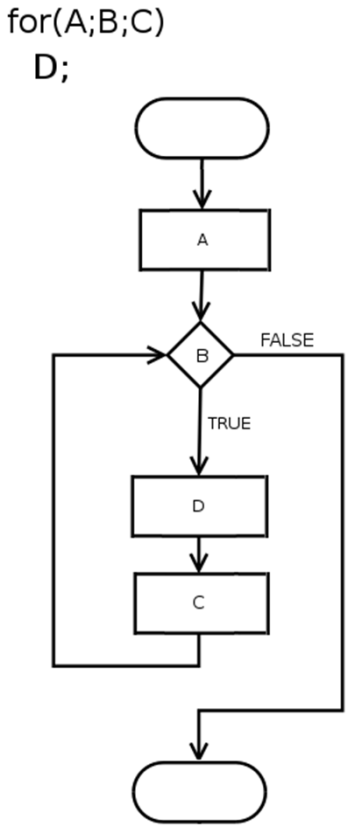
Clickseo.com

자료구조 & 알고리즘

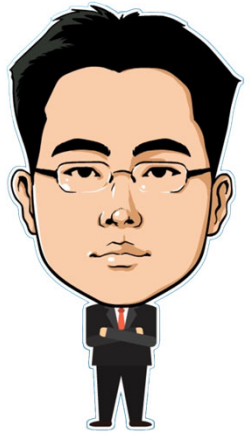


트리
(Tree)

Seo, Doo-Ok

Clickseo.com

clickseo@gmail.com



목 차

- 트리의 이해
- 이진 트리
- 우선 순위 큐와 힙

트리의 이해 (1/3)

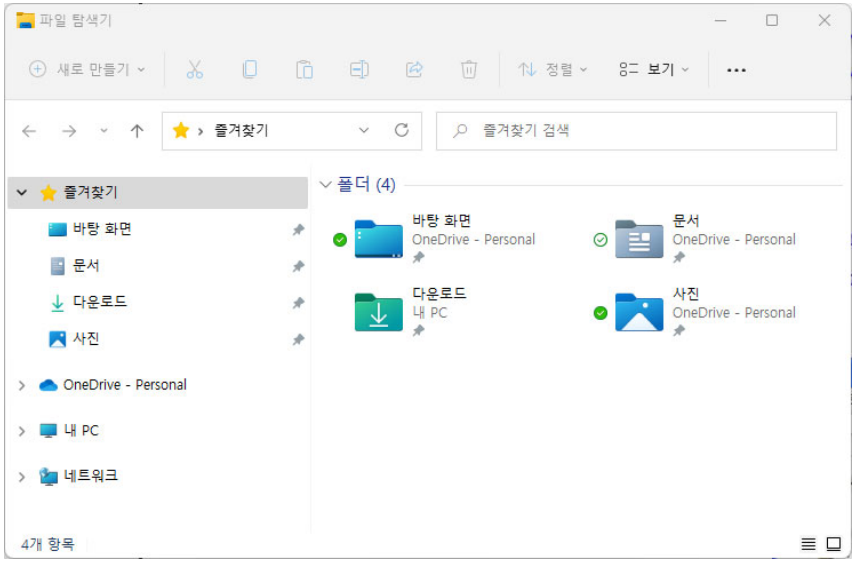
● 트리(Tree)

❍ 트리의 정의

- 원소들 간에 **1:多 관계**를 가지는 **비선형 자료구조**
- 원소들 간에 **계층 관계**를 가지는 **계층형 자료구조**
- 상위 원소에서 하위 원소로 내려가면서 확장되는 **나무 모양의 구조**

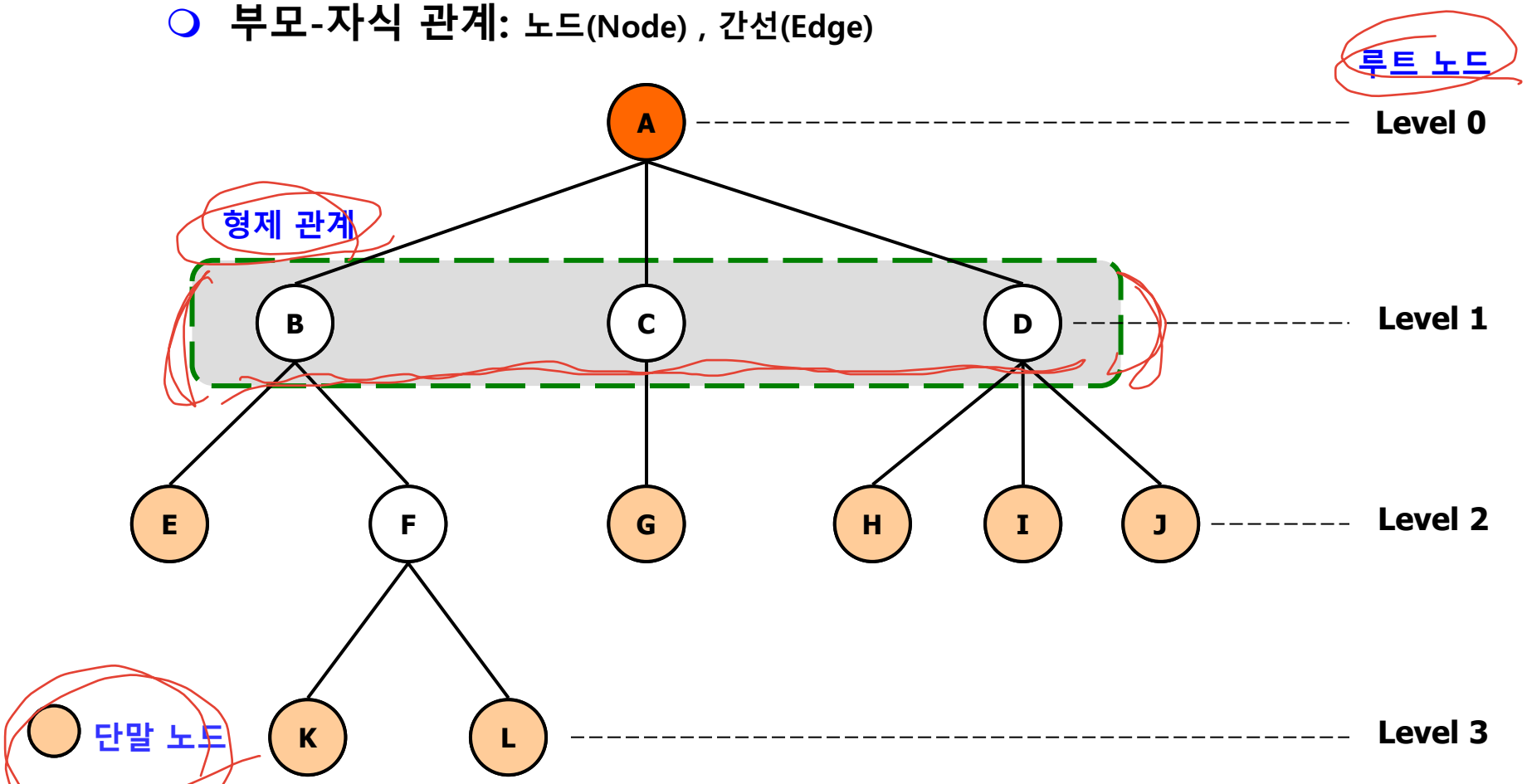
❍ 트리 구조의 예

- 컴퓨터 디렉터리(Directory) 구조
- 기업 구조(Organization Chart)
- 족보(Family Tree)
- 결정 트리(Decision Tree)



