비선형 자료구조

01 트리
02 힙
03 그래프
04 트라이

신 제 용

트리 (Tree)

01 트리 이론

02 트리의 구현

신 제 용

01 트리 이론

계층이 있는 비선형 자료구조인 트리 자료구조를 배우고,
트리에서 자료를 탐색하는 방법을 학습합니다.

학습 키워드 - 트리, 비선형, 깊이우선탐색, 너비우선탐색

Chapter 01
트리 이론

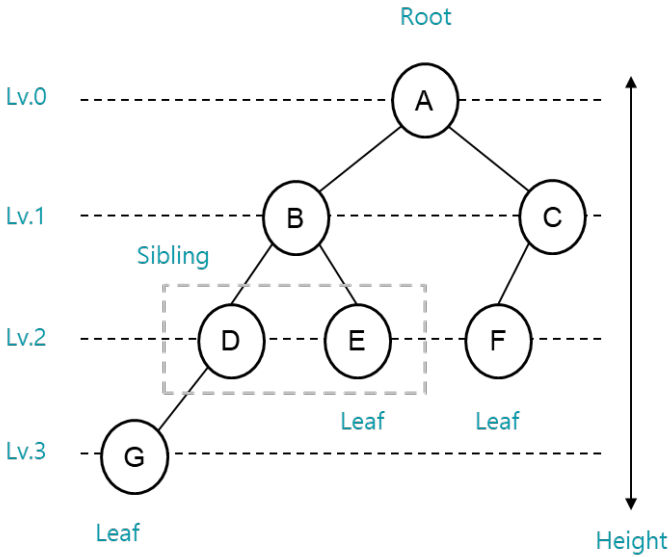
ZEROBASE SCHOOL

트리 (Tree)

- 부모와 자식 관계로 이어진 노드(node)의 집합으로 이루어진 자료구조

- 계층적 구조를 나타낼 때 주로 사용
  → 폴더 구조, 조직도, 가계도, …

트리의 구조



노드(Node): 트리 구조의 자료 값을 담고 있는 단위

에지(Edge): 노드 간의 연결선 (=link, branch)

루트 노드(Root): 부모가 없는 노드, 가장 위의 노드

잎새 노드(Leaf): 자식이 없는 노드 (=단말)

내부 노드(Internal): 잎새 노드를 제외한 모든 노드

부모(Parent): 연결된 두 노드 중 상위의 노드

자식(Child): 연결된 두 노드 중 하위의 노드

형제(Sibling): 같은 부모를 가지는 노드

깊이(Depth): 루트에서 어떤 노드까지의 간선의 수

레벨(Level): 트리의 특정 깊이를 가지는 노드 집합

높이(Height): 트리에서 가장 큰 레벨 값

크기(Size): 자신을 포함한 자식 노드의 개수

차수(Degree): 각 노드가 지닌 가지의 수

트리의 차수: 트리의 최대 차수

트리의 특징

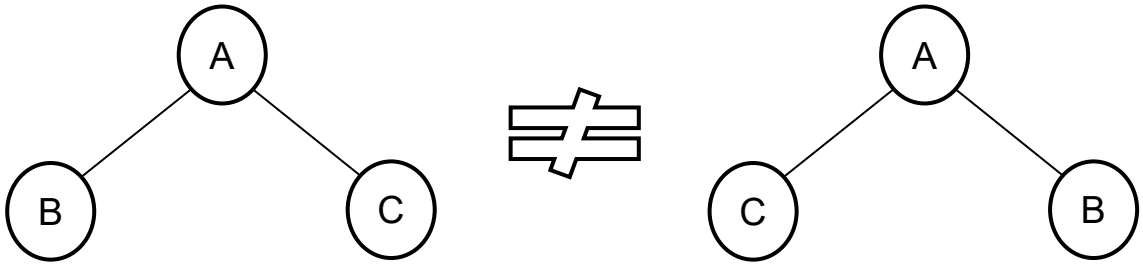
- 하나의 노드에서 다른 노드로 이동하는 경로는 유일
- 노드가 N개인 트리의 간선의 수는 $N - 1$개
- Acyclic (Cycle이 존재하지 않음)
- 모든 노드는 서로 연결되어 있음
- 하나의 간선을 끊으면 2개의 부분 트리(sub-tree)로 분리됨

