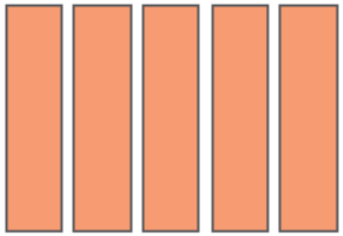
데이터구조 6장

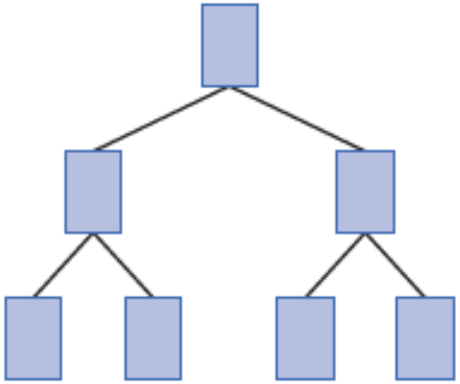
06-1. 트리개념

트리(TREE)

- 트리 : 계층적인 구조를 나타내는 자료구조
- 트리는 부모-자식 관계의 노드들로 이루어짐
- 대표적인 비선형 자료구조



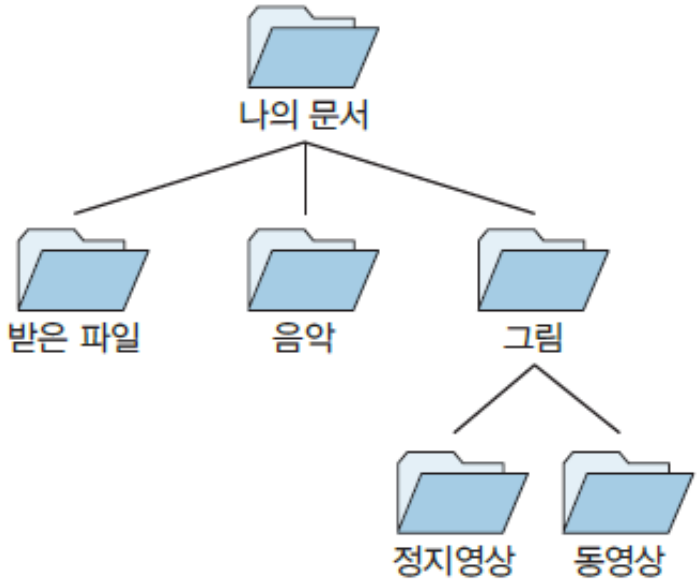
선형 자료구조

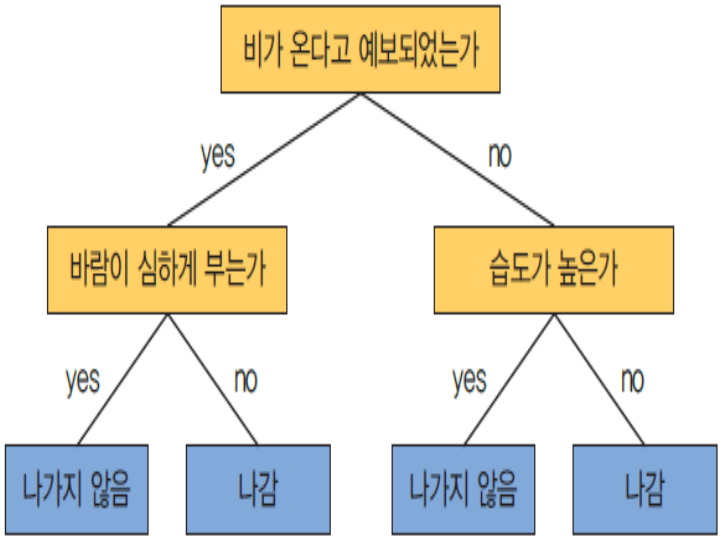


비선형 자료구조

트리의 용도

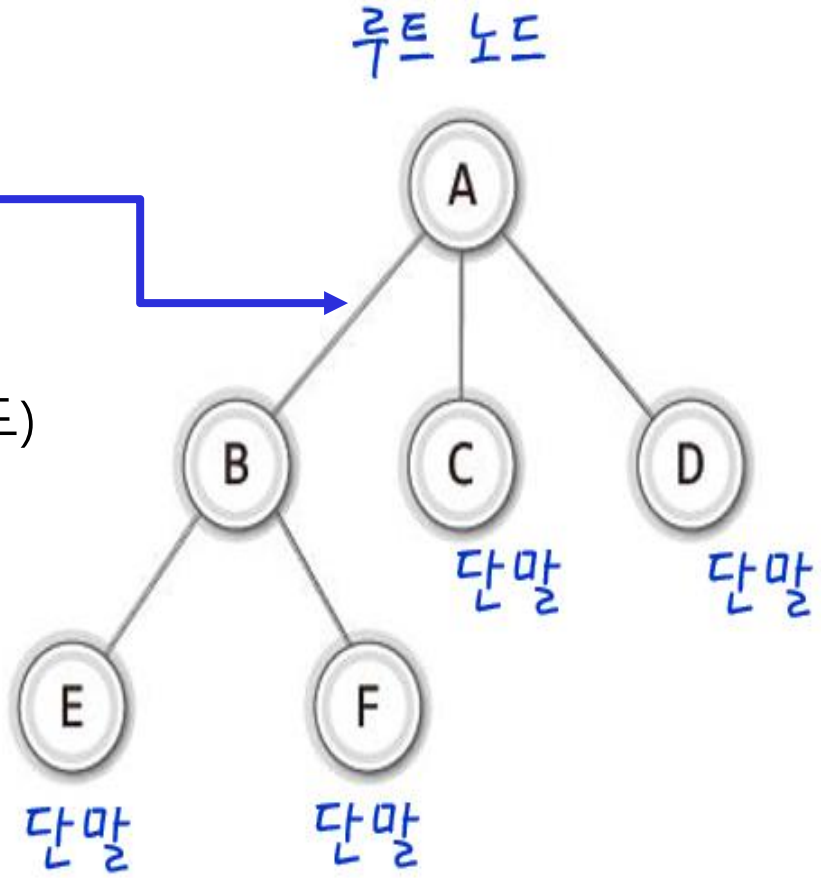
- 계층적인 조직 표현
- 컴퓨터 디스크의 디렉토리 구조
- 인공지능에서의 결정트리 (decision tree)





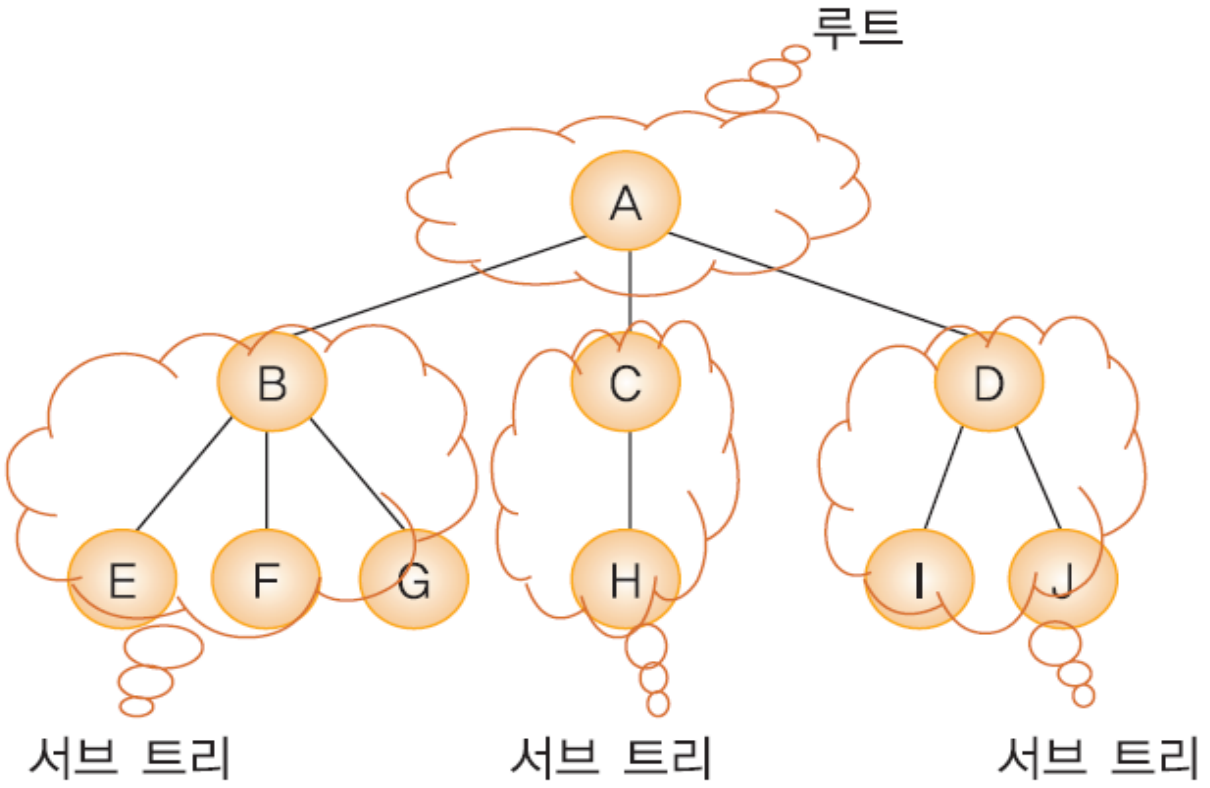
트리의 구조

- 이해가 필요한 용어
  - 노드(node) :
    - 트리의 구성요소
  - 간선(edge) :
    - 노드와 노드를 연결하는 선
  - 루트(root) :
    - 부모가 없는 노드(A)
  - 단말노드(terminal, 리프leaf노드)
    - 자식이 없는 노드(E, F, C, D)
  - 비단말노드
    - 자식을 가지는 노드(A,B)

