면 또 수많은 폴더와 파일들로 나뉘게 되는 것이죠.

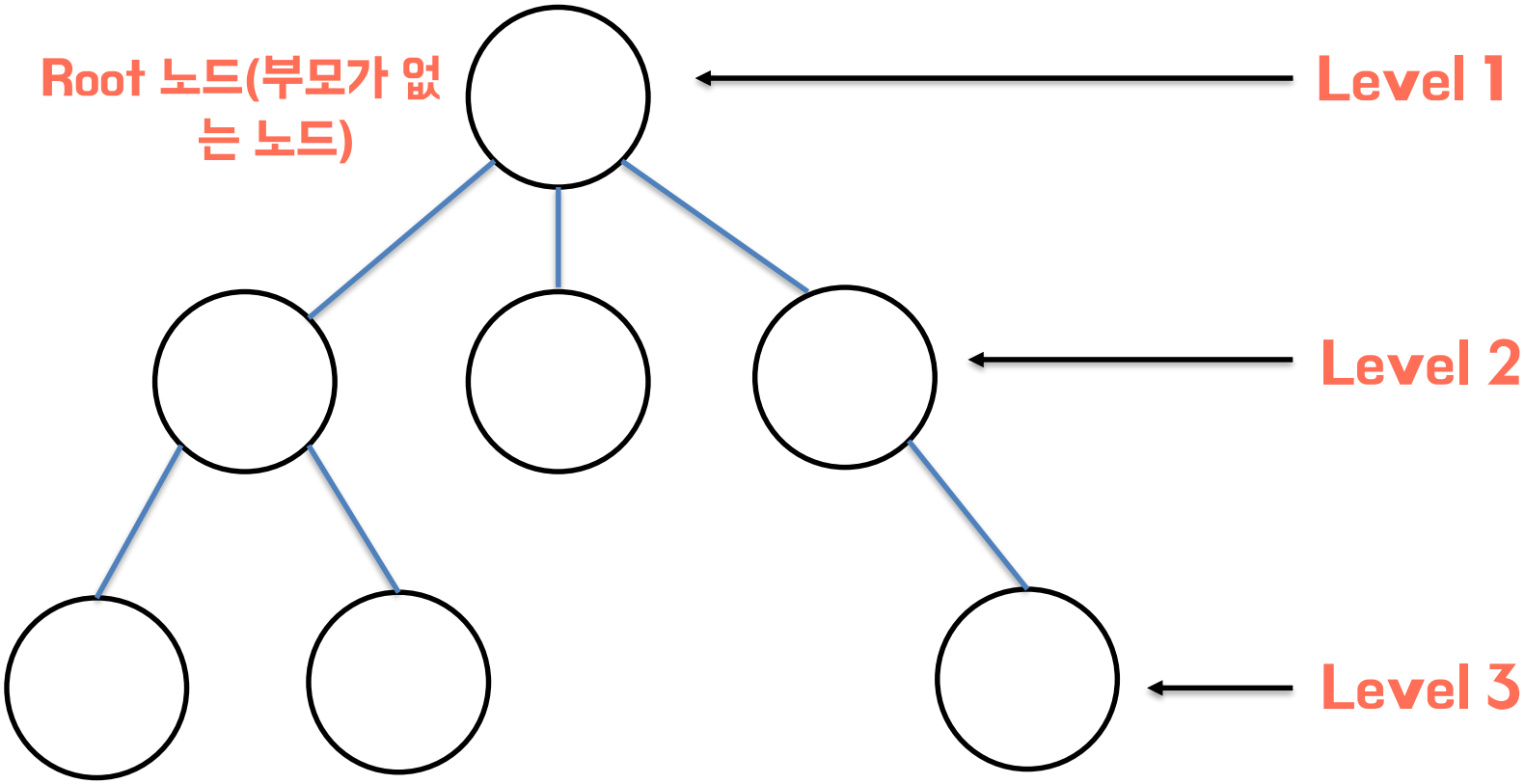
트리 구조는 복잡하고 많은 파일들을 알아보기 손쉽게 정리할 수 있을 뿐 아니라, 목적이나 계획, 계층이나 중요도에 맞게 나열할 수 있기 때문에 일상생활 뿐 아니라 컴퓨터에서도 자주 사용되는 자료구조 입니다.

1 Tree

Tree?

대진표, 지휘계통, 가족관계도 등 단계나 계층이 있는 구조를 표현하기 위한 자료구조

1 Tree 구조



트리의 Height(높이) : 2

1 Tree 용어정리

노드(node) : 트리를 구성하는 기본 원소
간선(edge) : 노드와 노드를 연결하는 선

루트(root) 노드 : 부모가 없는 최상위 노드(트리의 시작점)
리프(leaf) 노드(외부 노드, external node) : 자식이 없는 노드
내부 노드(internal node) : 자식이 하나라도 있는 노드

레벨(level) : 트리의 층의 번호(루트 : 1)
노드의 깊이(depth) : 어떤 노드의 조상의 수(루트 : 0)
트리의 높이(height) : 트리에서 가장 깊은 노드의 깊이

노드의 차수(degree) : 한 노드의 자식 수
트리의 차수(degree of tree) : 트리의 최대 차수

1 Tree 용어정리

부모(parent) 노드 : 노드 u의 상위 노드
자식(child) 노드 : 노드 u의 하위 노드
형제(sibling) 노드 : 부모가 같은 노드

조상(ancestor) 노드 : 어떤 노드의 부모 노드들(부모의 부모, 부모의 부모의 부모 등등)
후손(descendant) 노드 : 어떤 노드의 자식 노드들(자식의 자식, 자식의 자식의 자식 등등)

Subtree : 기존 트리에 속한 부분적인 트리

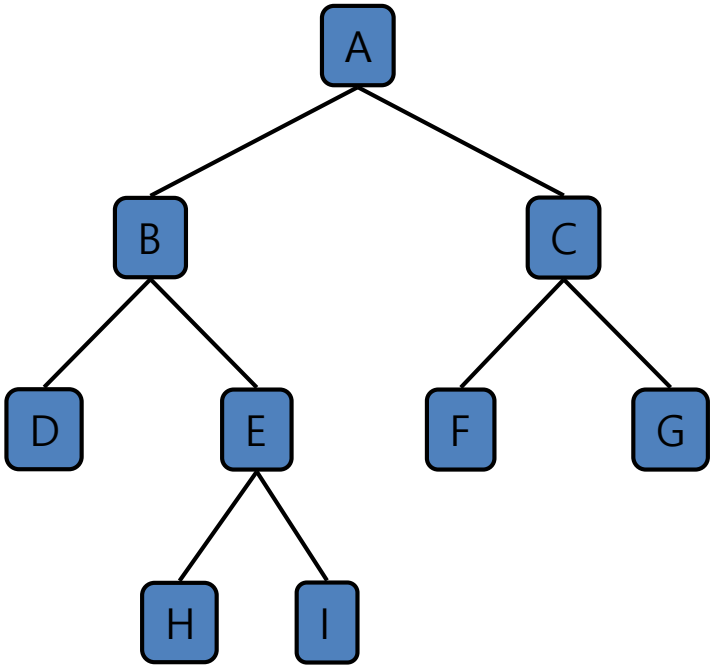
2. 이진트리

2 이진트리

Binary Tree(이진 트리)