이진 트리 (Binary tree)

- 각 노드는 최대 2개의 자식을 가질 수 있음
- 자식 노드는 좌우를 구분함

  → 왼쪽 자식: 부모 노드의 왼쪽 아래

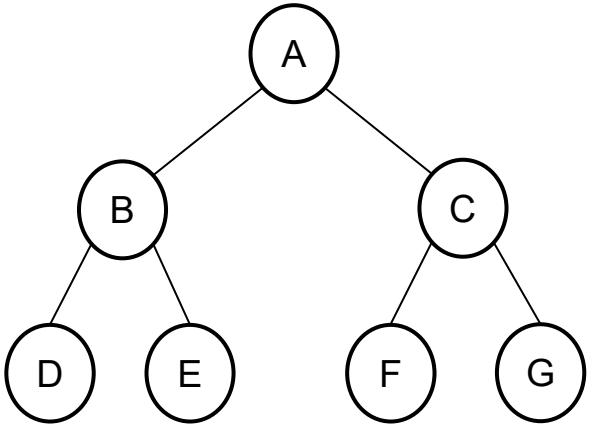
  → 오른쪽 자식: 부모 노드의 오른쪽 아래



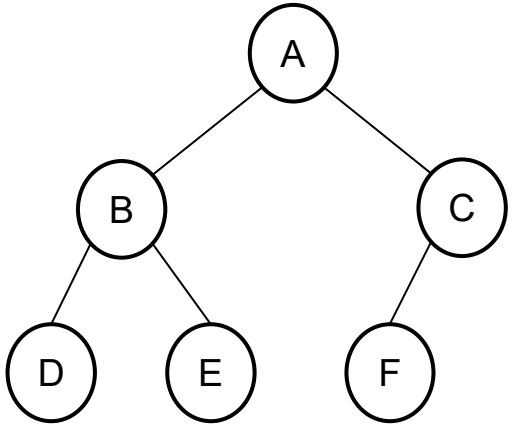
이진 트리의 종류 (1)

- 포화 이진 트리 (Perfect binary tree) - 모든 레벨에서 노드들이 꽉 채워져 있는 트리
- 완전 이진 트리 (Complete binary tree) - 마지막 레벨을 제외하고 노드들이 모두 채워져 있는 트리

< 포화 이진 트리 >



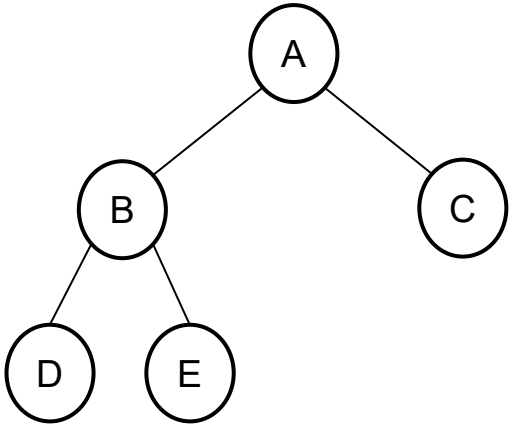
< 완전 이진 트리 >



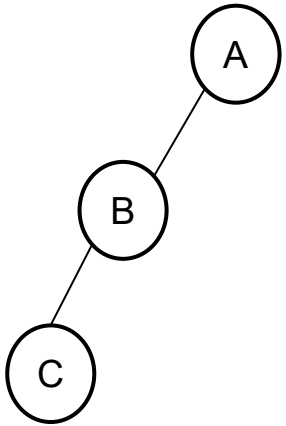
이진 트리의 종류 (2)