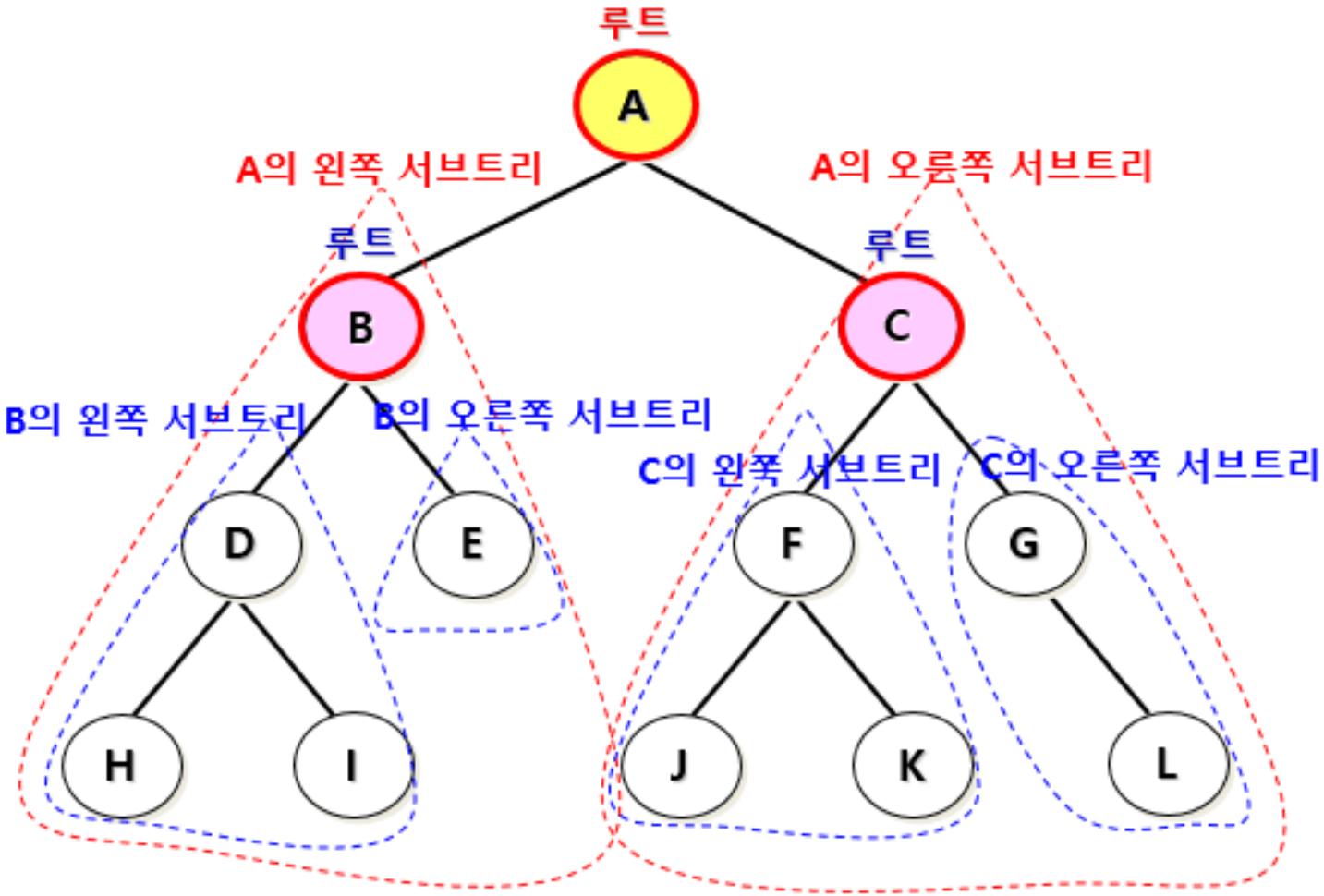


트리의 구조

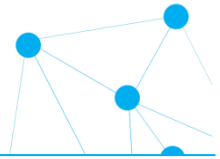
- 서브트리(subtree)
  - 하나의 노드와 자손들로 이루어짐



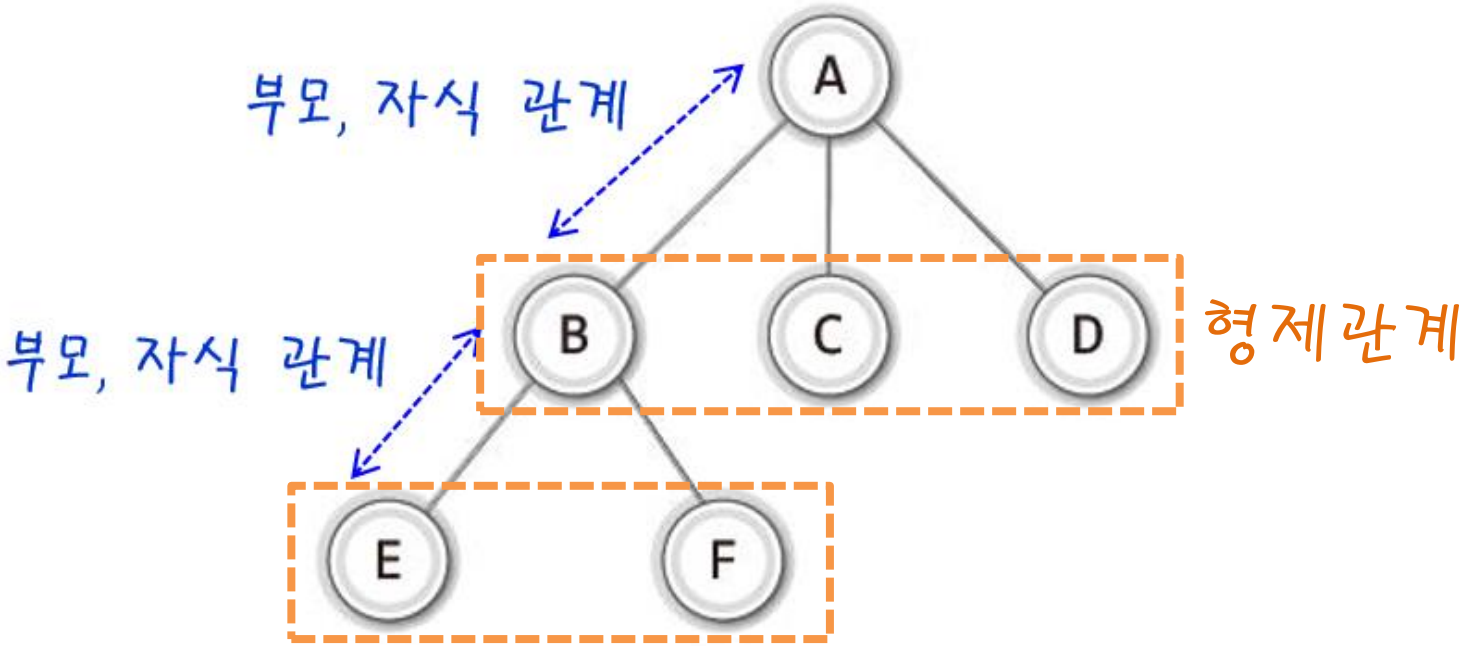
서브트리


루트
A
A의 왼쪽 서브트리
A의 오른쪽 서브트리
루트
루트
B
C
B의 왼쪽 서브트리
B의 오른쪽 서브트리
C의 왼쪽 서브트리
C의 오른쪽 서브트리
D
E
F
G
H
I
J
K
L