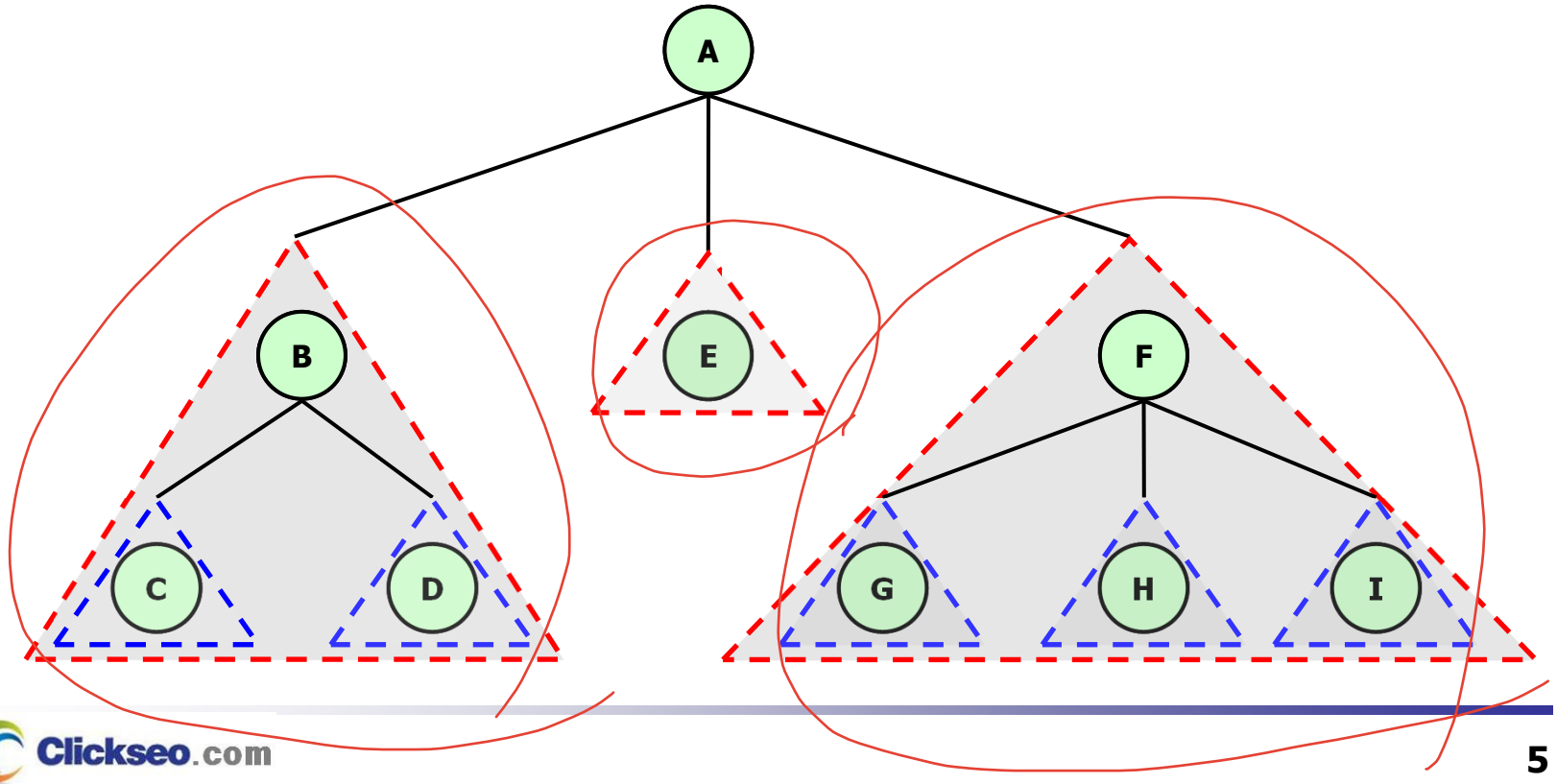
트리의 이해 (2/3)

- **트리 구조**
  - 부모-자식 관계: 노드(Node) , 간선(Edge)



트리의 이해 (3/3)

- **트리 구조: 부분 트리**
  - **부분 트리(Subtree)**
    - 자식 노드들은 각각 독립하여 새로운 트리를 구성할 수 있다.
    - 각 노드는 자식 노드 수만큼의 서브 트리를 갖는다.