· 모든 노드가 최대 두 개의 자식 노드를 가지는 트리

· 왼쪽 자식을 left, 오른쪽 자식을 right로 표현



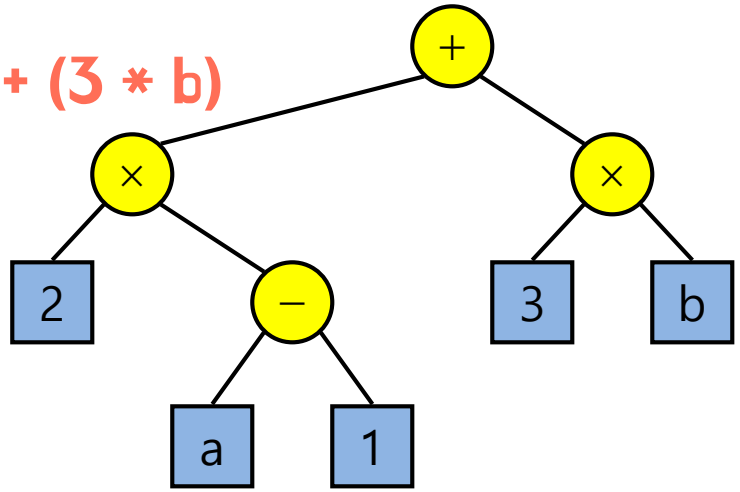
2 이진트리의 활용

Arithmetic Expression Tree

산술식을 이진트리로 표현한 트리

- 내부 노드 : 연산자
- 외부 노드 : 피연산자

(2 * (a − 1)) + (3 * b)

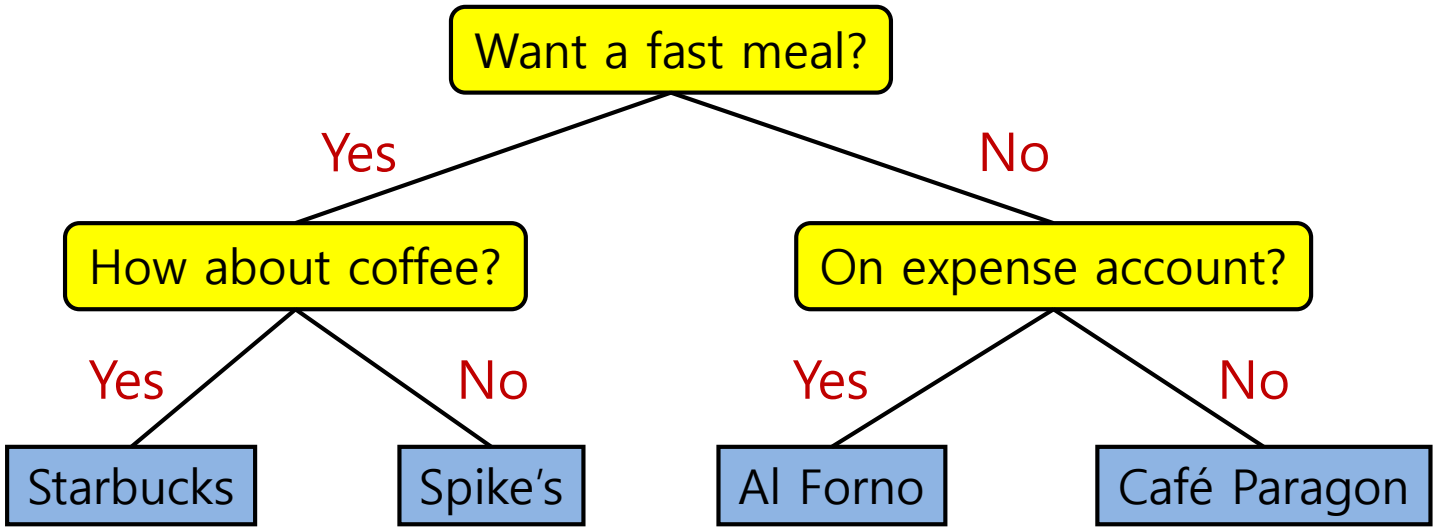


2 이진트리의 활용

Decision Tree

의사 결정 과정을 트리로 표현

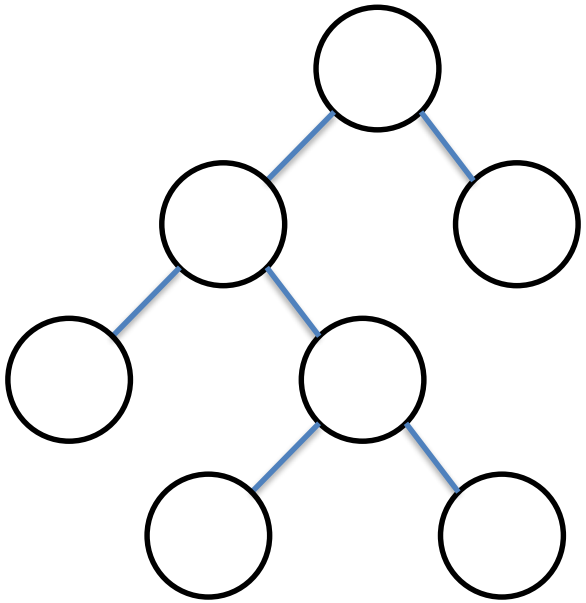
- 내부 노드 : 질문(yes or no)
- 외부 노드 : 최종 의사



2 이진트리의 종류

Full(Proper) Binary Tree
(정 이진트리)

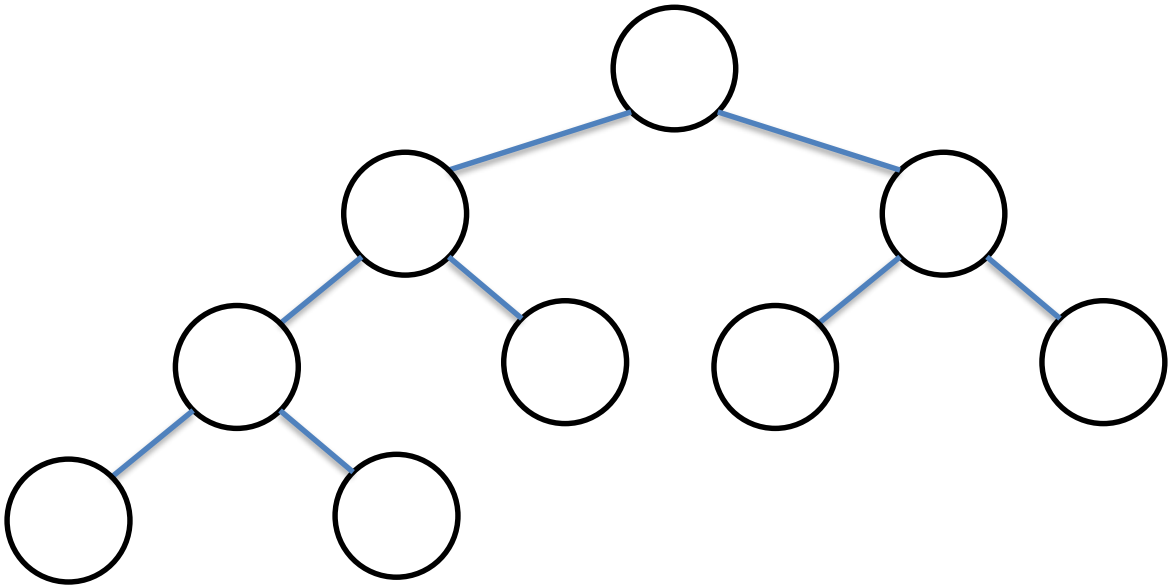
모든 노드가 0개 또는 2개의 자식을 가짐



2 이진트리의 종류

Complete Binary Tree
(완전 이진트리)