- 정 이진 트리 (Full binary tree) - 모든 노드가 0개 또는 2개의 자식 노드를 갖는 트리
- 편향 트리 (Skewed binary tree) - 한쪽으로 기울어진 트리

< 정 이진 트리 >



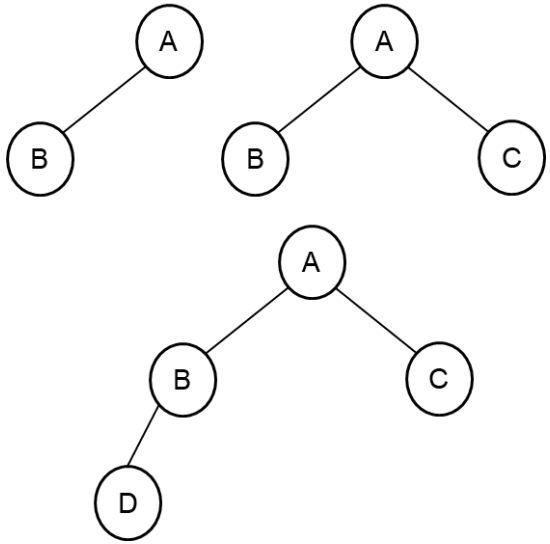
< 편향 트리 >



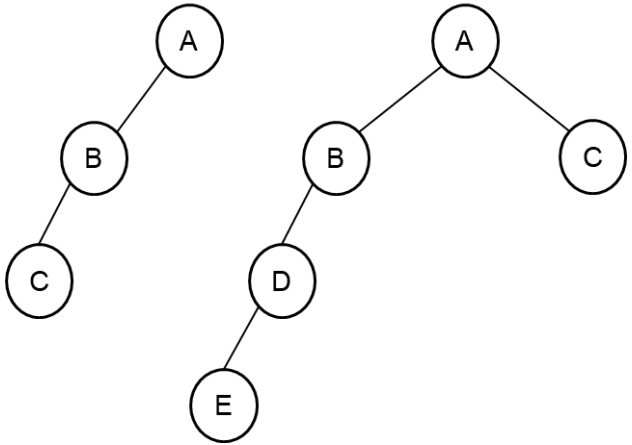
이진 트리의 종류 (3)

- 균형 이진 트리 (Balanced binary tree) 모든 노드의 좌우 서브 트리 높이가 1이상 차이 나지 않는 트리

< 균형 이진 트리 >



< 균형 이진 트리 X >



이진 트리의 특징

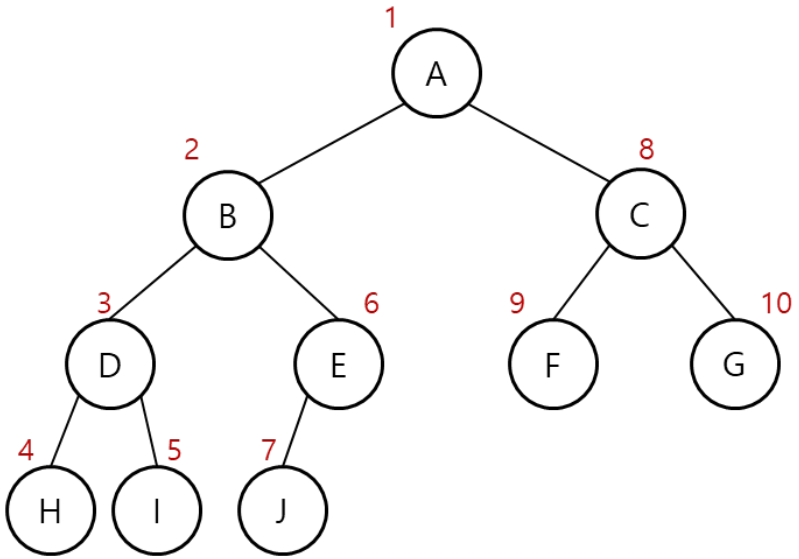
- 포화 이진 트리의 높이가 h일 때, 노드의 수는 $2^{h+1} - 1$개
- 포화(or 완전) 이진 트리의 노드가 N개 일 때, 높이는 $\log_2 N$
- 이진 트리의 노드가 N개 일 때, 최대 가능 높이는 N