트리의 구조

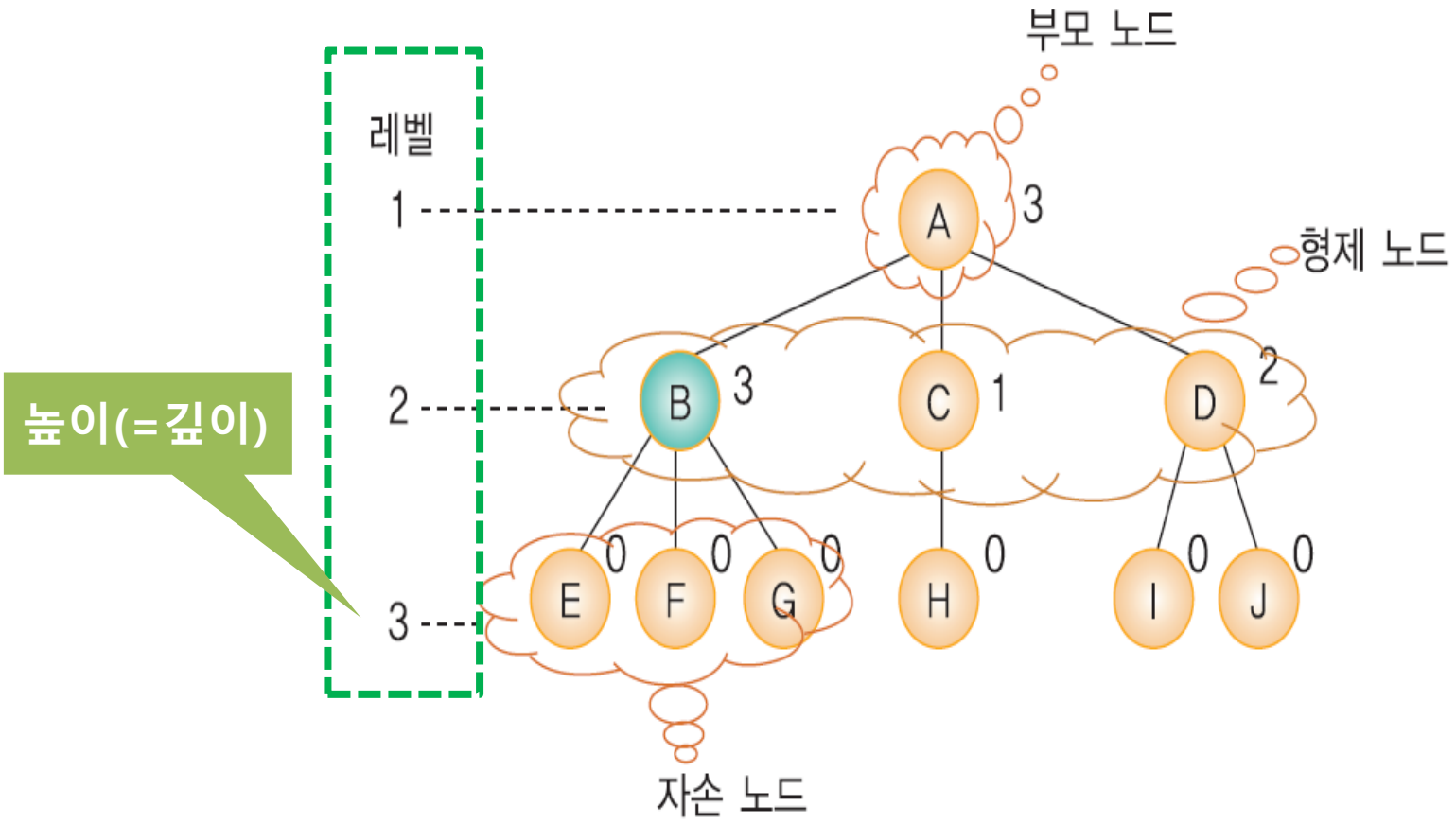


- 자식, 부모, 형제, 조상, 자손 노드 : 인간과 동일



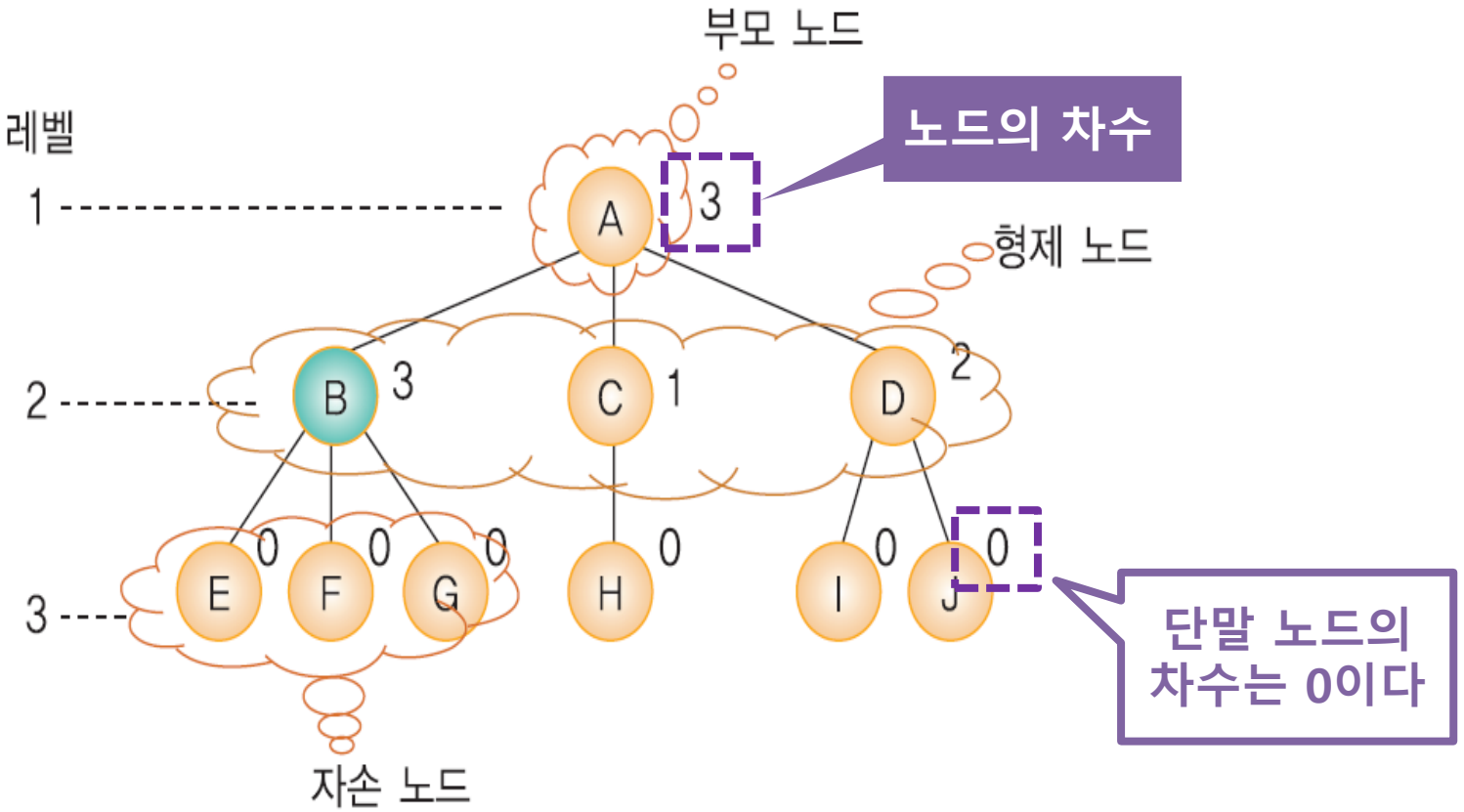
트리의 구조

- 레벨(level) : 루트 노드를 1로 봤을 때 트리의 각층의 번호
- 높이(height 또는 깊이 depth) : 트리의 최대 레벨

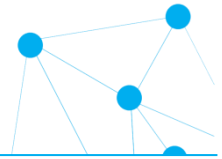


트리의 구조

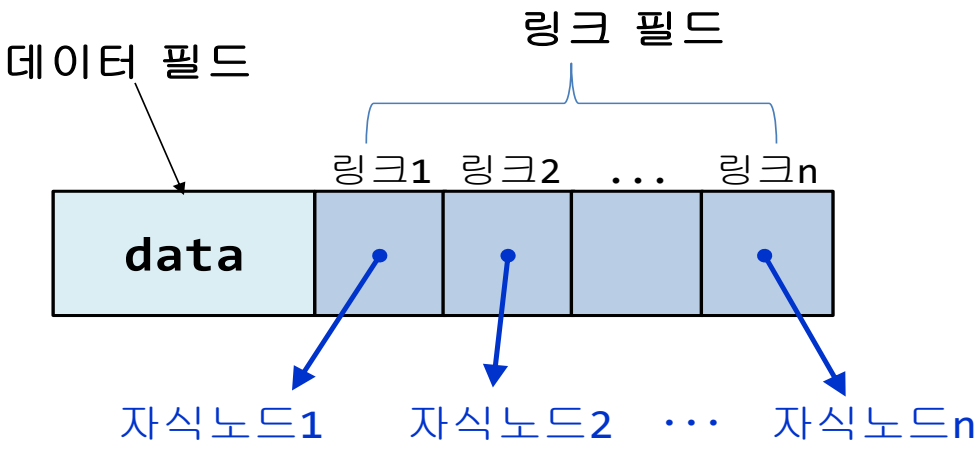
- 노드의 차수(degree) : 노드의 자식수
- 트리의 차수 : 노드 차수 중 최대값



트리의 표현

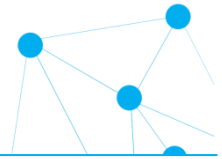


- 노드 구조를 이용한 구현
  - 값을 저장하는 데이터필드와 자식 노드를 가리키는 링크필드로 구성
  - 자식의 개수가 다르므로 노드마다 길이가 다르게 구현된다.



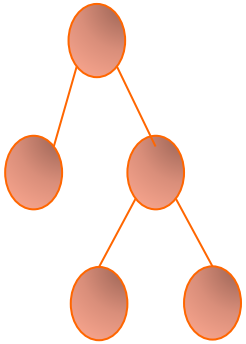
06-2. 이진트리

트리의 종류

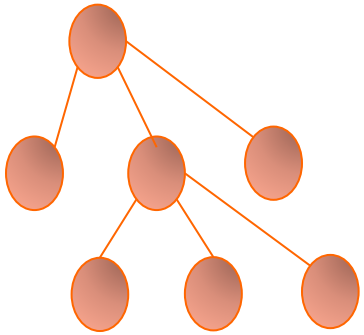


트리

- 이진트리



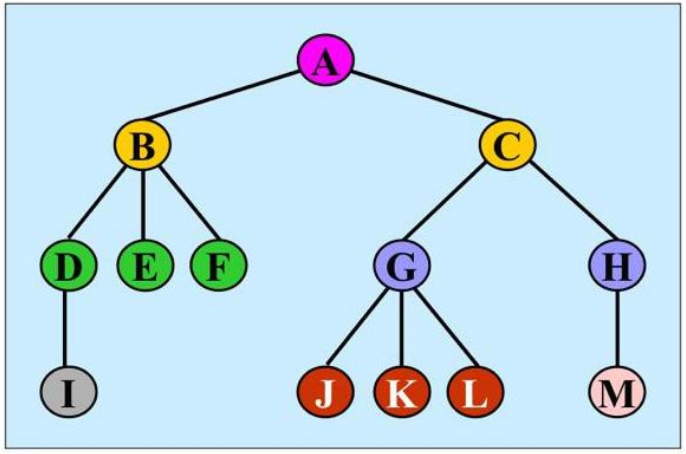
- 일반트리



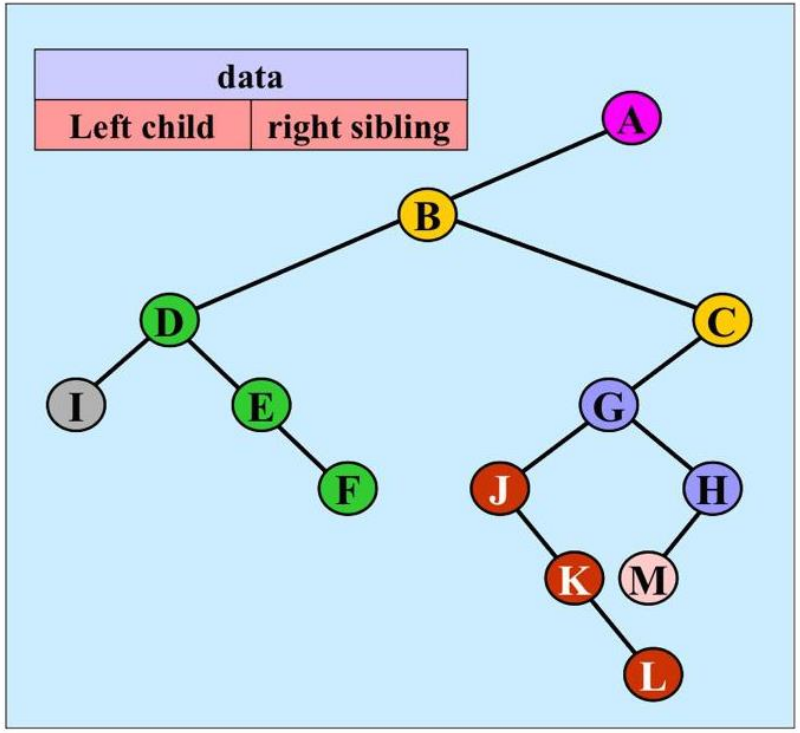
이진트리 변경

- 왼쪽노드는 왼쪽 첫번째 자식노드
- 오른쪽노드는 왼쪽 첫번째 형제노드

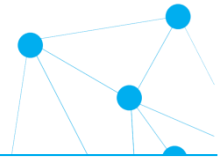
• 3차 트리



• 대응되는 이진 트리

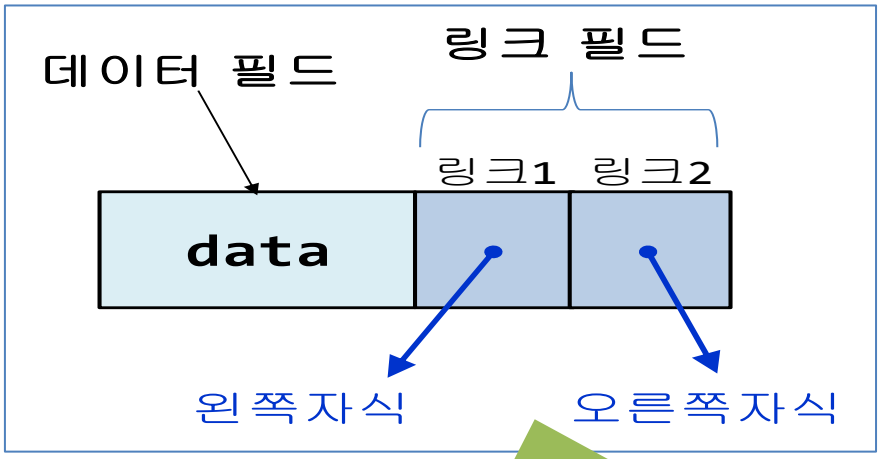


이진트리(binary tree)