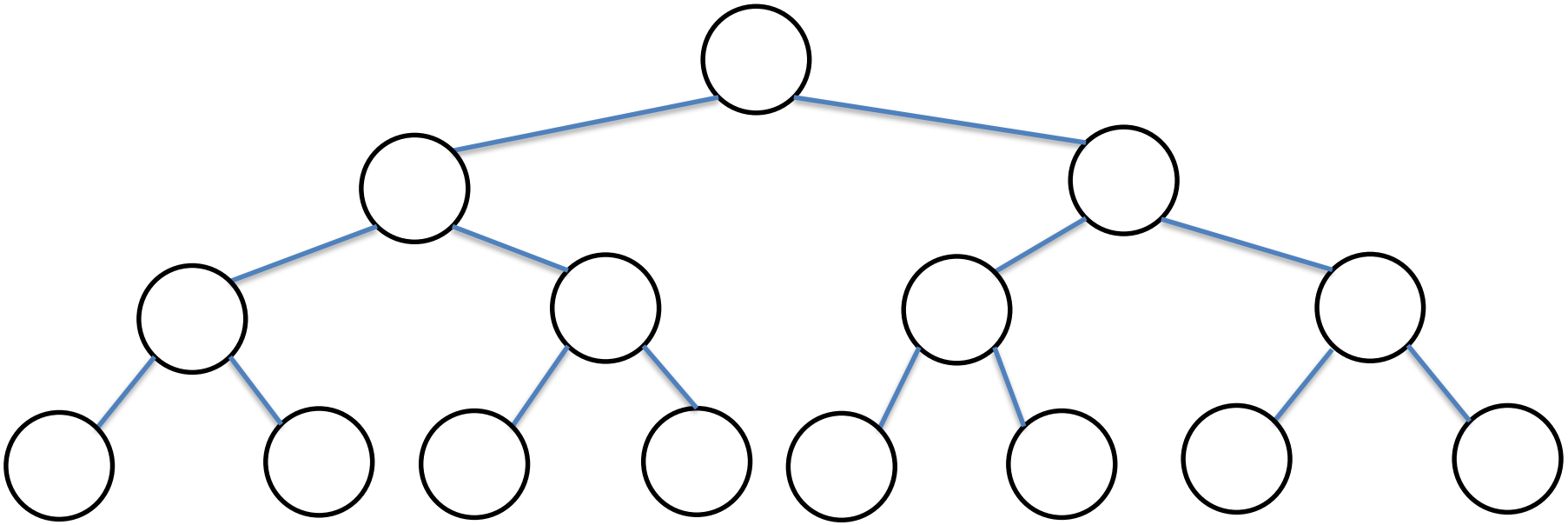
마지막 레벨을 제외하고 모든 레벨이 완전 채워져 있으며, 마지막 레벨의 노드는 왼쪽부터 채워져있음



2 이진트리의 종류

Perfect Binary Tree
(포화 이진트리)

모든 레벨의 노드가 완전히 채워져 있으며, 리프 노드들이 모두 같은 depth를 가짐(full + complete)



3. 트리의 순회

3 Tree의 순회

· 트리의 노드들을 탐색하는데, 선형구조가 아니므로 순회하는데 일정한 규칙이 있어야함.

· 크게 네 가지로 나뉨

3 Tree의 순회

루트의 위치만 기억하면 편함

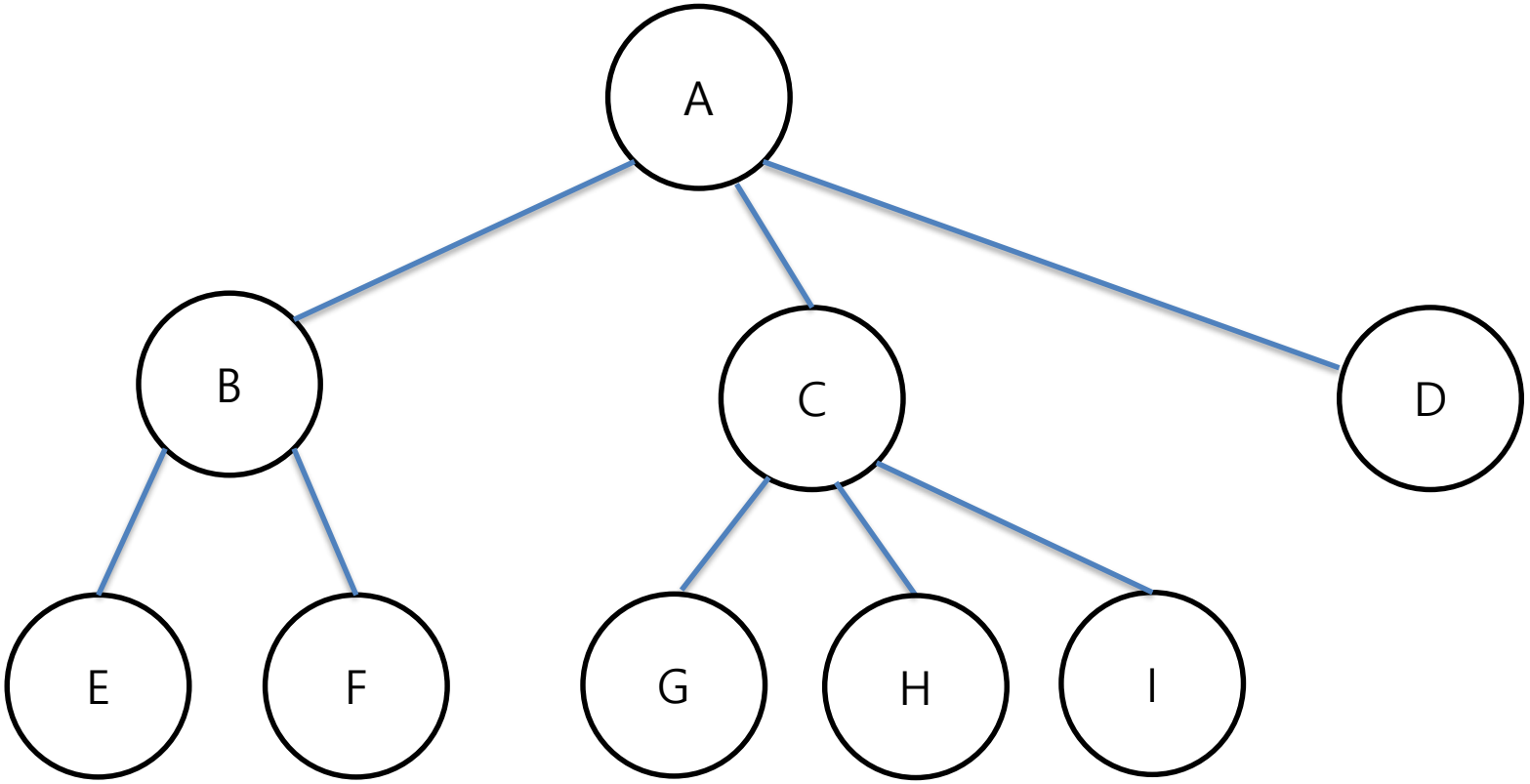
DFS

- 전위 순회(preorder) :
  1. Root 노드 탐색
  2. Left subtree 탐색
  3. Right subtree 탐색
- 중위 순회(inorder) :
  1. Left subtree 탐색
  2. Root 노드 탐색
  3. Right subtree 탐색
- 후위 순회(postorder) :
  1. Left subtree 탐색
  2. Right subtree 탐색
  3. Root 노드 탐색