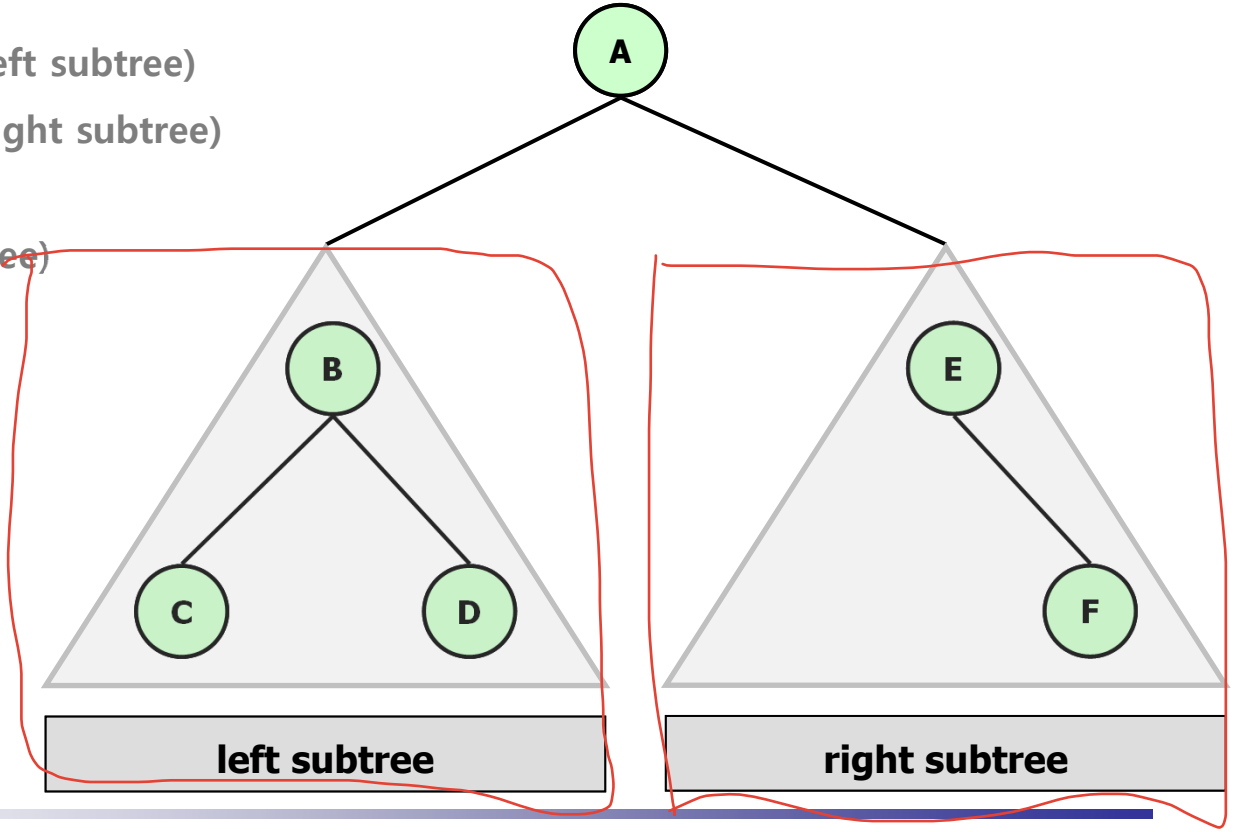


이진 트리

- 트리의 이해
- **이진 트리**
  - 이진 트리 순회
  - 이진 트리 구현
- 우선 순위 큐와 힙

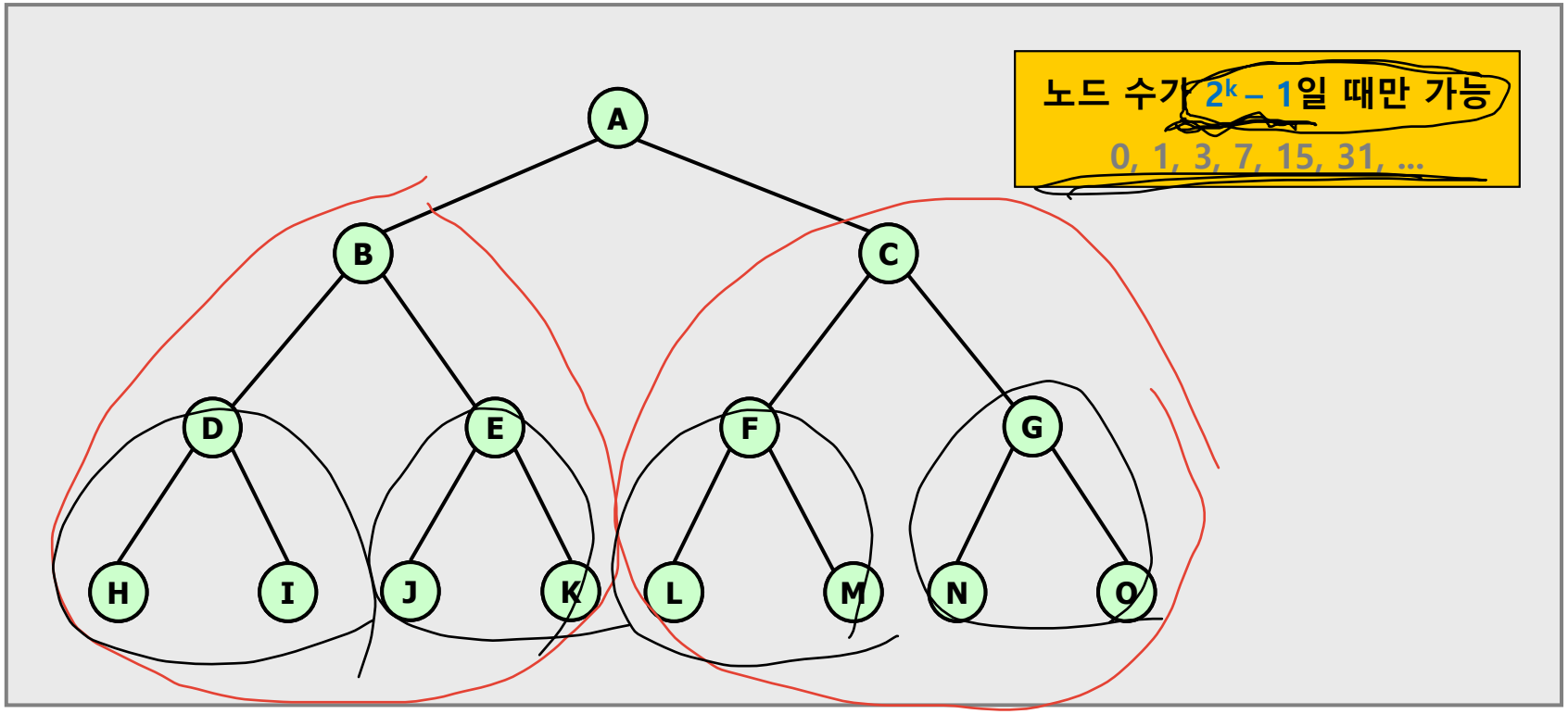
이진 트리 (1/4)

- **이진 트리**(Binary Tree)
  - 최대 두 개까지의 자식 노드를 가질 수 있는 트리
    - 하나의 노드는 0, 1, 혹은 2개의 서브 트리를 가질 수 있다.
    - 좌 서브 트리(left subtree)
    - 우 서브 트리(right subtree)
    - 널 트리(null tree)



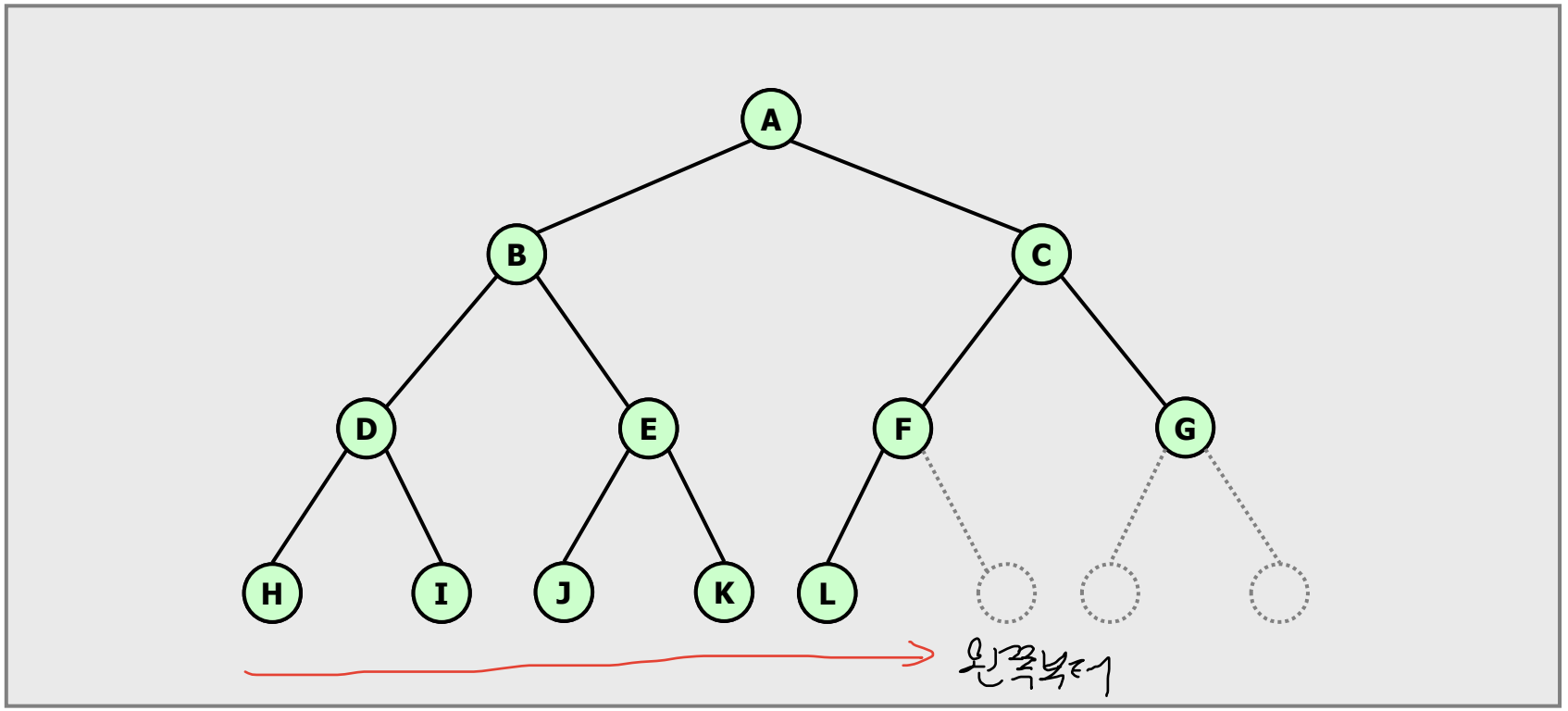
이진 트리 (2/4)