- 이진트리 : 모든 노드가 2개의 서브트리를 가지고 있는 트리
  - 서브트리는 공집합일 수 있다.
  - 각 노드에는 최대 2개까지의 자식 노드가 존재
  - 모든 노드의 차수가 2 이하가 된다 → 구현하기가 편리함
  - 서브 트리간의 순서가 존재 (왼쪽, 오른쪽)

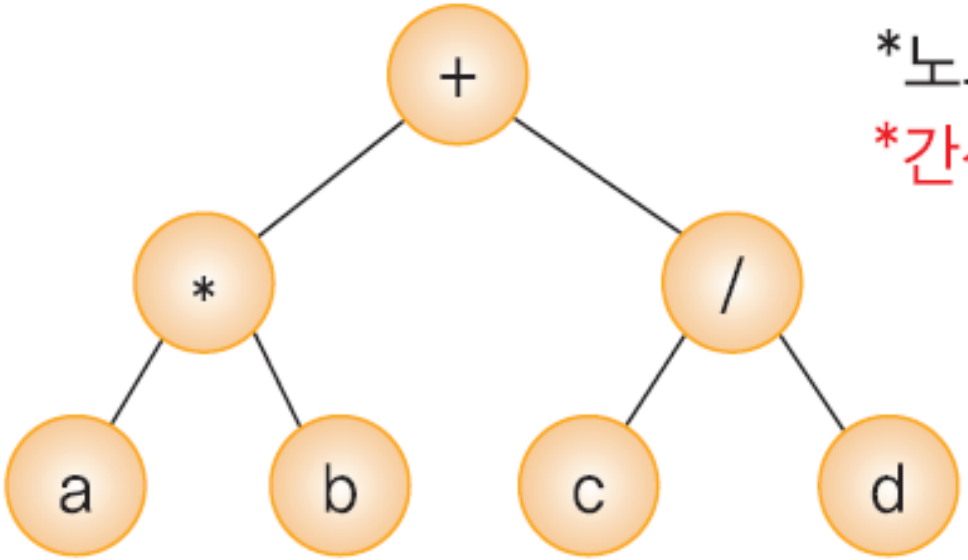
이진트리의 표현



[정의] 이진 트리는 공집합이거나
루트와 왼쪽 서브 트리, 오른쪽 서브 트리로 구성된 노드들의 유한 집합으로 정의된다. 이진 트리의 서브 트리들은 모두 이진 트리여야 한다.