너비 우선 순회(BFS) : 레벨 순서에 따라서 순회

3 Tree의 순회

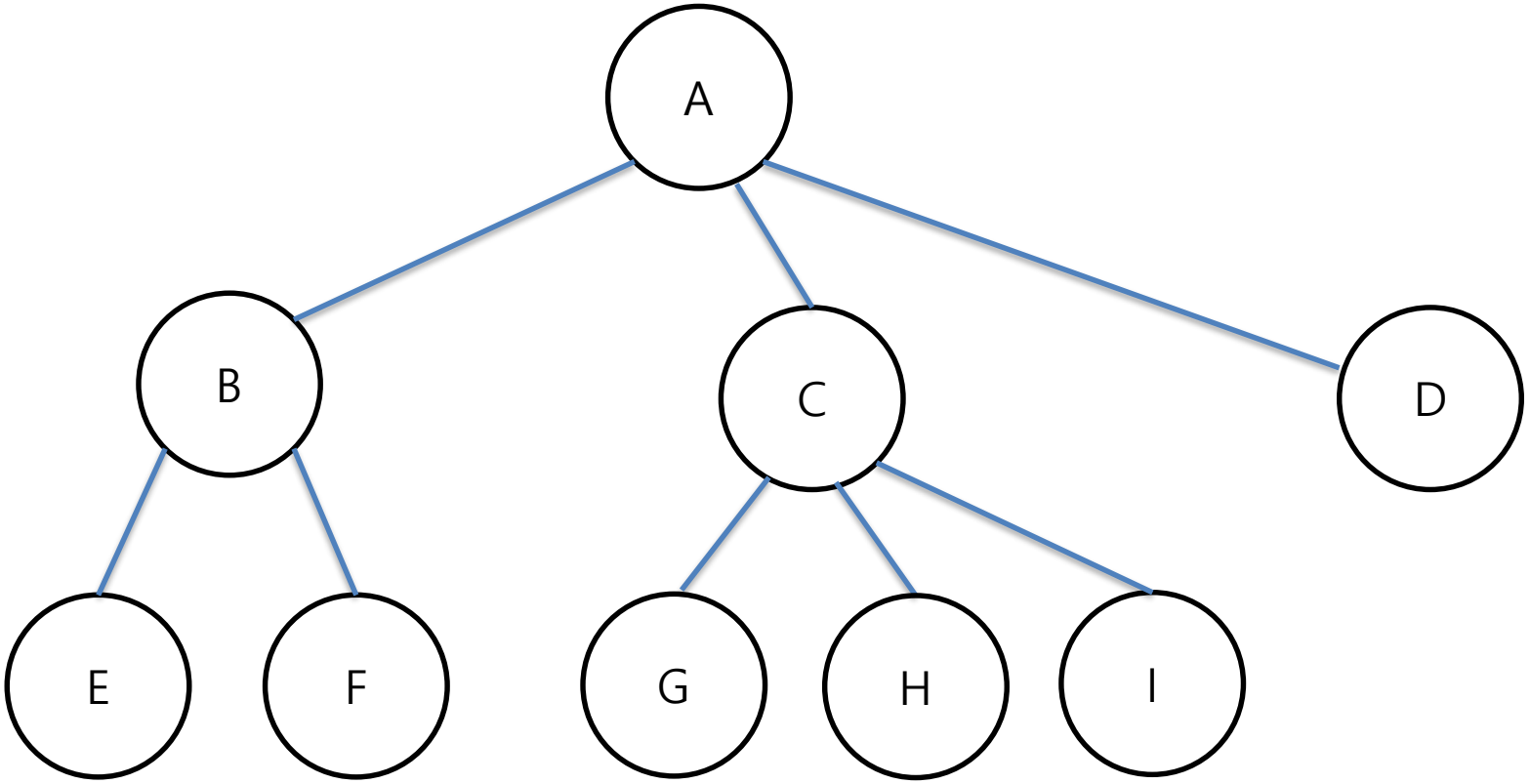
전위 순회(preorder) : 부모–〉왼–〉오



A–B–E–F–C–G–H–I–D

3 Tree의 순회

중위 순회(inorder) : 왼-〉부모-〉오

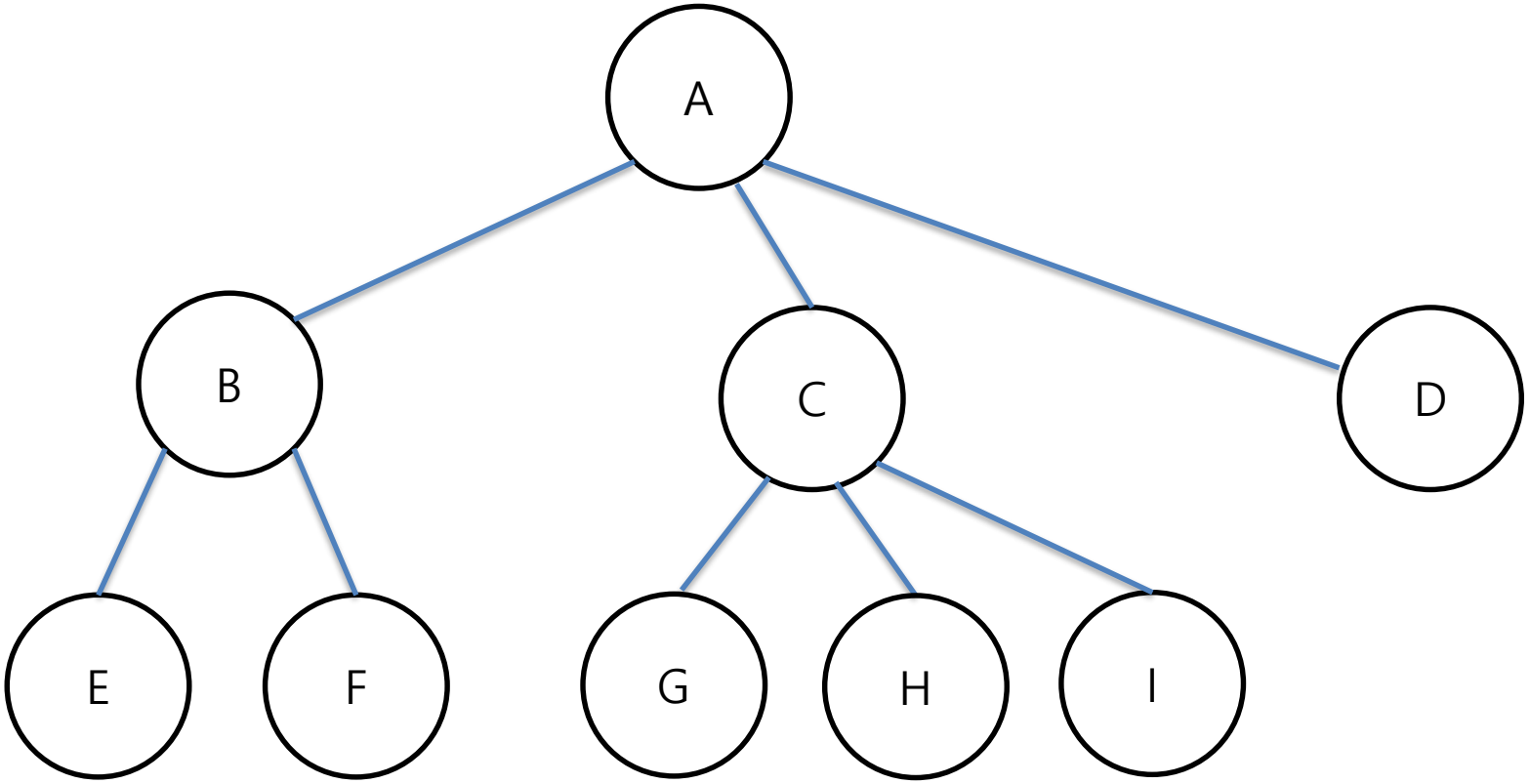


???