- **포화 이진 트리**(Full Binary Tree)
  - 루트로부터 시작해서 모든 노드가 정확히 두 개씩의 자식 노드를 가지도록 꽉 채워진 트리



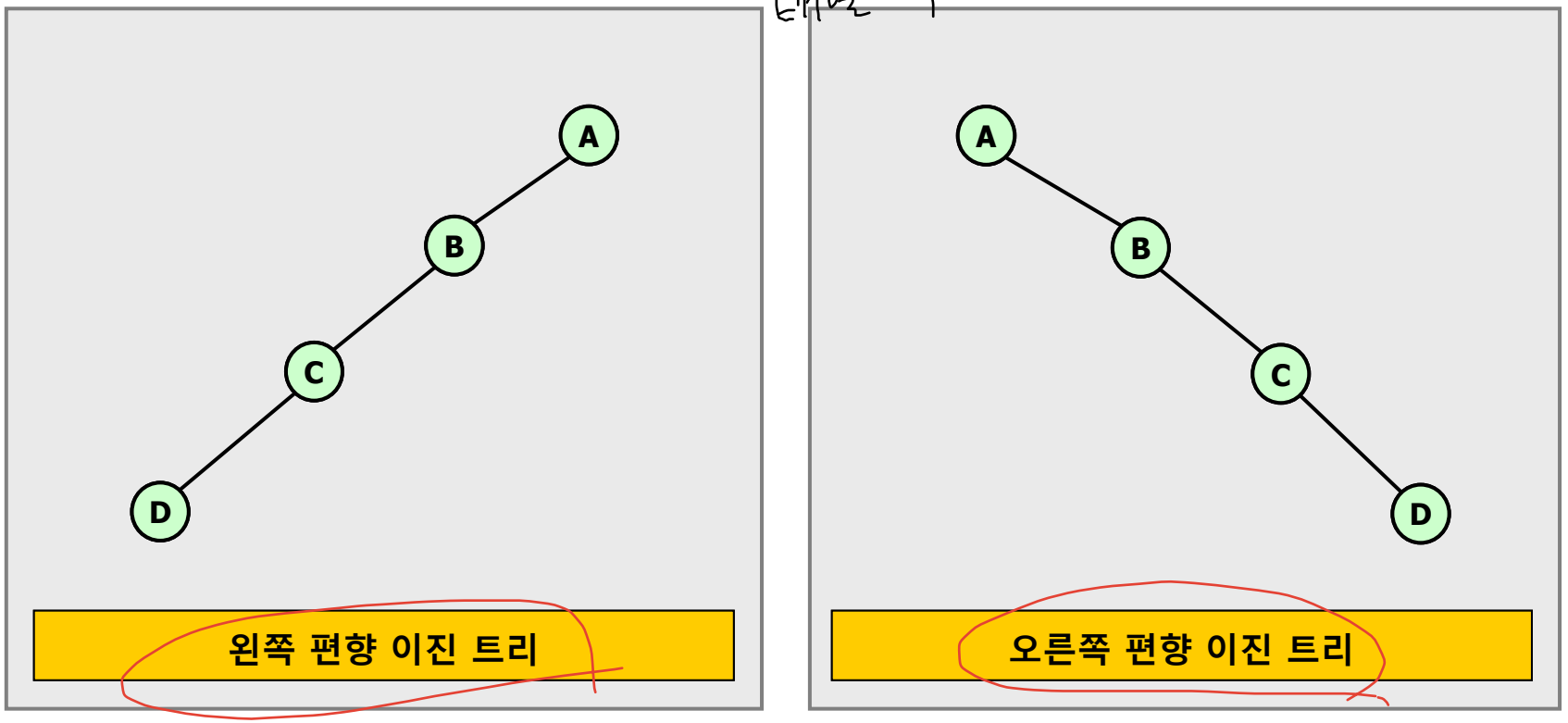
이진 트리 (3/4)

- **완전 이진 트리**(Complete Binary Tree)
  - 노드의 수가 맞지 않아 포화 이진 트리를 만들 수 없으면 맨 마지막 레벨은 왼쪽부터 채워 나간다.



이진 트리 (4/4)

- **편향 이진 트리**(Skewed Binary Tree)
  - 이진 트리 중에서 최소 개수의 노드를 가지면서 왼쪽이나 오른쪽 서브 트리만 가지고 있는 트리

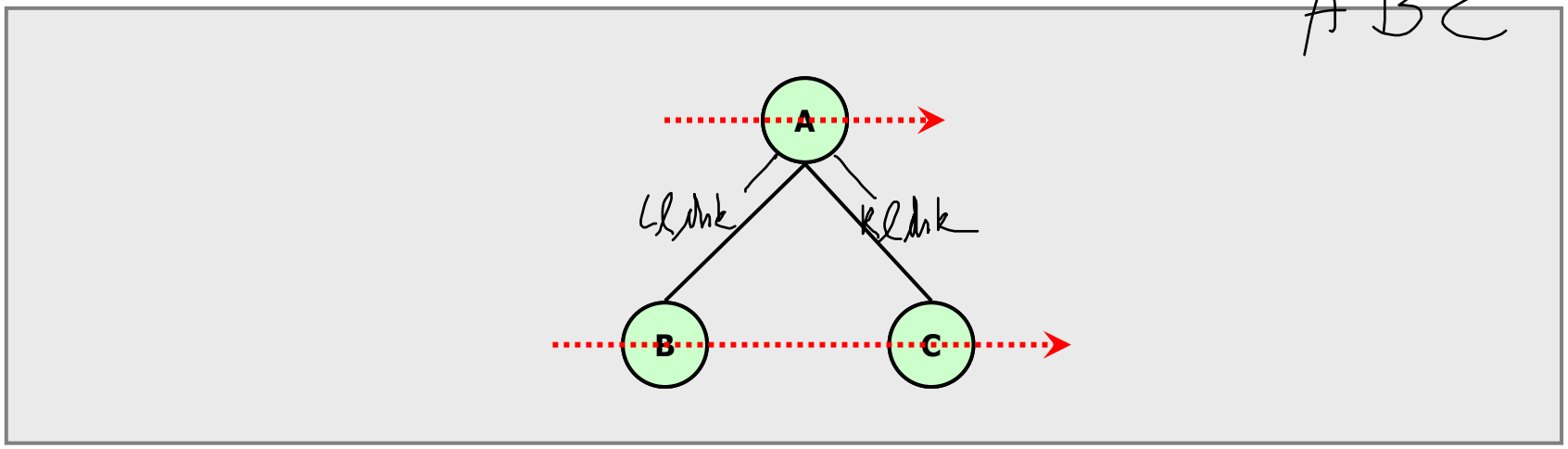


이진 트리

이진 트리 순회

이진 트리 순회 (1/4)