이진트리의 성질

- 노드의 개수가 n개이면 간선의 개수는 n-1

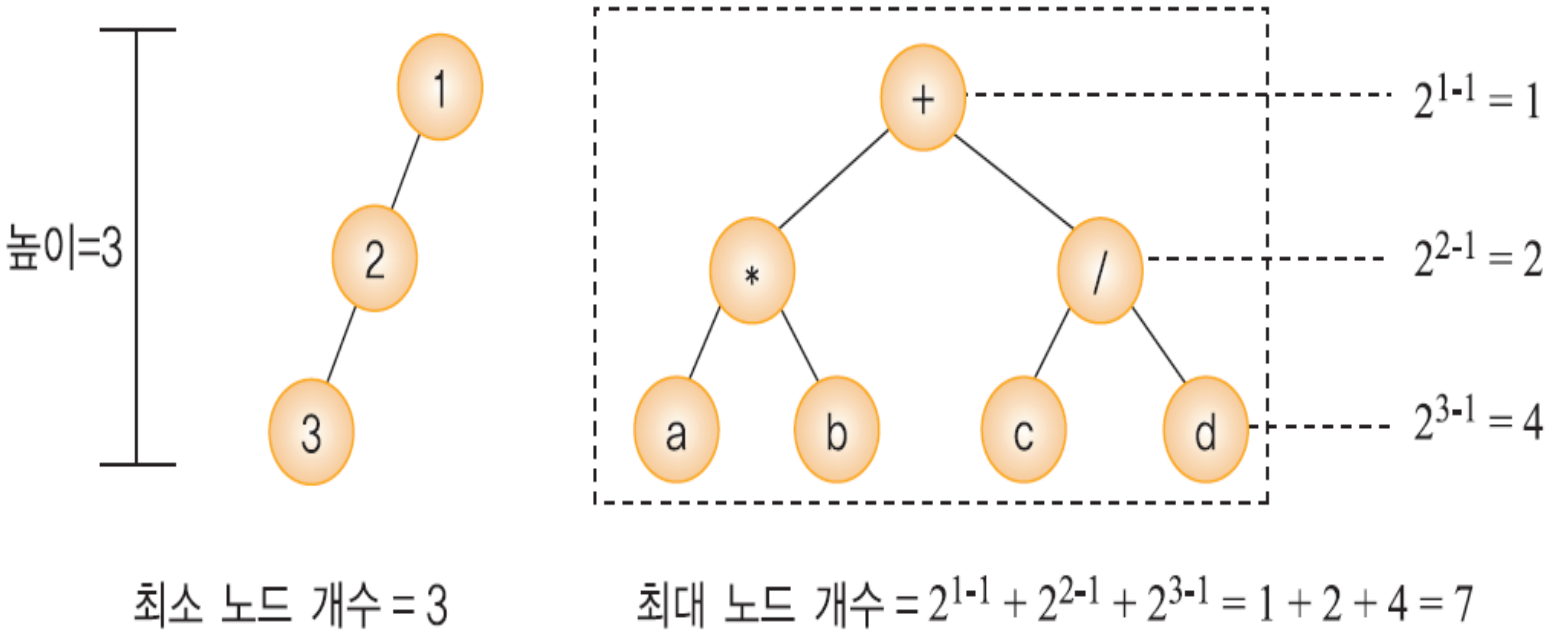


*노드의 개수: 7

*간선의 개수: 6

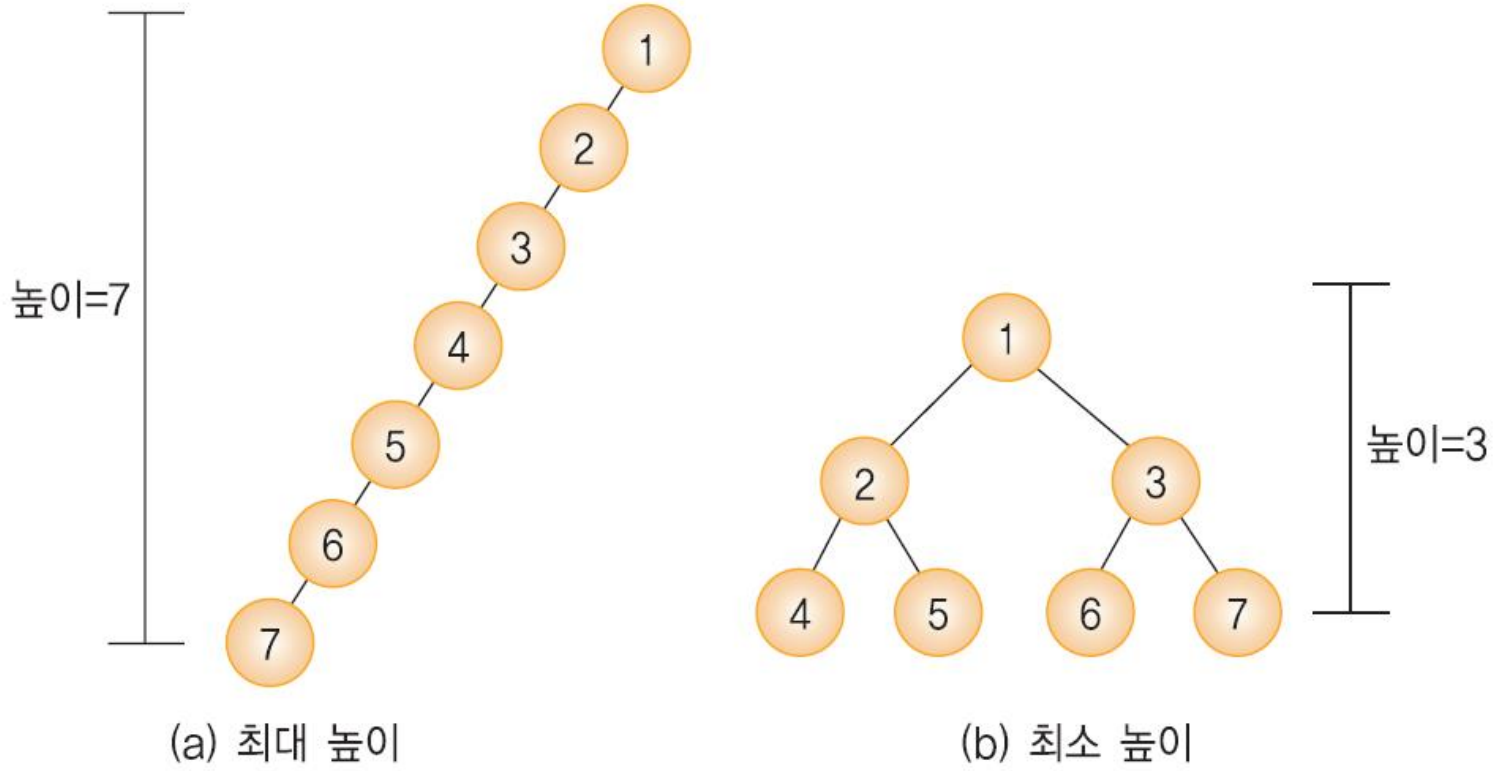
이진트리의 성질

- 높이 h : 최소 h개 ~ 최대 2^h-1개의 노드를 가짐



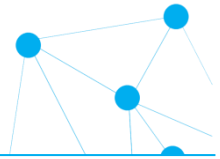
이진트리의 성질

- n개 노드의 이진트리 높이: 최소 $\lceil log_2(n+1) \rceil$ ~ 최대 n

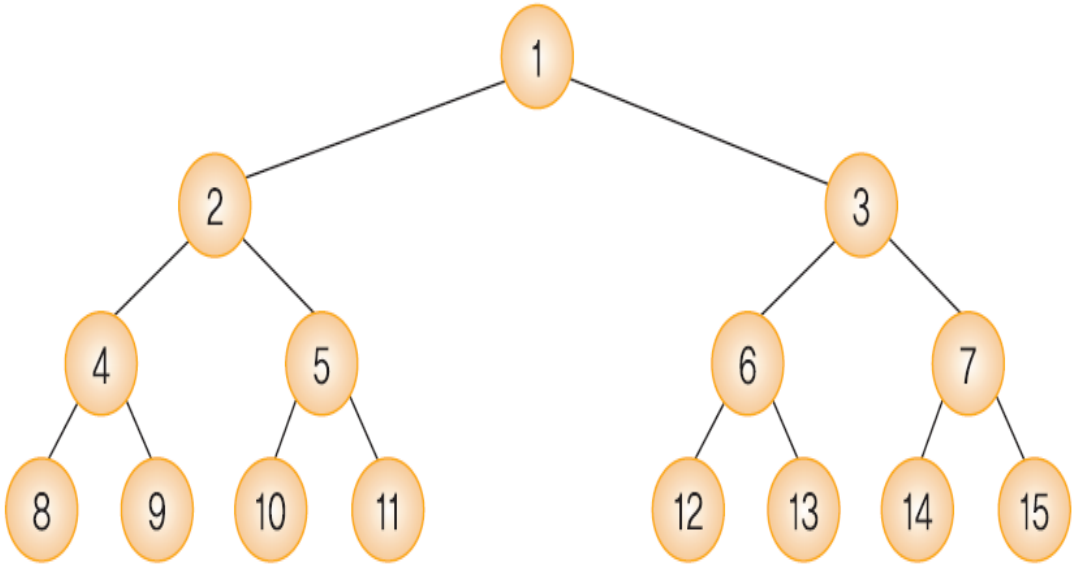


(a) 최대 높이 (b) 최소 높이

이진트리의 분류

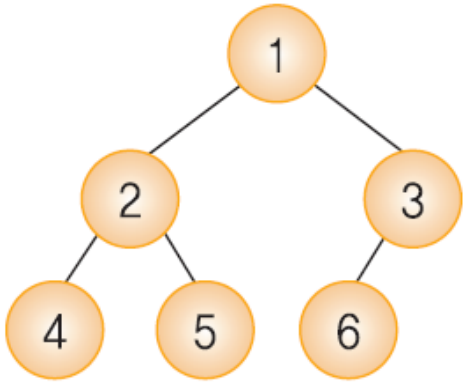


- 포화 이진트리(full binary tree)
  - 트리의 각 레벨에 노드가 꽉 차있는 이진트리
  - 노드의 번호

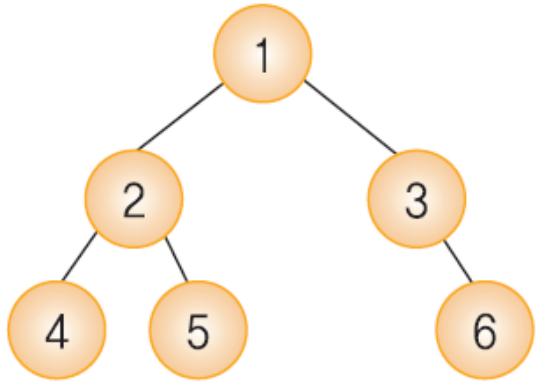


이진트리의 분류

- 완전 이진트리(complete binary tree)
  - 높이가 h일 때 레벨 1부터 h-1까지는 노드가 모두 채워짐
  - 마지막 레벨 h에서는 왼쪽부터 오른쪽으로 노드가 순서대로 채워져 있는 이진 트리



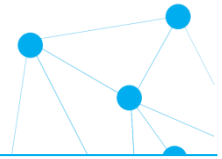
(a) 완전 이진 트리



(b) 완전 이진 트리가 아님

모든 포화 이진트리는 완전 이진트리다.

이진트리의 표현

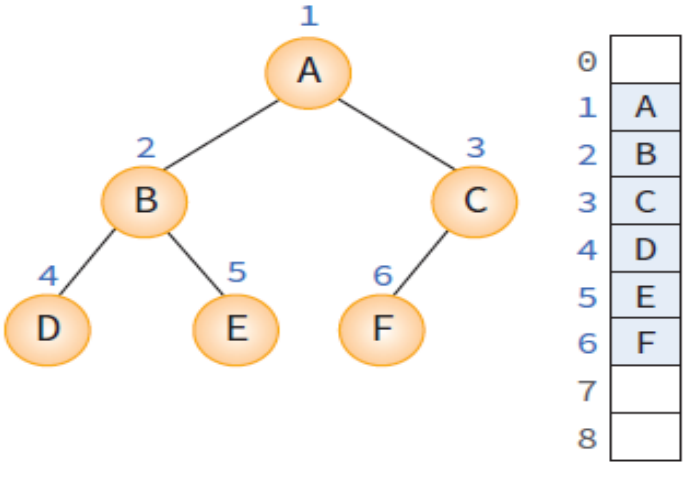


- 배열을 이용하는 방법
  - 포화 또는 완전 이진트리 저장에 주로 사용

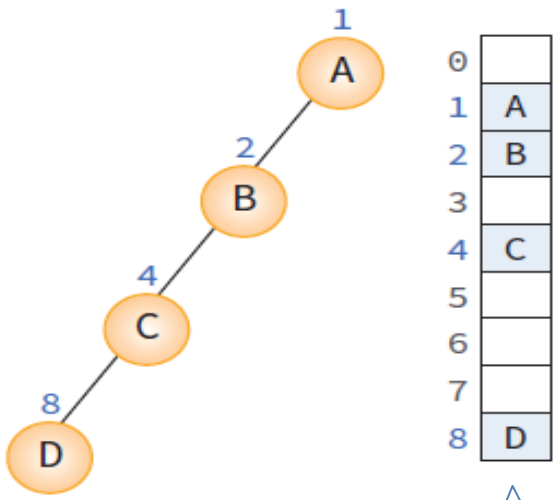
- 포인터(링크)를 이용하는 방법
  - 연결리스트처럼 주소를 이용하여 연결하는 방법
  - 연결리스트는 다음 노드 만을 가리키는 1차원 구조라면 링크표현 방법은 자식 노드들을 연결하는 2차원적인 연결 구조

이진트리의 표현 : 배열표현법

- 모든 이진트리를 포화 이진트리라고 가정한 뒤 각 노드에 번호를 붙여서 그 번호를 배열의 인덱스로 삼아 데이터를 저장하는 방법
  - 편의상 배열의 첫번째 요소는 사용하지 않는다.



(a) 완전 이진트리

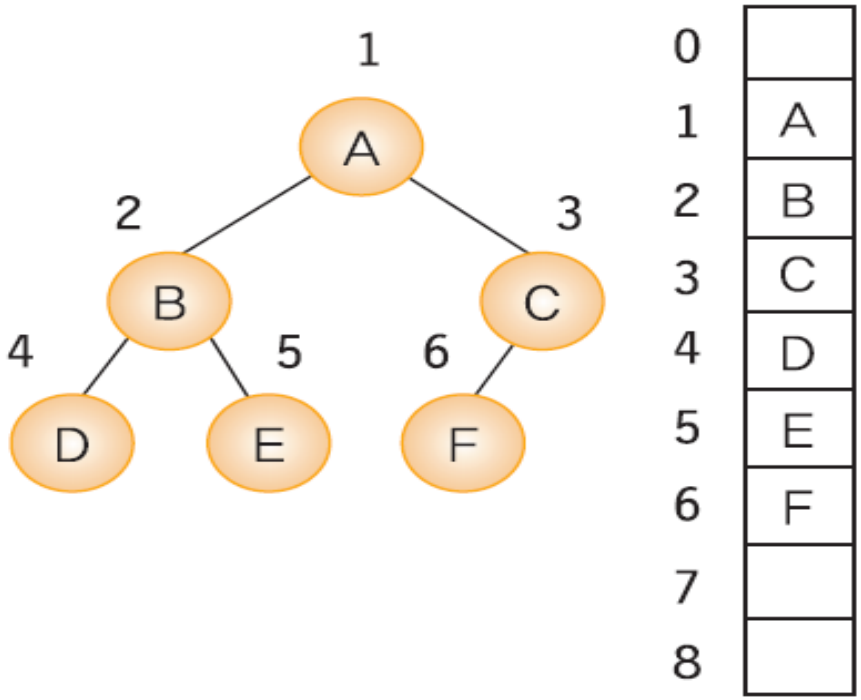


(b) 경사 이진트리

메모리 낭비가 심하다.