이진트리가 아니면 순서를 정의하기 어려움

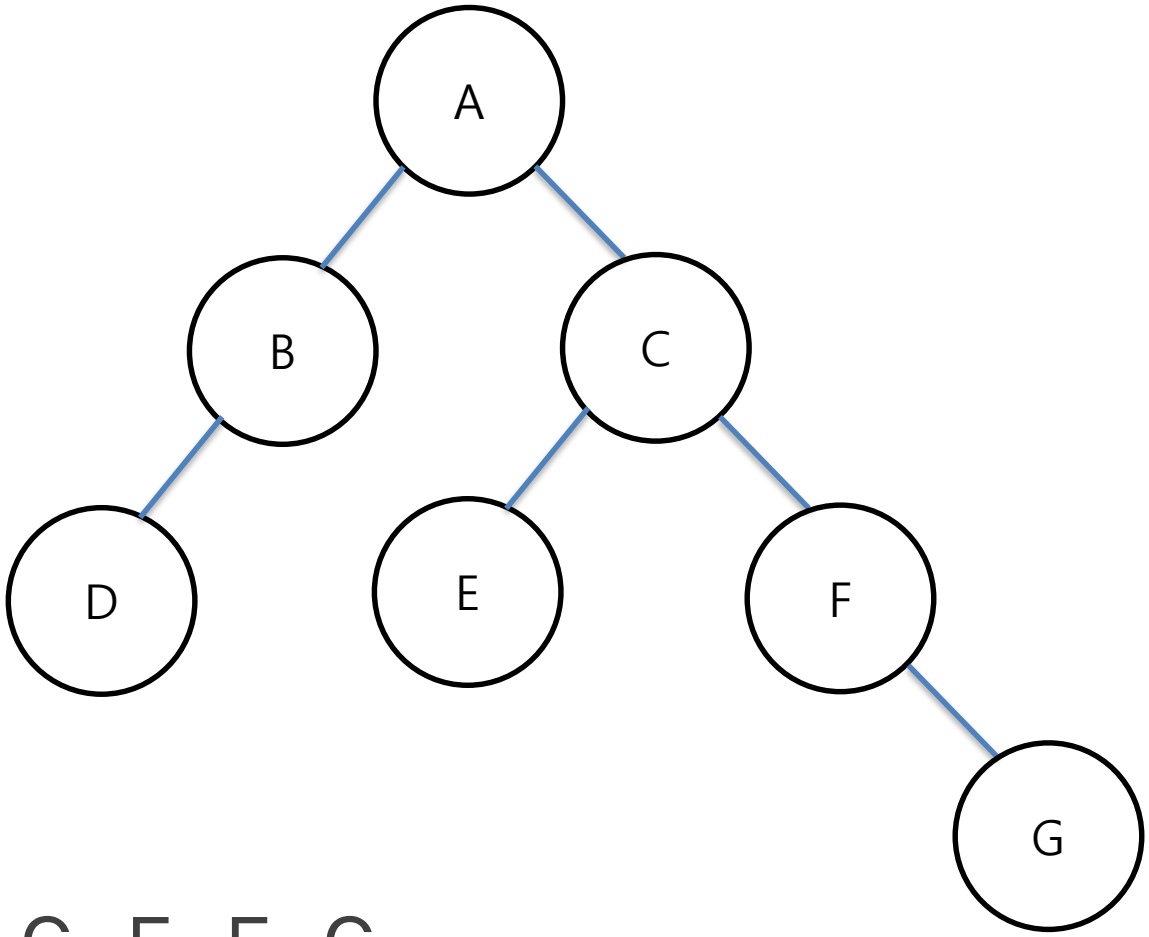
3 Tree의 순회

후위 순회(postorder) : 왼-〉오-〉부모



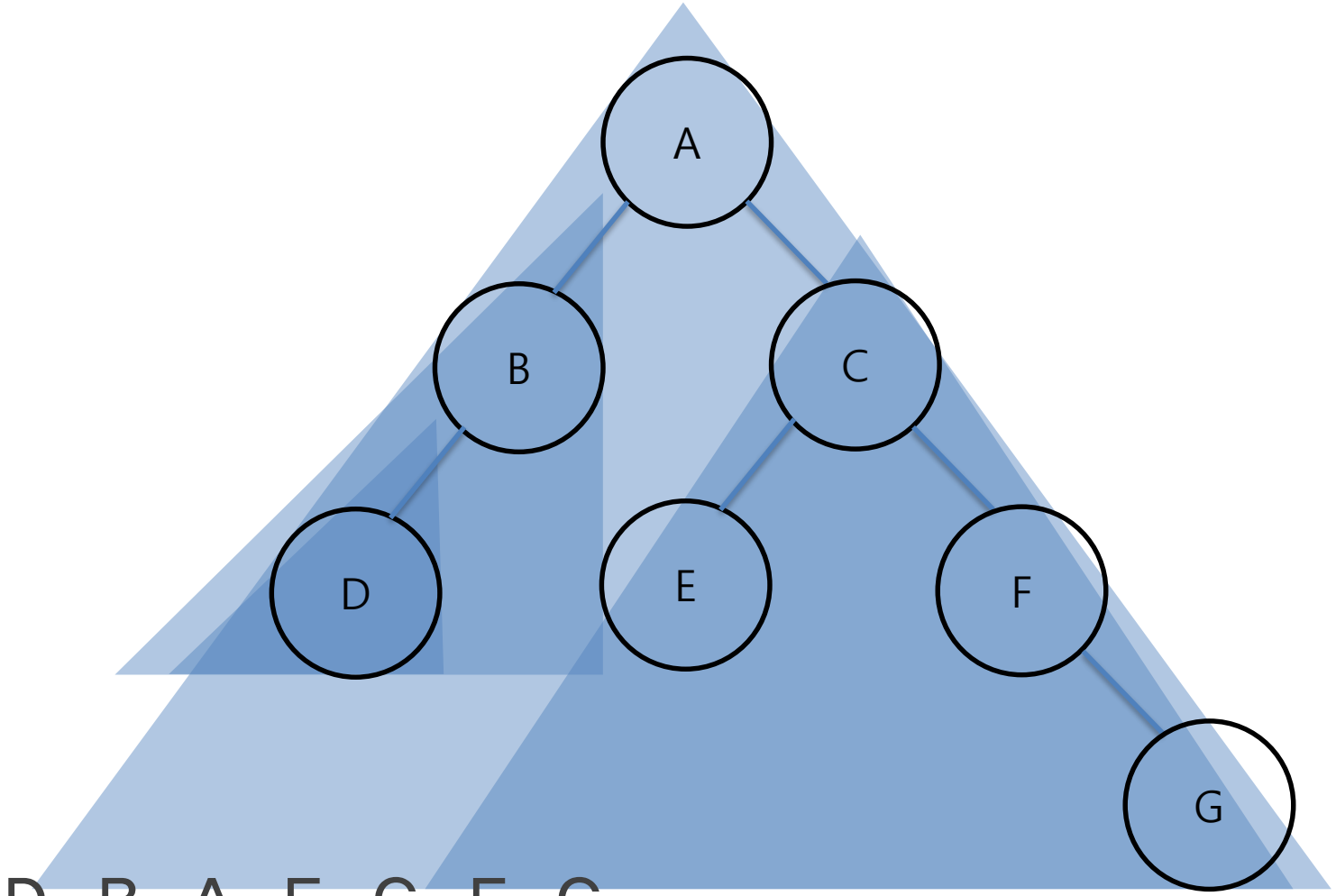
E-F-B-G-H-I-C-D-A

전위 순회(preorder) : 부모–〉왼–〉오



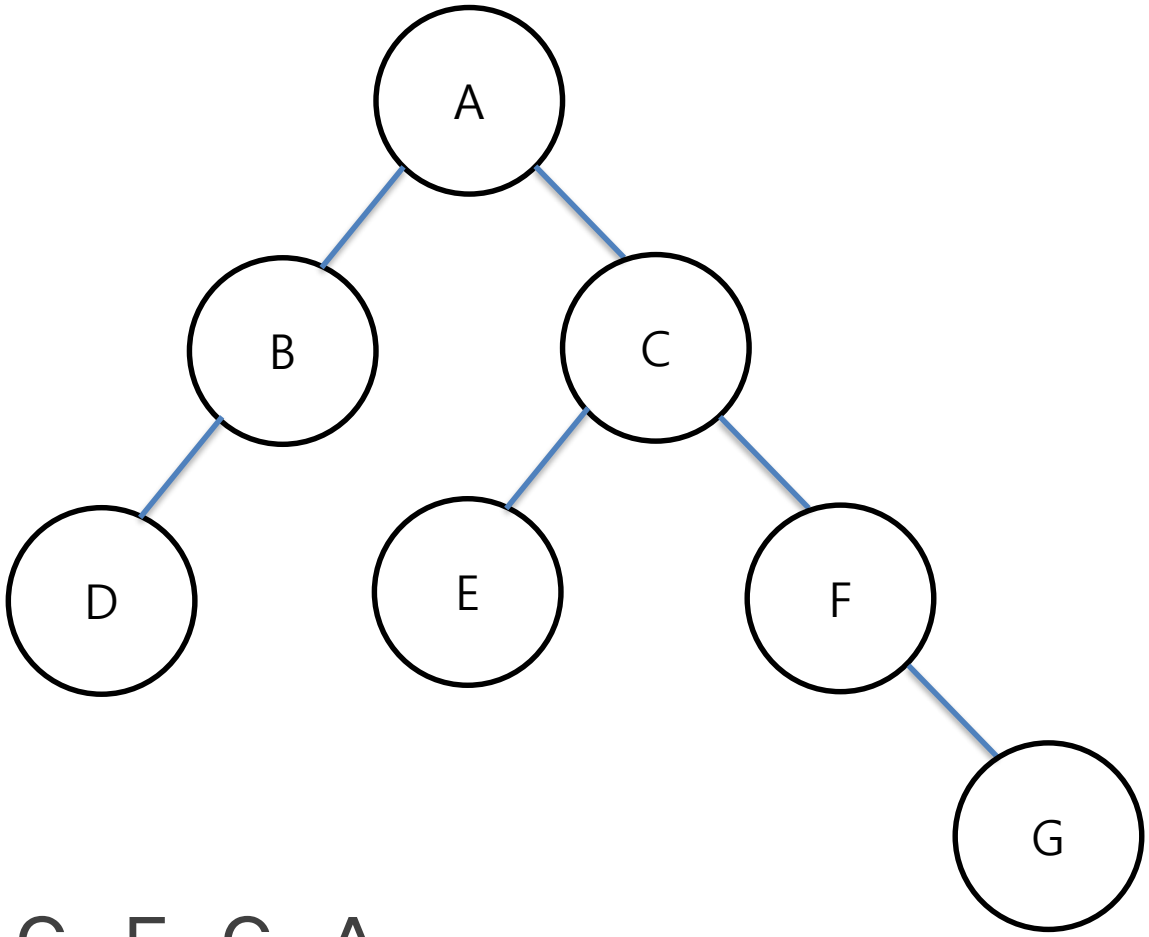
A–B–D–C–E–F–G

중위 순회(inorder) : 왼-〉부모-〉오