이진 트리의 순회 (Traversal)

- 모든 노드를 빠뜨리거나 중복하지 않고 방문하는 연산
- 비선형 자료구조이기 때문에 다양한 순회 방법이 있다.
- 전위 순회, 대칭 순회, 후위 순회 – 재귀적 순회 방법
- 레벨 순회 – 레벨 별 순회 방법

전위 순회 (Preorder traversal)

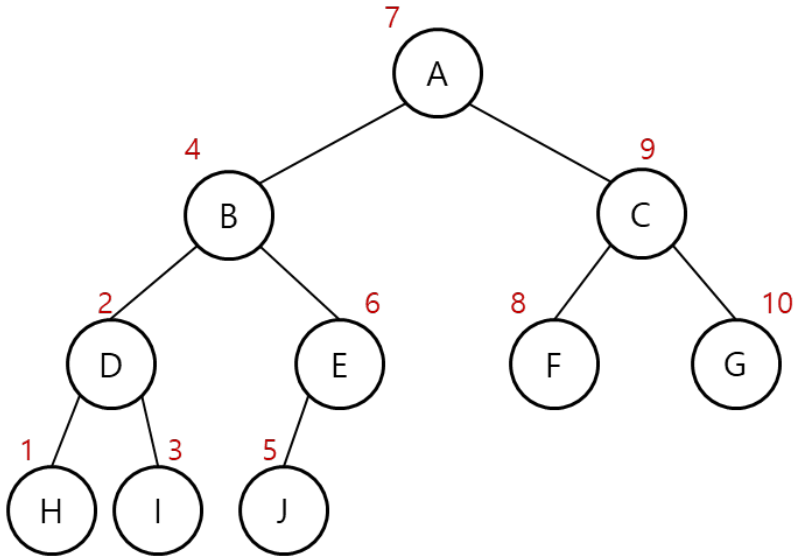
- 깊이 우선 순회(Depth first traversal)이라고도 한다.
- 방문 순서: 현재 노드 → 왼쪽 노드 → 오른쪽 노드



* 순회 경로: A → B → D → H → I → E → J → C → F → G

대칭 순회 (Inorder traversal)

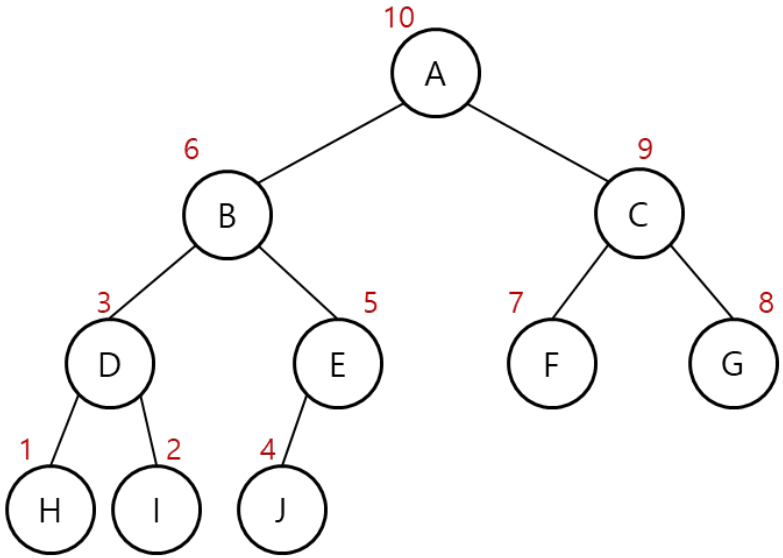
- 방문 순서: 왼쪽 노드 → 현재 노드 → 오른쪽 노드



* 순회 경로: H → D → I → B → J → E → A → F → C → G

후위 순회 (Postorder traversal)