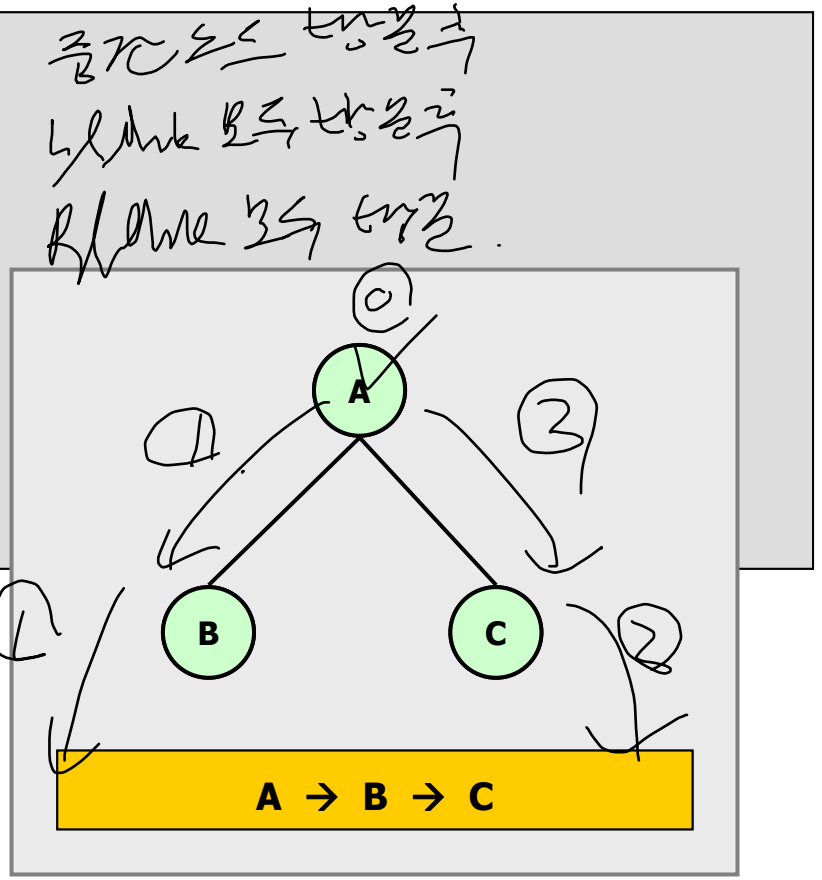
- **순회**(traversal)
  - 깊이 우선 순회: 스택을 이용하여 구현
    - 전위 순회(preorder traversal)
    - 중위 순회(inorder traversal)
    - 후위 순회(postorder traversal)
  - 너비 우선 순회: 큐를 이용하여 구현
    - 다음 레벨의 노드들을 처리하기 전에 노드의 자식 모두를 처리



이진 트리 순회 (2/4)

- **깊이 우선 순회:** 전위 순회
  - 전위 순회(preorder traversal)

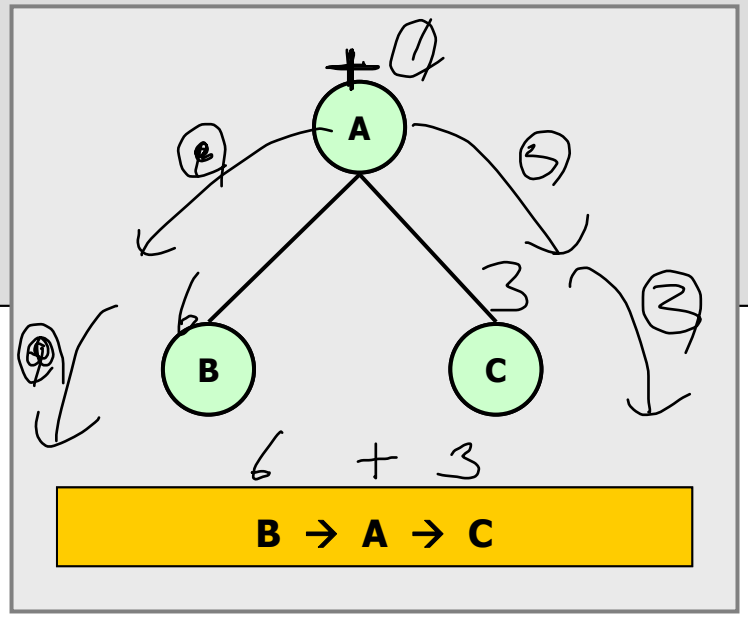
```
preorder(T)
    if(T ≠ NULL) then
    {
          visit T.data;
          preorder(T.Llink);
          preorder(T.Rlink);
    }
end preorder()
```



이진 트리 순회 (3/4)

- **깊이 우선 순회:** 중위 순회
  - 중위 순회(inorder traversal)

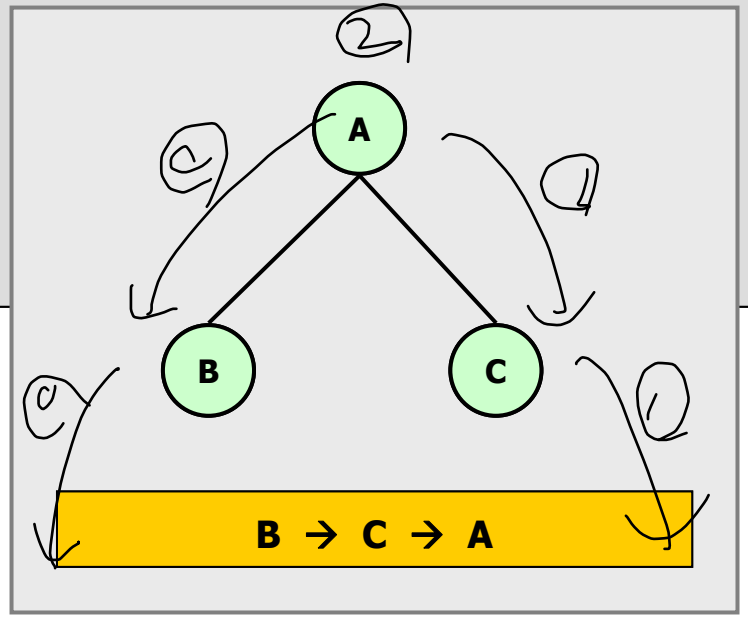
```
inorder(T)
    if(T ≠ NULL) then
    {
          inorder(T.Llink)
          visit T.data;
          inorder(T.Rlink);
    }
end inorder()
```



이진 트리 순회 (4/4)

- **깊이 우선 순회: 후위 순회**
  - 후위 순회(postorder traversal)

```
postorder(T)
    if(T ≠ NULL) then
    {
        postorder(T.Llink)
        postorder(T.Rlink);
        visit T.data;
    }
end postorder()
```