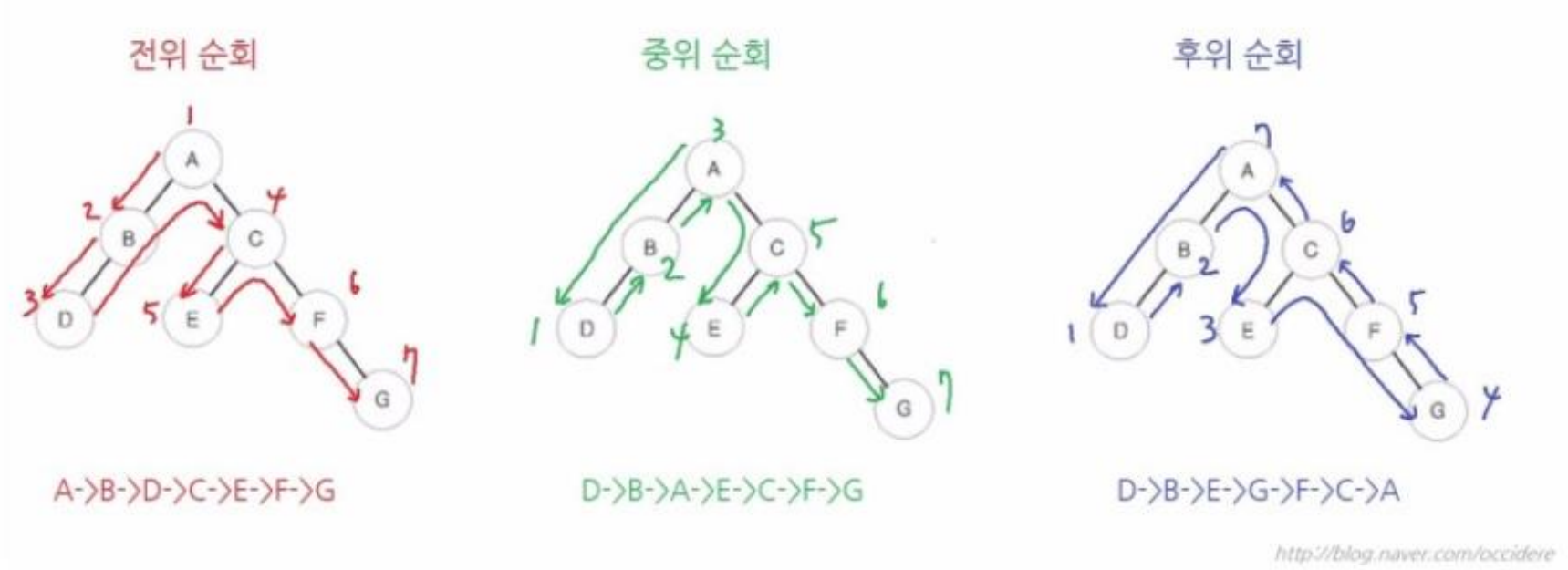
D-B-A-E-C-F-G

후위 순회(postorder) : 왼->오->부모



D-B-E-G-F-C-A

3 Tree의 순회


전위 순회
A->B->D->C->E->F->G
중위 순회
D->B->A->E->C->F->G
후위 순회
D->B->E->G->F->C->A
http://blog.naver.com/occidere

4. 트리의 표현

4 Tree의 표현

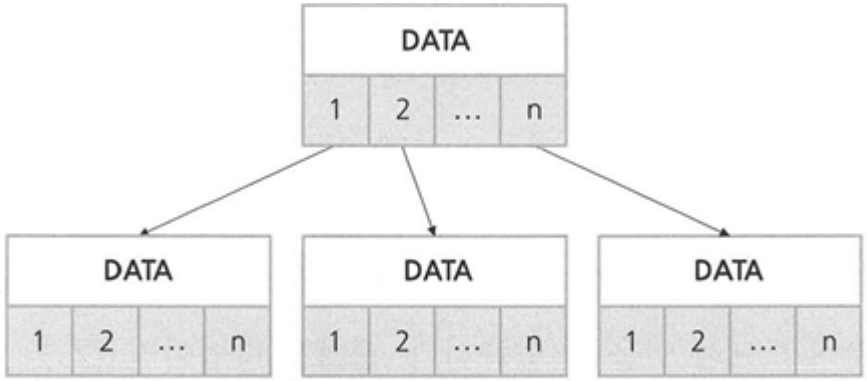