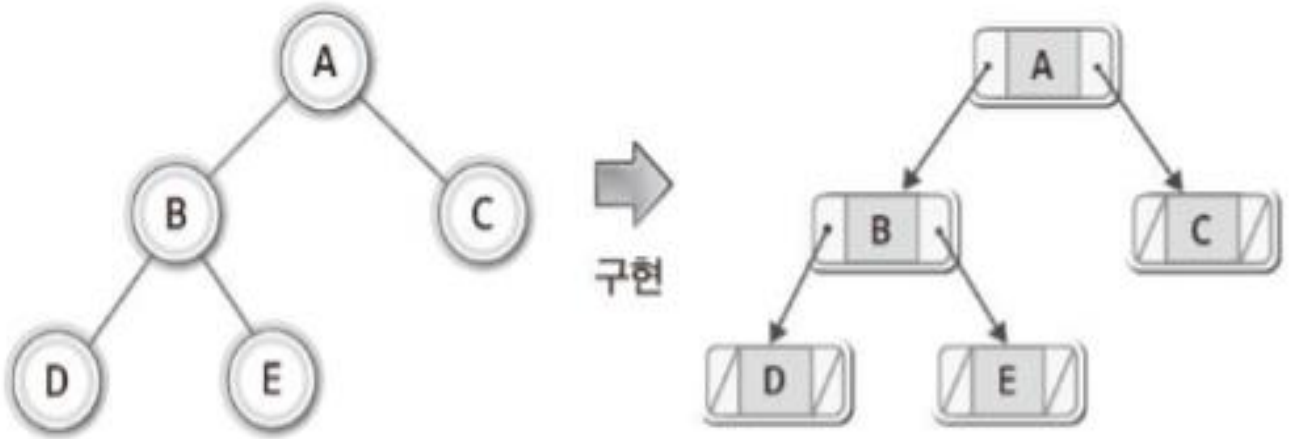
배열표현에서 부모와 자식 인덱스 관계

- 노드 i의 부모 노드 인덱스 = i/2
- 노드 i의 왼쪽 자식 노드 인덱스 = 2i
- 노드 i의 오른쪽 자식 노드 인덱스 = 2i+1

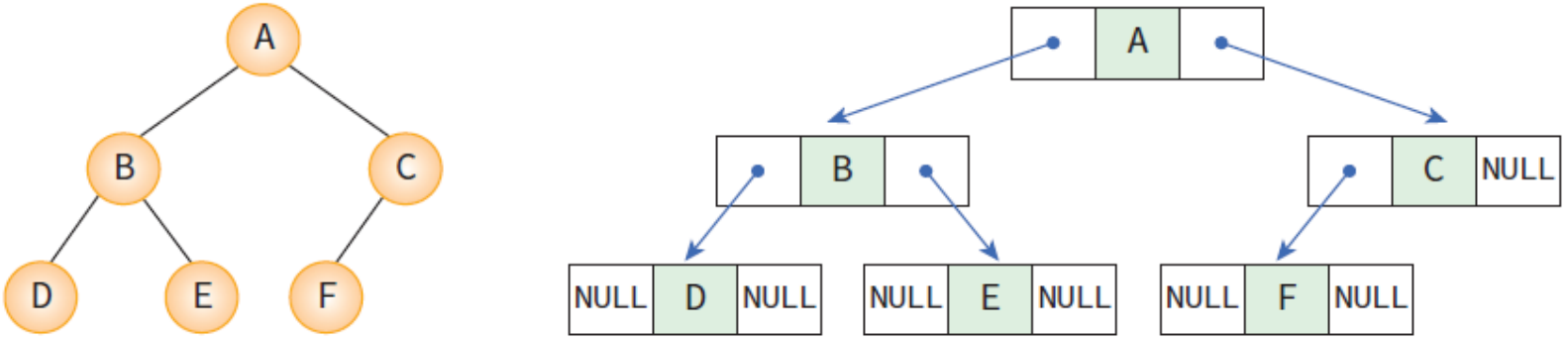


이진트리의 표현: 링크표현법

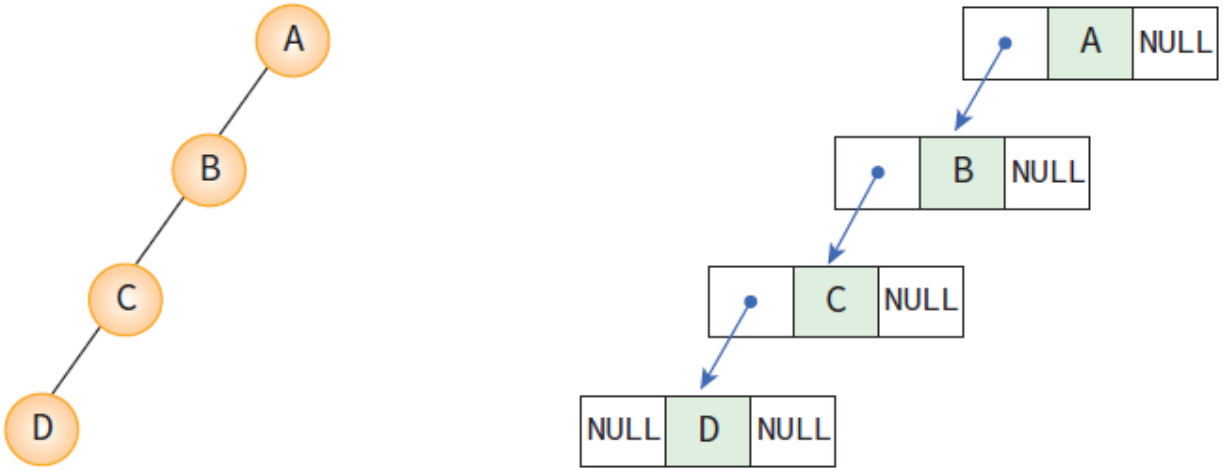
- 포인터를 이용하여 부모노드가 자식노드를 가리키게 하는 방법



이진트리의 표현: 링크표현법



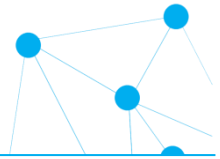
(a) 완전 이진트리



(b) 경사 이진트리

06-3. 이진트리순회

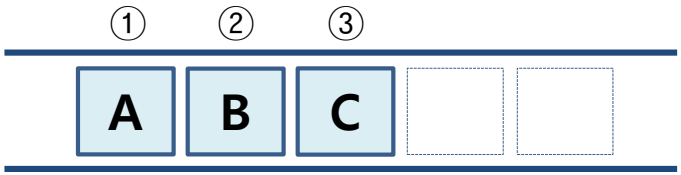
이진트리의 순회



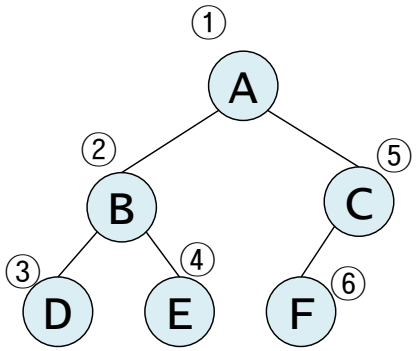
- 순회(traversal)
  - 트리에 속하는 모든 노드를 한 번씩 방문하여 노드가 가지고 있는 데이터를 목적에 맞게 처리하는 것
  - 트리의 노드들을 체계적으로 방문하는 것
  - 트리도 자료 구조이고 자료를 저장하기 위한 목적으로 사용하므로 트리의 노드에 저장된 데이터를 처리하기 위해서는 트리가 가지고 있는 데이터를 적절한 순서로 순회할 필요가 있다.

이진트리의 순회

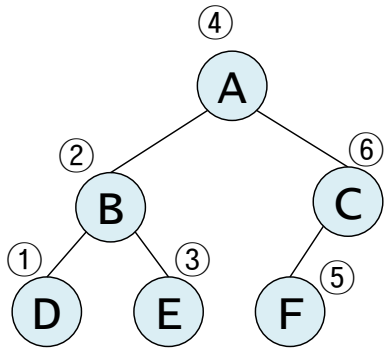
- 선형 자료구조(큐)에서의 순회
  - 하나의 방법만 존재함



- 이진 트리에서의 순회
  - 다양한 순회 방법이 존재함 (비선형 자료구조)