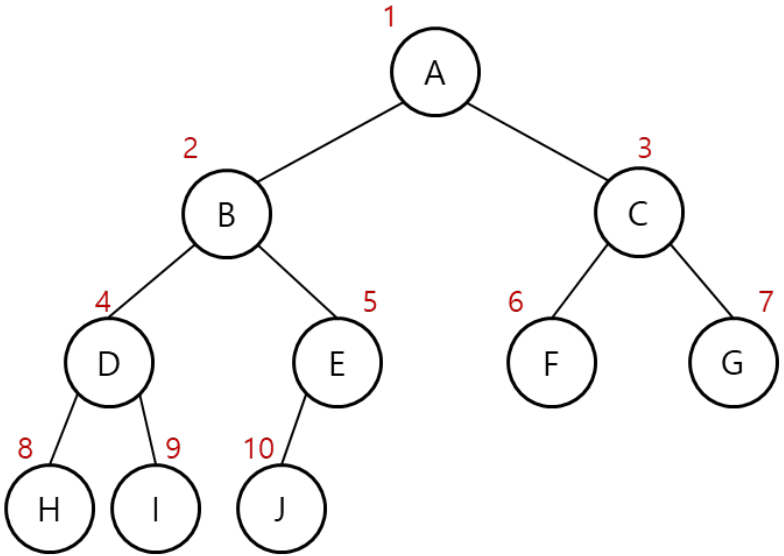
- 방문 순서: 왼쪽 노드 → 오른쪽 노드 → 현재 노드



* 순회 경로: H → I → D → J → E → B → F → G → C → A

레벨 순회 (Level order traversal)

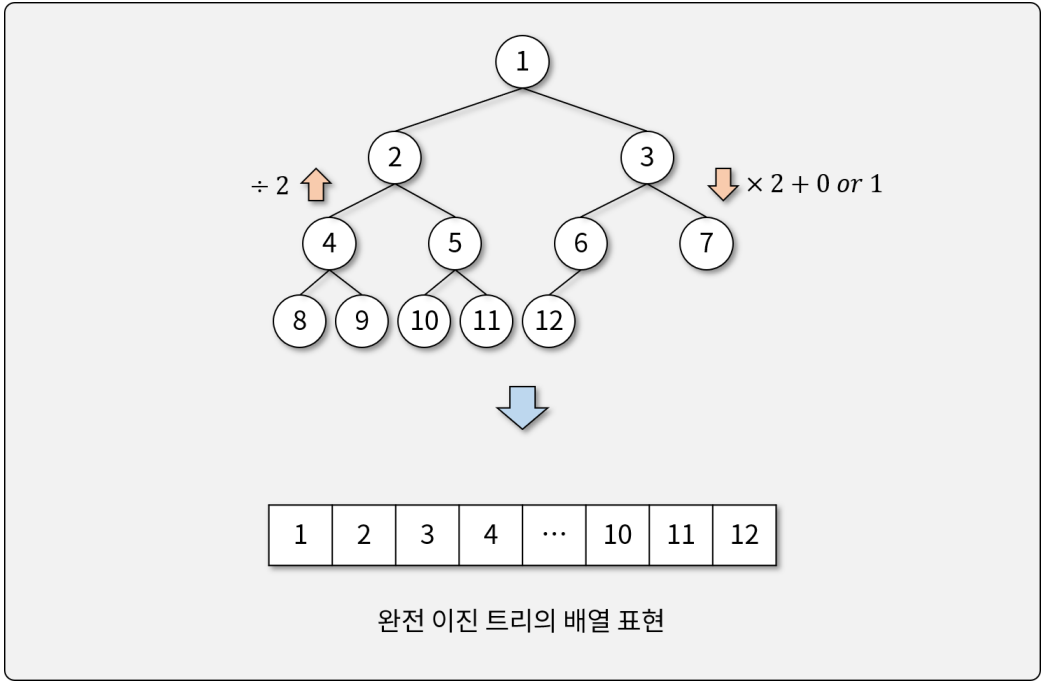
- 너비 우선 순회 (Breadth first traversal)이라고도 한다.
- 방문 순서: 위쪽 레벨 부터 왼쪽 노드 → 오른쪽 노드