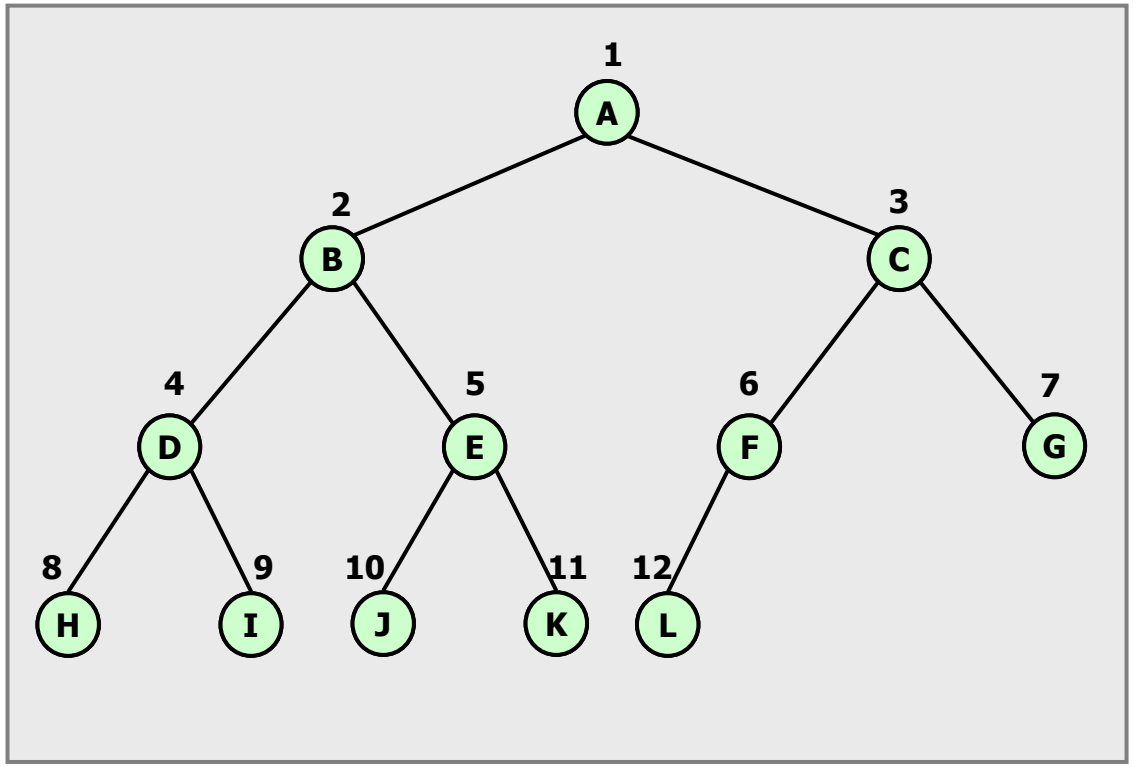


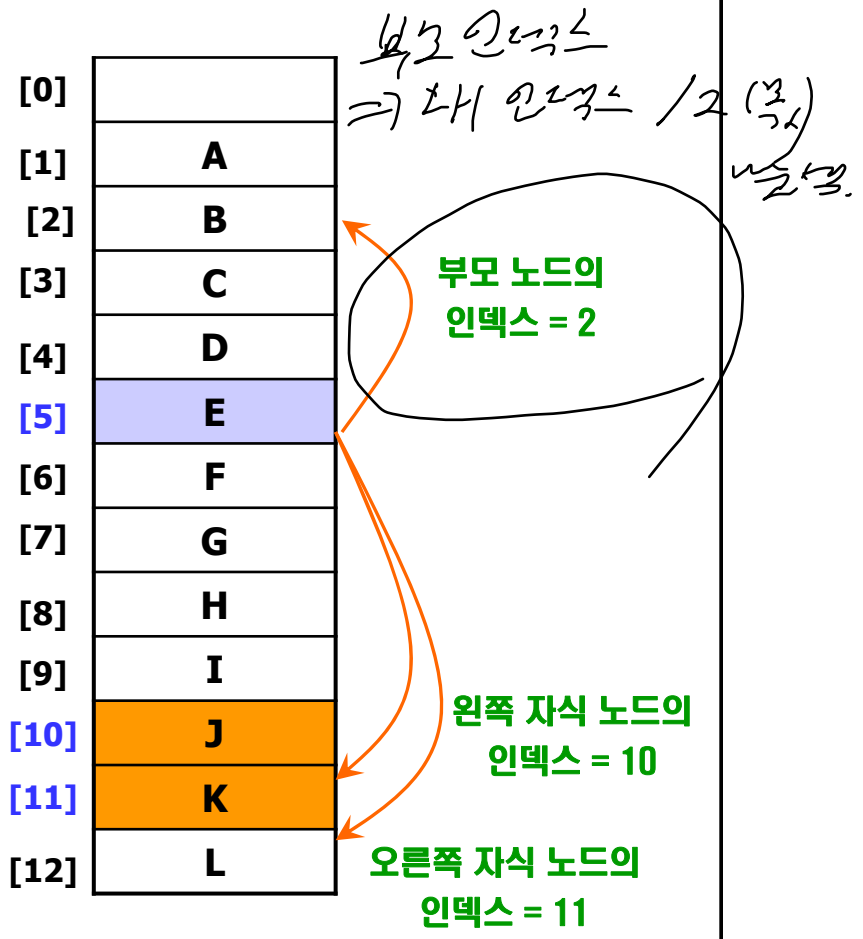
이진 트리

이진 트리 구현: 순차 자료구조

이진 트리 구현: 순차 자료 구조 (1/2)

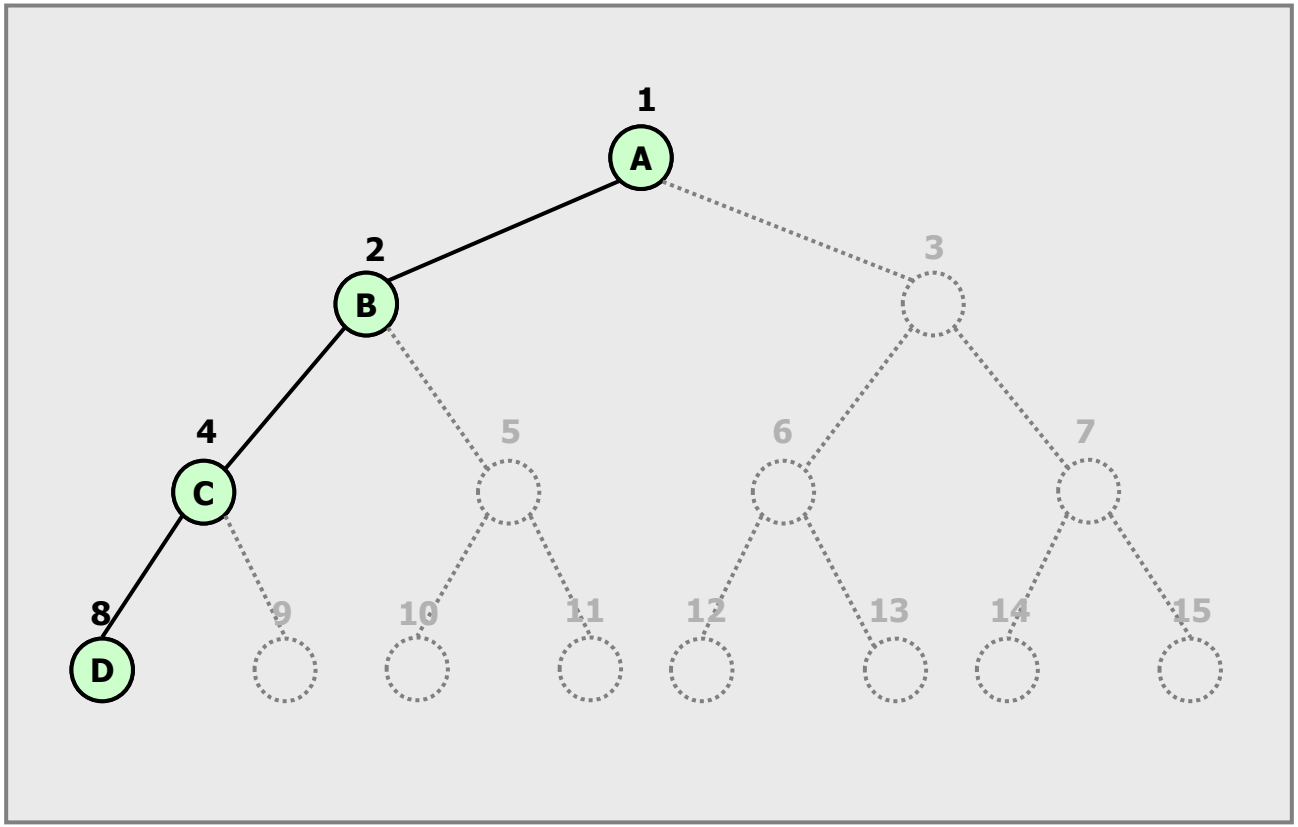
- **이진 트리 구현:** 순차 자료구조
  - 완전 이진 트리의 배열 표현