1. N-링크 표현법

2. 왼쪽 자식 – 오른쪽 형제 표현법

4 N-링크 표현법

N-링크 표현법

- 각 노드가 N개만큼의 링크를 가지고 있음
- 링크들이 각각 자식 노드를 가리킴



4 N-링크 표현법

장점

- 트리의 구조를 그대로 표현해서 직관적임
- 대부분의 상황에서 시간복잡도가 우수함

단점

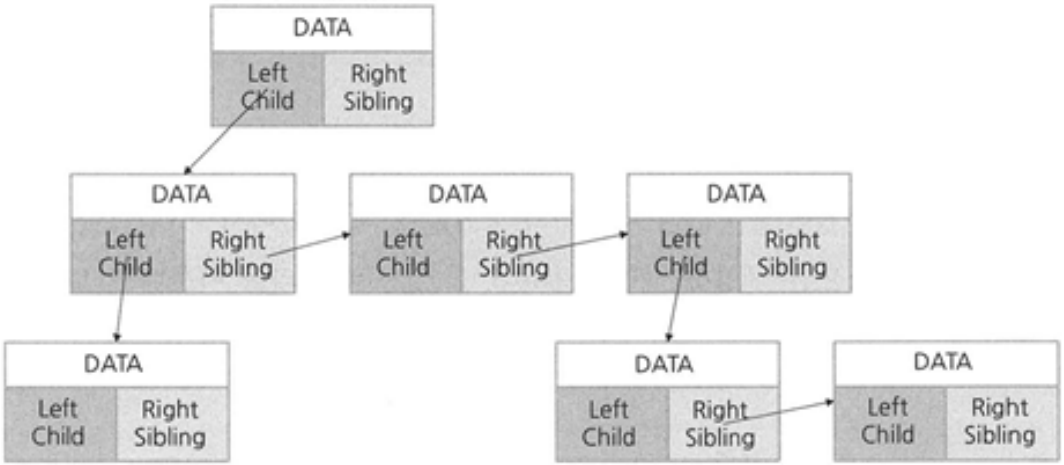
- 모든 노드가 N개의 링크를 가지고 있으므로 메모리 낭비가 심함
- 이를 해결하려면 연결리스트 사용해야하는데, 구현이 개복잡