* 순회 경로: A → B → C → D → E → F → G → H → I → J

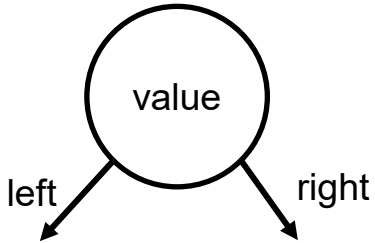
완전 이진 트리의 구현

- 배열에 레벨 순회 순서대로 배열로 구성



일반적인 이진 트리의 구현

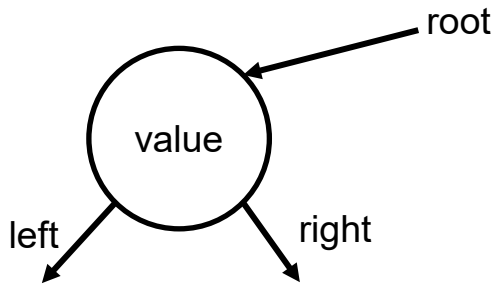
- Node 클래스를 이용하여 구현
  → value, left, right 멤버 변수를 가지는 클래스




```
class Node:
    def __init__ (self, value, left=None, right=None):
        self.value = value
        self.left = left
        self.right = right
```

일반적인 이진 트리의 구현 (2)

- Root node를 멤버 변수로 가지는 클래스 Tree로 구현




```
class Tree:
    def __init__ (self, value):
        self.root = Node(value)
```

02 트리의 구현

다음 챕터에서는 이론적으로 배운 트리를 직접 구현해 봅니다.

Chapter 01
트리 이론

ZEROBASE SCHOOL

02 트리의 구현

트리를 직접 구현하여 자료를 표현하고 탐색하는 과정을
확실하게 학습합니다.

학습 키워드 - 트리, 구현, 재귀, 반복

Chapter 02
트리의 구현

ZEROBASE SCHOOL

```python
# 배열(리스트)를 기반으로 하는 완전이진트리를 구현하시오.

class BinaryTree:
    def __init__(self, data):
        self.data = data

    def preorder(self):
        pass

    def inorder(self):
        pass

    def postorder(self):
        pass

    def levelorder(self):
        pass

    def bfs(self, value):
        return False

    def dfs(self, value):
        return False
```

Chapter 02
트리의 구현

ZEROBASE SCHOOL

```python
# Node를 이용하여 완전 이진 트리를 구현하시오.

class Node:
    def __init__(self, value, left=None, right=None):
        self.value = value
        self.left = left
        self.right = right


class BinaryTree:
    def __init__(self, array):
        node_list = [Node(value) for value in array]
        for ind, node in enumerate(node_list):
            left = 2 * ind + 1
            right = 2 * ind + 2
            if left < len(node_list):
                node.left = node_list[left]
            if right < len(node_list):
                node.right = node_list[right]

        self.root = node_list[0]
```

```python
    def preorder(self):
        pass

    def inorder(self):
        pass

    def postorder(self):
        pass

    def levelorder(self):
        pass

    def bfs(self, value):
        return False

    def dfs(self, value):
        return False
```

Chapter 02
트리의 구현

ZEROBASE SCHOOL