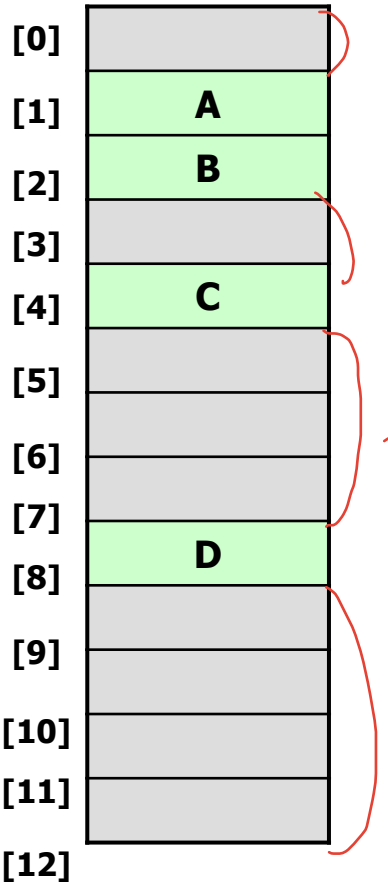




이진 트리 구현: 순차 자료 구조 (2/2)

- **이진 트리 구현:** 순차 자료구조
  - 편향 이진 트리의 배열 표현



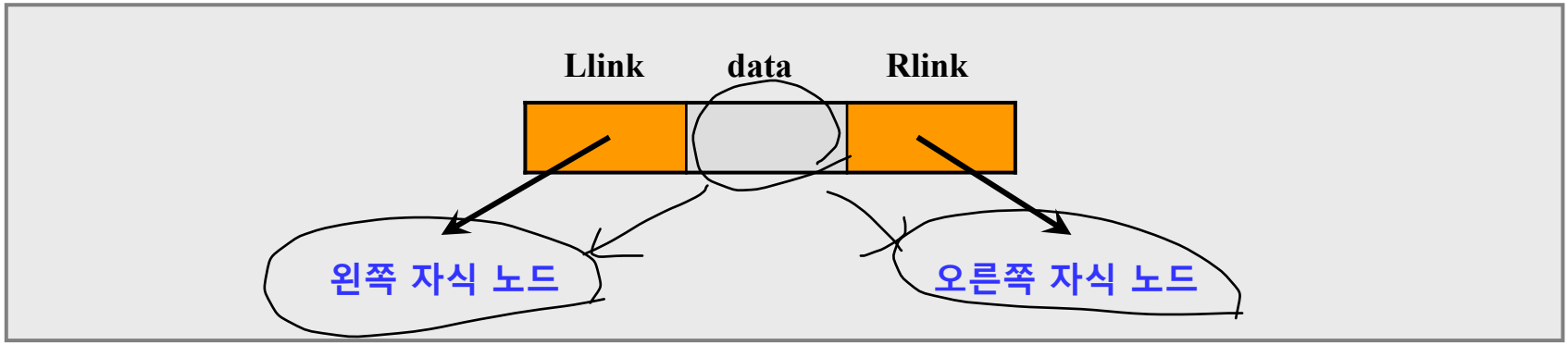


이진 트리

이진 트리 구현: 연결 자료구조

이진 트리 구현: 연결 자료 구조 (1/6)

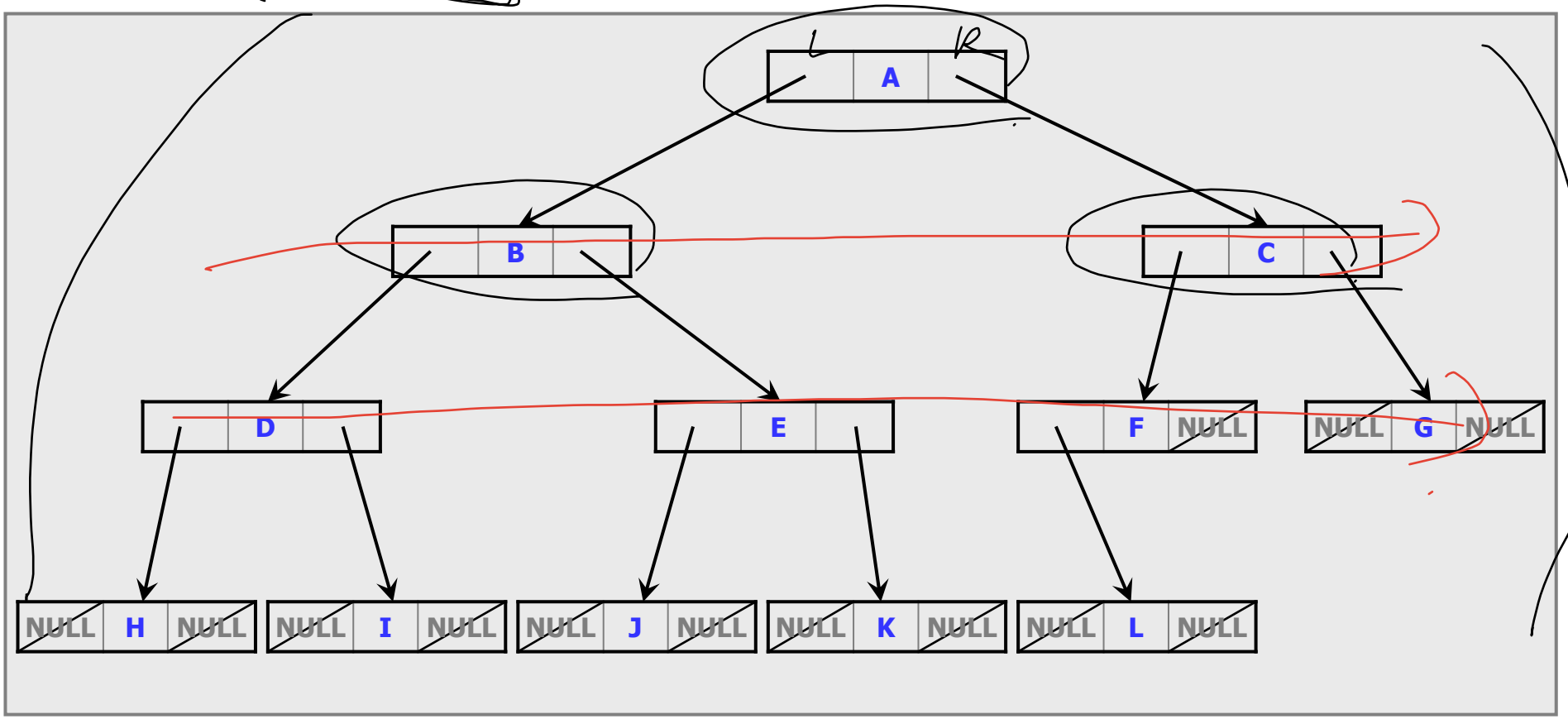
- **이진 트리 구현:** 연결 자료 구조




```
typedef struct _BTreenode
{
    int                     data;
    struct _BTreenode*      Llink;
    struct _BTreenode*      Rlink;
}BTreeNode;
```