이런 이유로 N-링크 표현법은 거의 쓰이지 않음

4 왼쪽 자식 - 오른쪽 형제 표현법

왼쪽 자식 – 오른쪽 형제 표현법

- 모든 트리를 이진트리로 구현하는 방법
- left child는 자식을, 오른쪽 child는 형제를 표현



4 왼쪽 자식 - 오른쪽 형제 표현법

장점

- 구현이 매우 간편함(사용도 간편함)
- 메모리 사용에 효과적

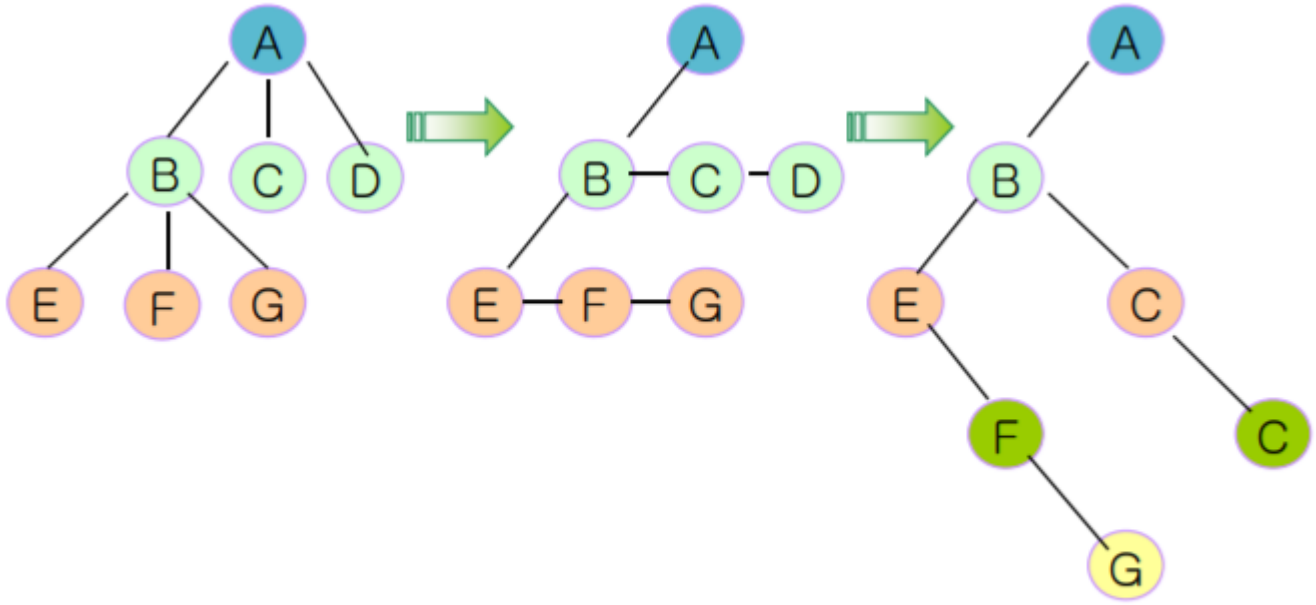
단점

- 구조가 바뀌므로 직관적이지 않음
- 자식 노드를 방문할 때 한번에 방문할 수 없음

진짜 웬만하면 거의 다 이 방식 사용

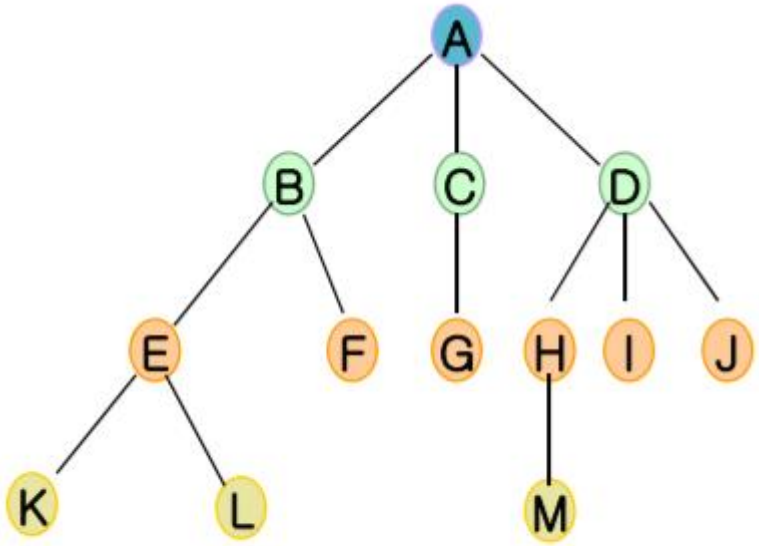
문제가 이진트리로 주어지거나,

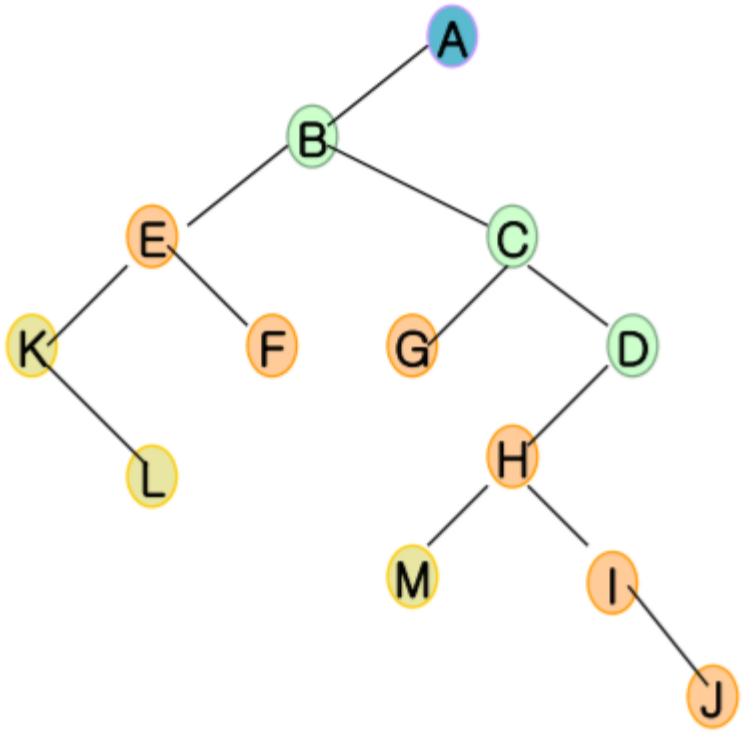
4 이진 트리로 변형



트리 왼쪽자식-오른쪽형제 트리 이진 트리

4 이진 트리로 변형

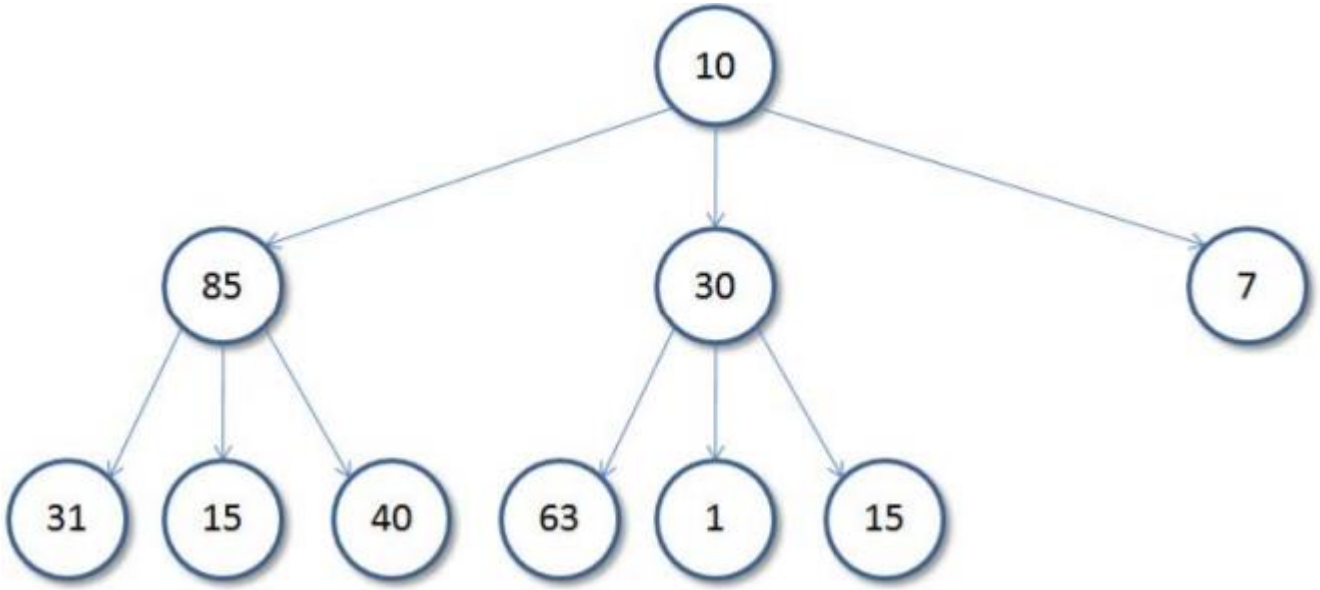




일반 트리 → 이진 트리

4 이진 트리로 변형

Q) 다음 차수가 3인 트리를 이진트리로 변형해 보아라.



감사합니다.

Made by 규정