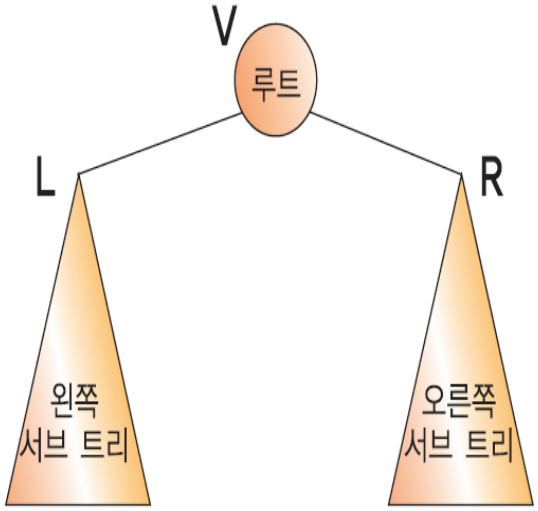
순회방법 1



순회방법 2

이진트리의 기본 순회

- 루트 노드를 언제 방문하느냐에 따라 구분
- 전위순회(preorder traversal) : VLR
  - 루트→왼쪽 자식→오른쪽 자식

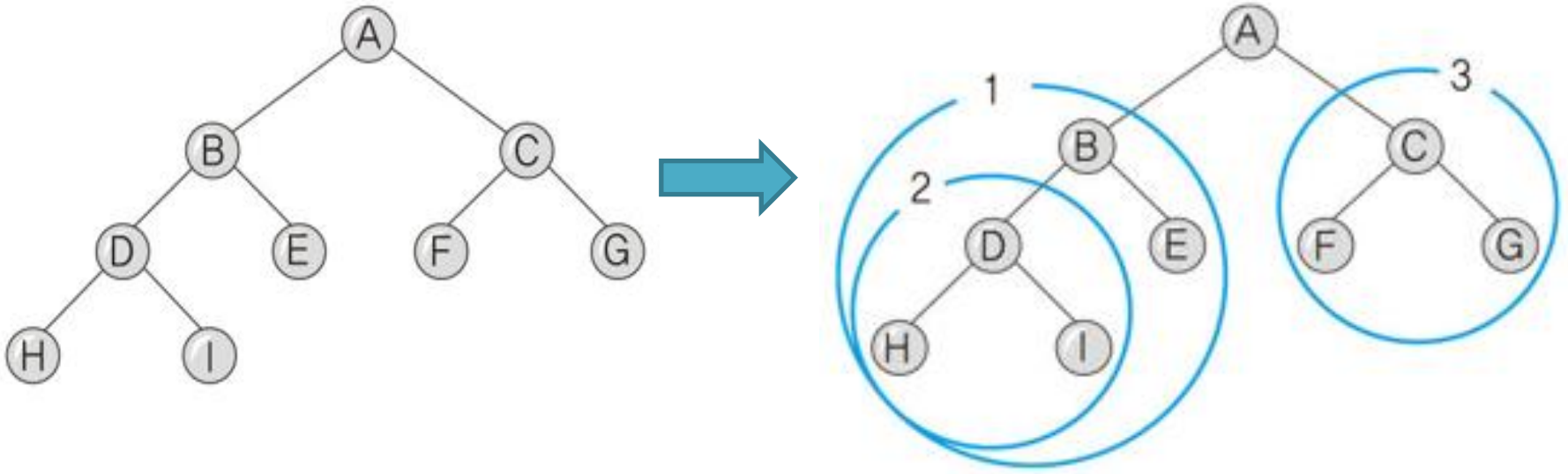


- 중위순회(inorder traversal) : LVR
  - 왼쪽 자식→루트 →오른쪽 자식

- 후위순회(postorder traversal) : LRV
  - 왼쪽 자식→오른쪽 자식→루트

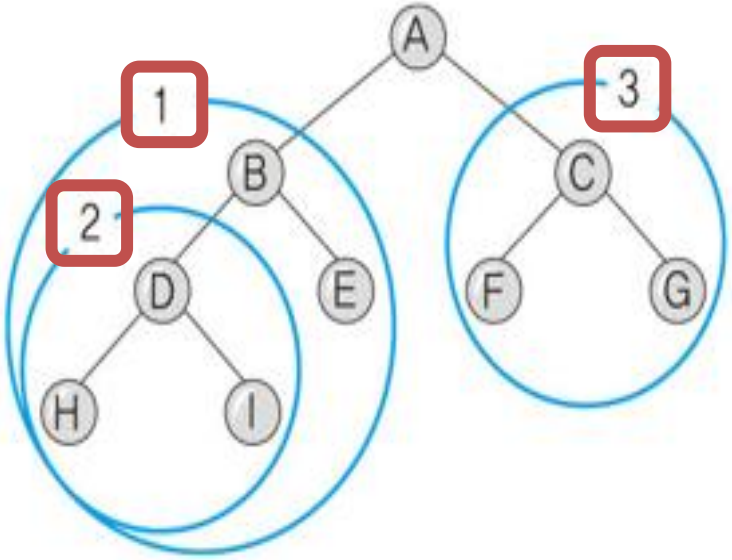
이진트리의 순회

① 서브 트리를 하나의 노드로 생각할 수 있도록 서브 트리 단위로 묶는다.



② 전위, 중위, 후위 각각의 방법으로 순회한다.

전위순회



- Root → Left → Right 순서

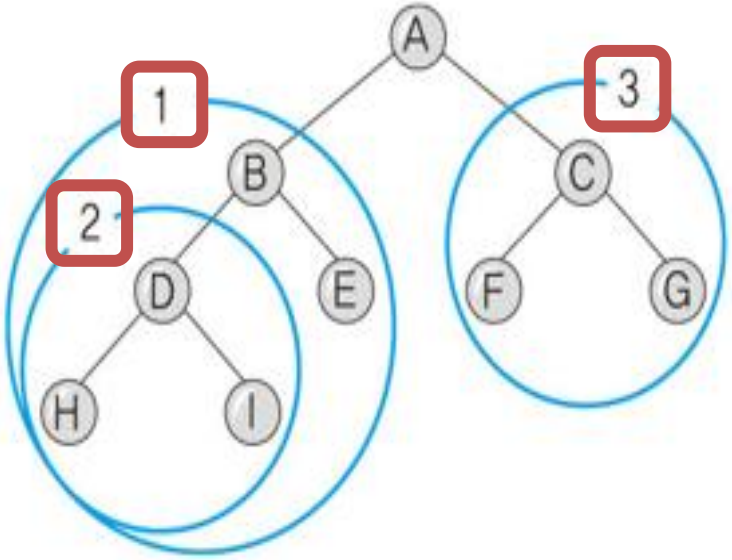
① **A 1 3** 순으로 preorder
② **1** 은 **B2E** 순으로 preorder
③ 따라서 **AB2E3**

④ **2**는 **DHI** 순으로 preorder
⑤ 따라서 **ABDHIE3**

⑥ **3**은 **CFG** 순으로 preorder
⑦ 따라서 **ABDHIECFG**

- 결국 **ABDHIECFG** 순으로 전위 순회

중위순회



- Left → Root → Right 순서

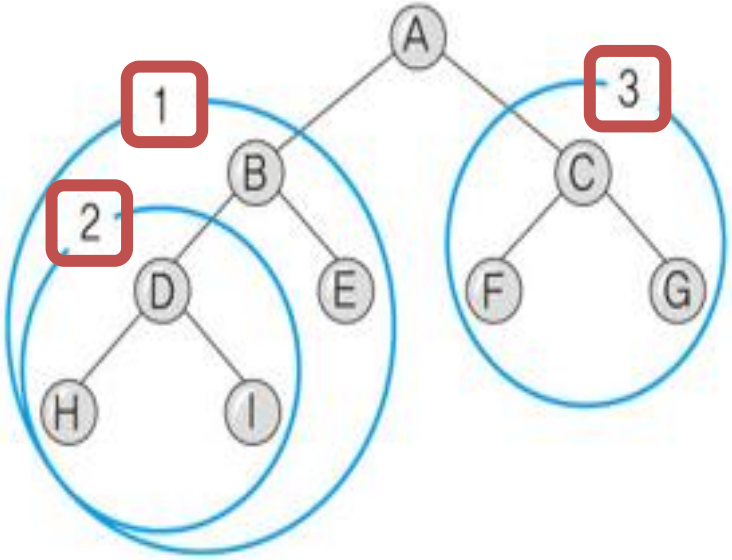
① **1 A 3** 순으로 inorder
② **1** 은 **2BE** 순으로 inorder
③ 따라서 **2BEA3**

④ **2**는 **HDI** 순으로 inorder
⑤ 따라서 **HDIBEA3**

⑥ **3**은 **FCG** 순으로 inorder
⑦ 따라서 **HDIBEAFCG**

- 결국 **HDIBEAFCG** 순으로 중위 순회

후위순회



- Left → Right → Root 순서

① **1 3 A** 순으로 postorder
② **1** 은 **2EB** 순으로 postorder
③ 따라서 **2EB3A**

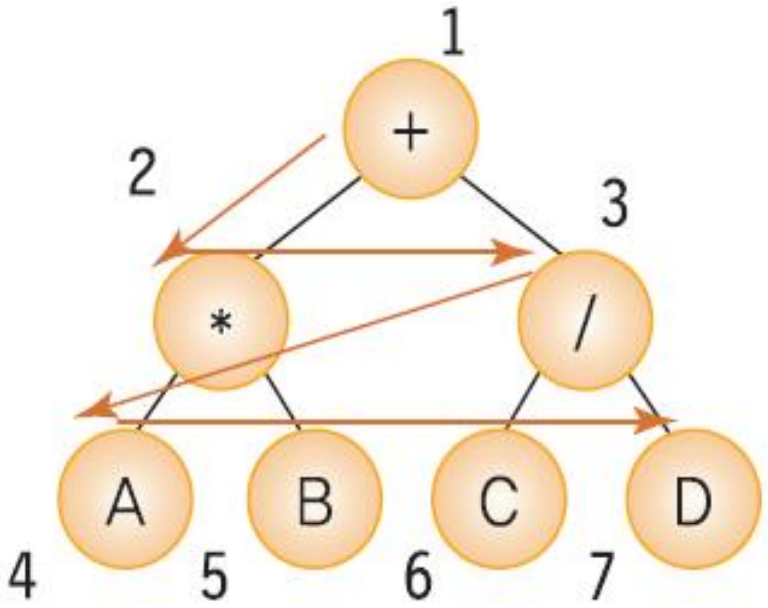
④ **2**는 **HID** 순으로 postorder
⑤ 따라서 **HIDEB3A**