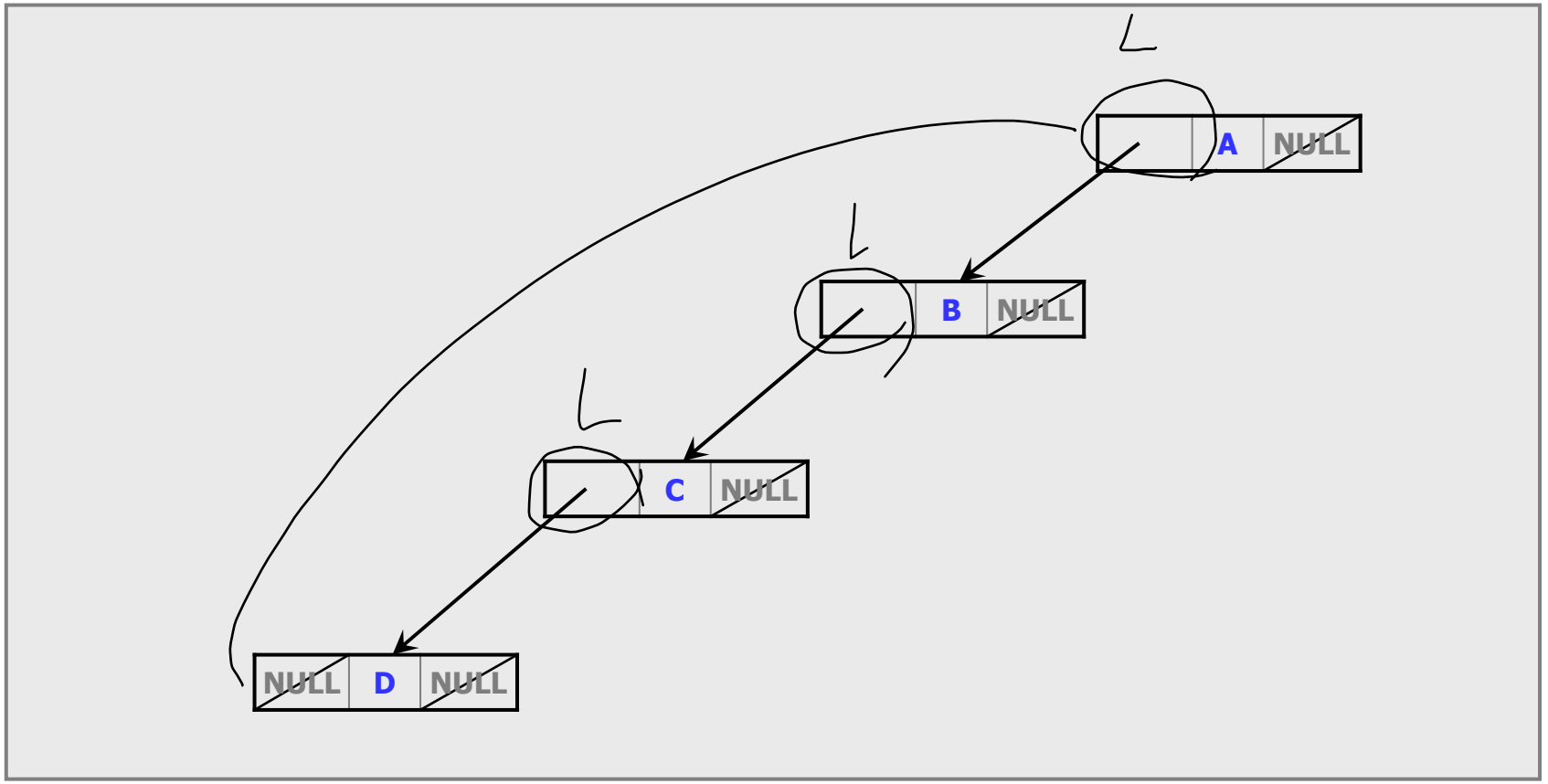
이진 트리 구현: 연결 자료 구조 (2/6)

- **이진 트리 구현:** 연결 자료 구조
  - 완전 이진 트리의 연결 자료 구조 형태



이진 트리 구현: 연결 자료 구조 (3/6)

- **이진 트리 구현:** 연결 자료 구조
  - 편향 이진 트리의 연결 자료 구조 형태



이진 트리 구현: 연결 자료 구조 (4/6)

- **이진 트리 구현:** 연결 자료구조

C

```c
// 이진 트리 구현: 단순 연결 리스트
// #pragma once
#ifndef __BinaryTree_H_
#define __BinaryTree_H_
// 이진 트리 노드: data, Llink, Rlink
typedef struct _treeNode {
        char              data;
        struct _treeNode*Llink;
        struct _treeNode*Rlink;
} treeNode;
#endif

// 이진 트리 구현: 이진 트리 생성
treeNode* makeBinaryTree(char*);
treeNode* makeTreeNode(char);
int       isOperator(int);
int       isLegal(char*);

// 이진 트리 순회: 전위.중위.후위 순회
void      Preorder(treeNode*);
void      Inorder(treeNode*);
void      Postorder(treeNode*);

// 이진 트리 순회: 너비 우선 순회
void      Preorder(treeNode*);
```

이진 트리 구현: 연결 자료 구조 (5/6)

- **이진 트리 구현:** 연결 자료구조

C++

```
template <typename E>
class BinaryTreeNode {
private:
    E                       data;
    BinaryTreeNode<E>*      Llink;
    BinaryTreeNode<E>*      Rlink;
    template <typename E> friend class LinkedBinaryTree;
};

template <typename E>
class LinkedBinaryTree {
private:
    BinaryTreeNode<E>* root;
public:
    LinkedBinaryTree();
    ~LinkedBinaryTree();
    BinaryTreeNode<E>* makeBinaryTree(const char& ch) const;
    BinaryTreeNode<E>* makeLinkedBinaryTree(const char* pStr) const;
    void    Preorder(void) const;
    void    Inorder(void) const;
    void    Postorder(void) const;
    void    Levelorder(void) const;
};
```

이진 트리 구현: 연결 자료 구조 (6/6)

- **이진 트리 구현:** 연결 자료구조

Python

```
from LinkedStack import LinkedStack
from LinkedQueue import LinkedQueue

def isOperator(op) -> bool :

class TreeNode :
    def __init__ (self, data, Llink=None, Rlink=None):
        self.data = data
        self.Llink = Llink
        self.Rlink = Rlink

class LinkedBinaryTree :
    def __init__(self):
        self.__root = None

    def makeLinkedBinaryTree(self, postfix) -> TreeNode:
    def Preorder(self):
    def Inorder(self) :
    def Postorder(self) :
    def Levelorder(self) :
```

참고문헌

[1] Michael T. Goodrich 외 2인 지음, 김유성 외 2인 옮김, “C++로 구현하는 자료구조와 알고리즘”, 한티에듀, 2020.

[2] “프로그래밍 대회 공략을 위한 알고리즘과 자료 구조 입문”, 와타노베 유타카 저, 윤인성 역, 인사이트, 2021.

[3] “IT CookBook, 쉽게 배우는 자료구조 with 파이썬”, 문병로, 한빛아카데미, 2022.

[4] “이것이 취업을 위한 코딩 테스트다 with 파이썬”, 나동빈, 한빛미디어, 2020.

[5] 문병로, “IT CookBook, 쉽게 배우는 알고리즘: 관계 중심의 사고법”(개정판), 개정판, 한빛아카데미, 2018.

[6] Richard E. Neapolitan, 도경구 역, “알고리즘 기초”, 도서출판 홍릉, 2017.

[7] 주우석, “IT CookBook, C・C++ 로 배우는 자료구조론”, 한빛아카데미, 2019.

[8] 이지영, “C 로 배우는 쉬운 자료구조”, 한빛아카데미, 2022.