⑥ **3**은 **FGC** 순으로 postorder
⑦ 따라서 **HIDEBFGCA**

- 결국 **HIDEBFGCA** 순으로 후위순회

레벨순회

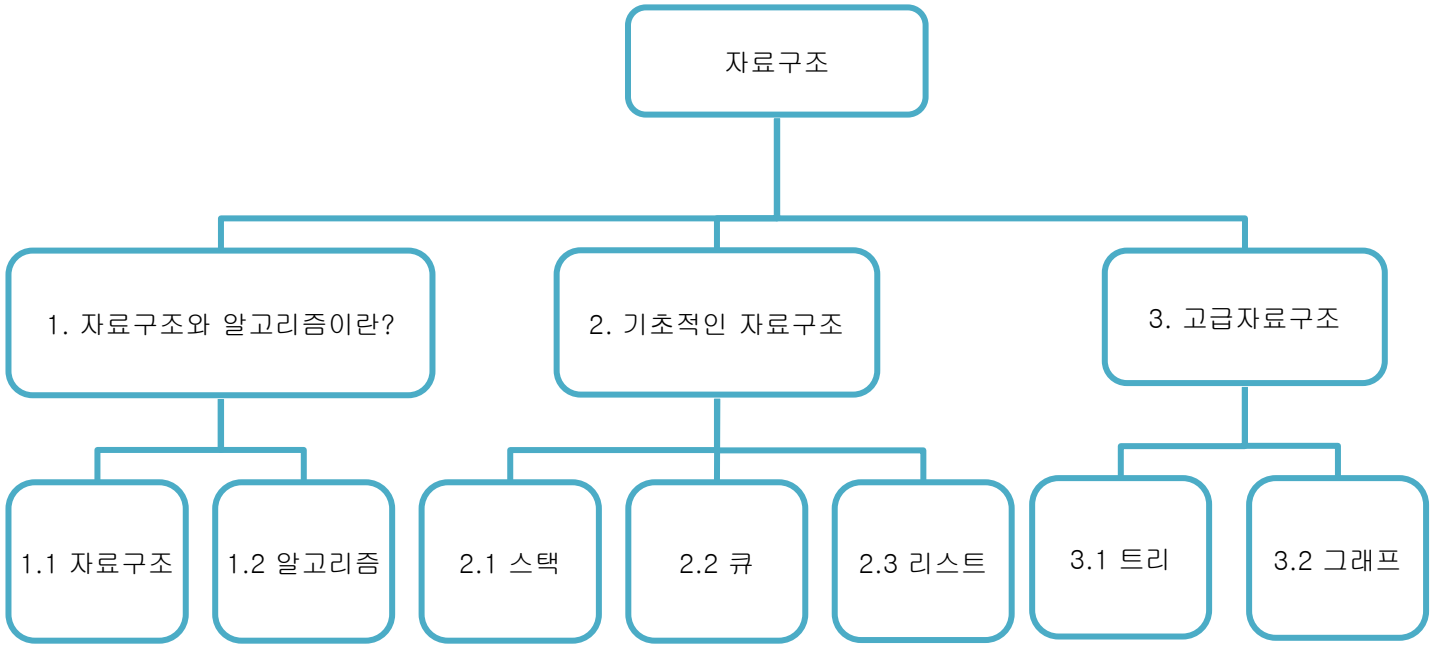
- 각 노드를 레벨 순으로 검사하는 순회 방법



06-4. 트리응용

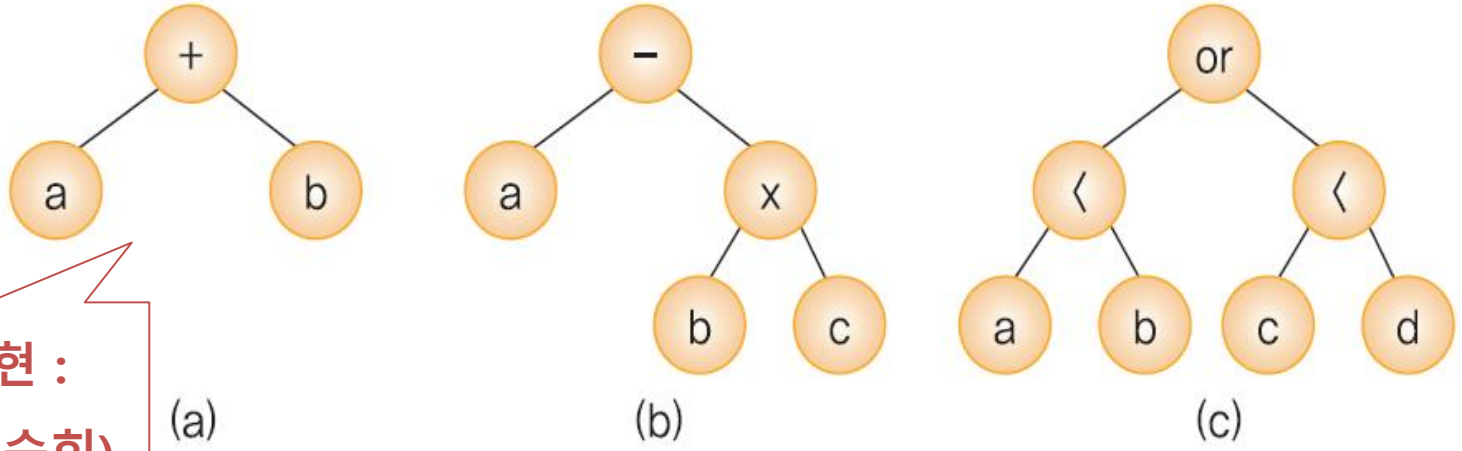
트리 응용

- 전위 순회 응용 : 구조화된 문서 출력
  - 제목 출력 후 소제목 출력



트리 응용

- 중위 순회 응용 : 수식트리 표현
  - 산술식을 트리형태로 표현한 것
    - 비단말노드 : 연산자(operator)
    - 단말노드 : 피연산자(operand)

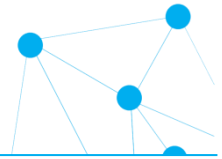


(a) (b) (c)

산술식 표현 :

a+b (중위순회)

트리 응용



- 수식트리 구성방법
  - 후위식으로 변경 후 스택을 이용해 트리로 표현

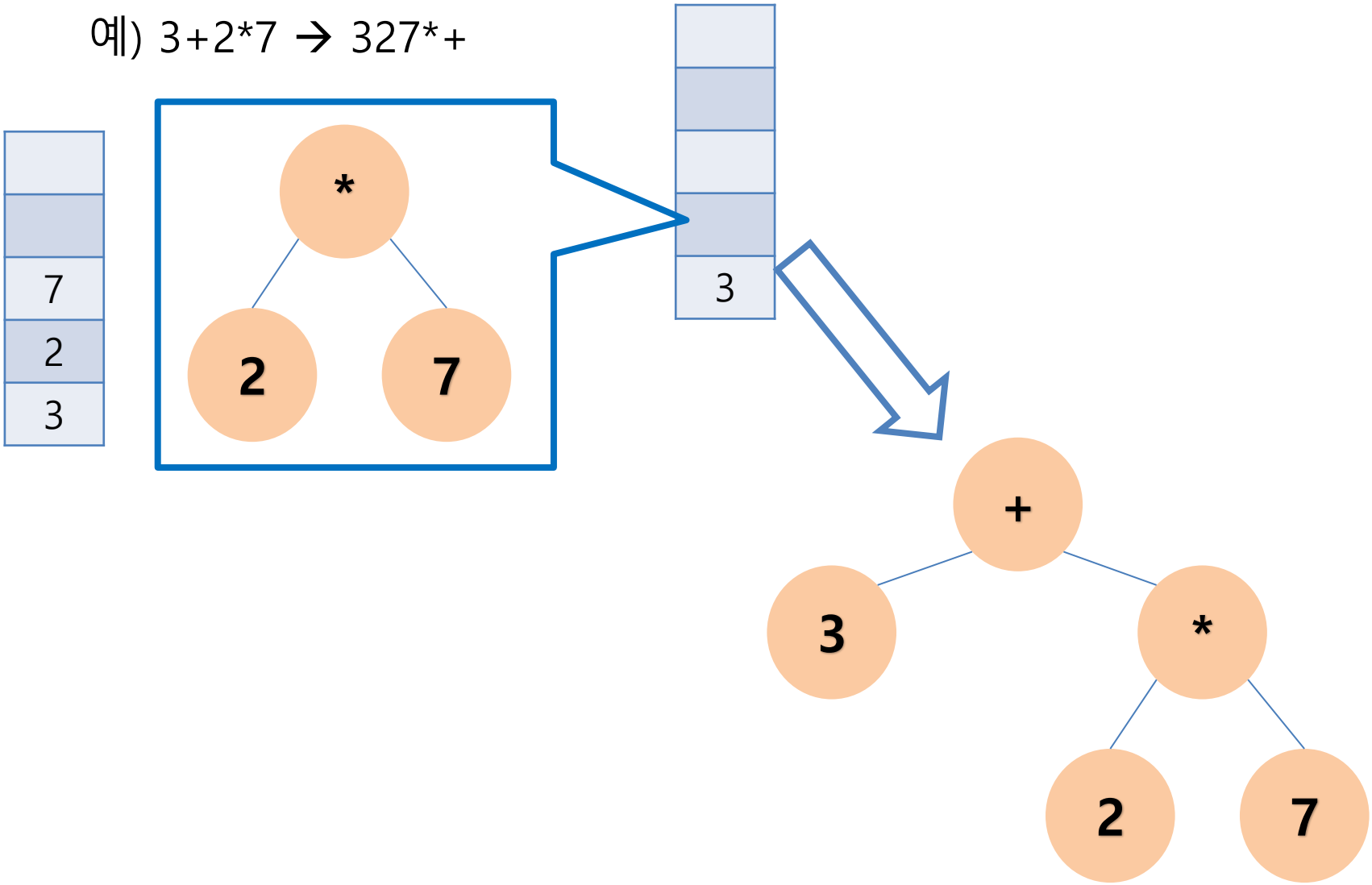
예) 3+2*7 → 327*+

<과정>

- 피연산자를 연산자가 나올 때까지 차례대로 스택에 넣는다.
- 연산자 만나면 스택에서 두 개의 피연산자를 꺼내어 트리 구성
- 순서대로 첫번째→오른쪽자식노드, 두번째 → 왼쪽자식노드로 구성한다.
- 완성된 서브트리를 다시 스택에 넣는다.
- 같은 과정을 식이 끝날 때까지 반복한다.

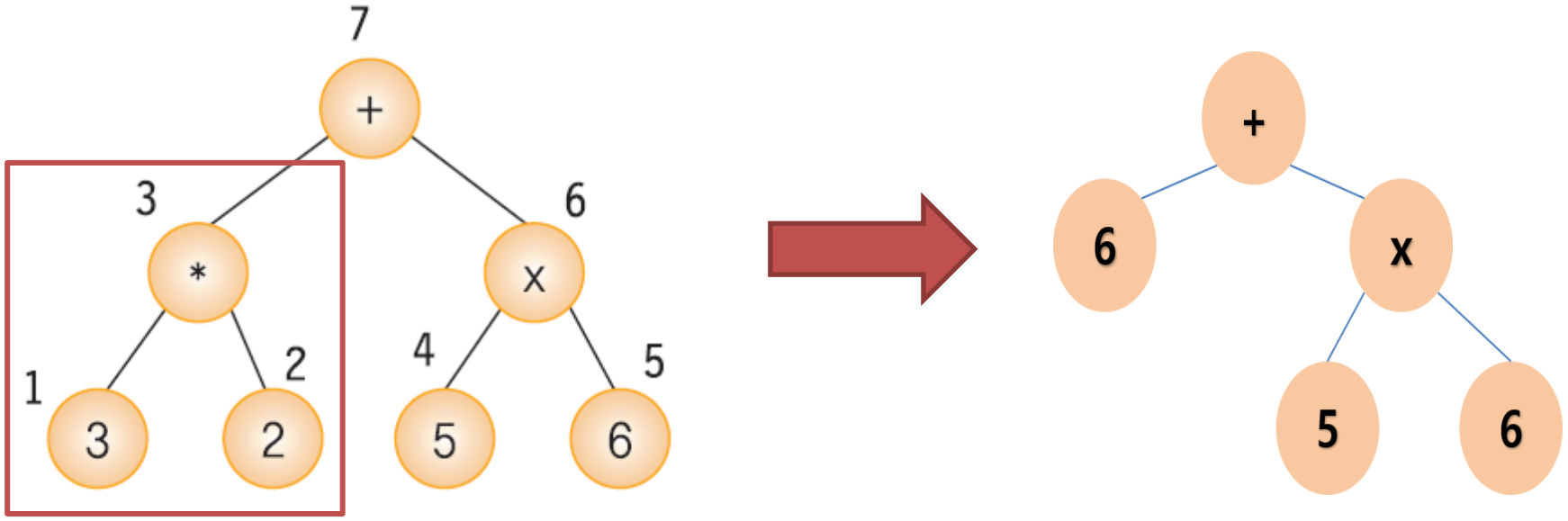
트리 응용

예) 3+2*7 → 327*+

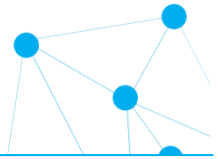


트리 응용

- 후위 순회 응용 : 수식트리 처리
  - 수식 트리의 루트 노드는 연산자이고 피연산자가 자식 노드므로 자식 노드들을 계산하면 루트에 대한 연산 결과를 계산할 수 있다. 자식을 먼저 연산, 즉 후위순회 사용, 연산의 결과를 루트에 저장



트리 응용



- 후위 순회 응용 : 디렉토리 용량 계산
  - 디렉토리의 용량은 하위 디렉토리 용량을 알아야 계산할 수 있다.