자료구조 5주차 실습

Tree

감성인공지능연구실
방윤석 임희수

인하대학교
INHA UNIVERSITY

퀴즈 안내 1

인하대학교 INHA UNIVERSITY

- 퀴즈 일정 : 4월 11일
- 퀴즈 범위 : 2주차 ~ 4주차
- 퀴즈 시간 : 금요일 오전 이론 수업시간(오전 9시 ~ 10시30분)
- 퀴즈는 VSCode로만 진행하며 개인컴퓨터 사용 금지
- 총 3문제

퀴즈 안내 2

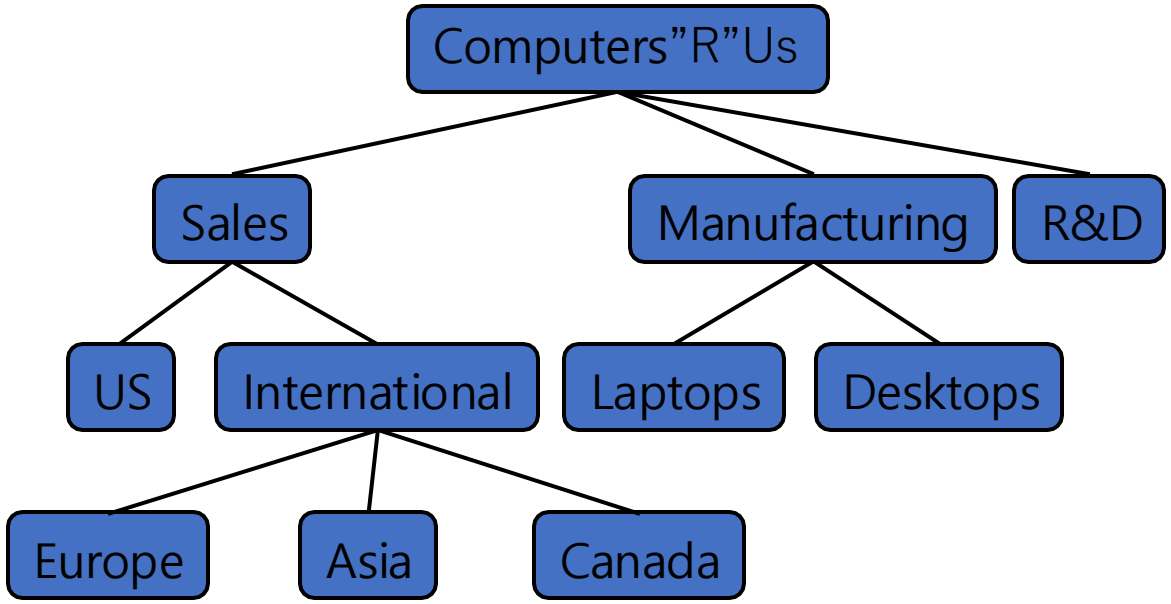
인하대학교 INHA UNIVERSITY

- 5분 이상 지각시 입실 금지(9시 05분부터 입실 금지)
- 입실시 시간과 서명, 퇴실시 시간과 서명
- 문제 파일과 VSCode를 제외한 파일이 열려있을시 퇴장조치
- 자동완성기능 적발시 퇴장 조치(OT 파일에 끄는 법 자세히 안내)
- 총 3문제 : Problem 2문제 + 새로운 문제 1문제

Tree

Revisit Tree

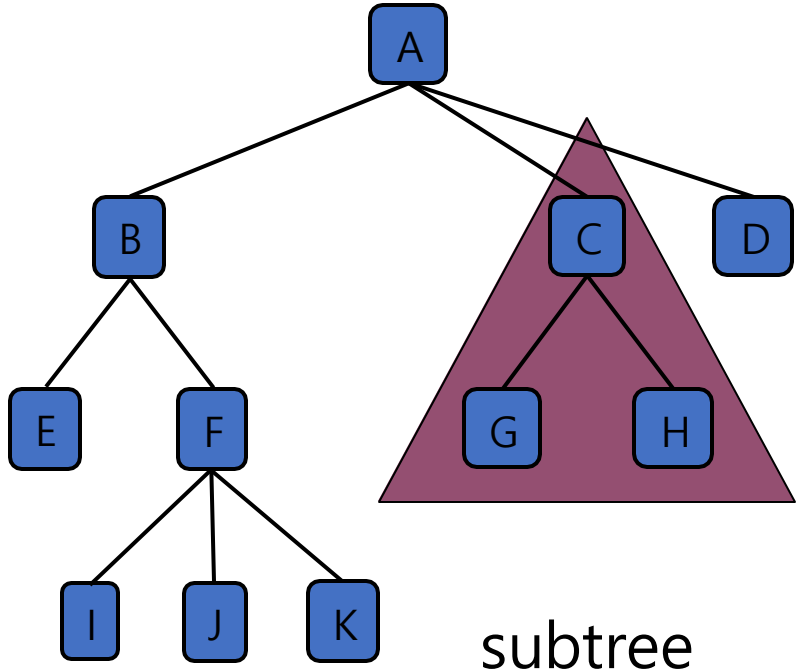
- In computer science, a tree is an abstract model of a hierarchical structure
- A tree consists of nodes with a parent-child relation
- Applications:
  - Organization charts
  - File systems
  - Programming environments



Revisit Tree

- **Root**: node without parent (A)
- **Internal node**: node with at least one child (A, B, C, F)
- **External node** (a.k.a. leaf): node without children (E, I, J, K, G, H, D)
- **Ancestors** of a node: parent, grandparent, grand-grandparent, etc.
- **Depth** of a node: number of ancestors
- **Height** of a tree: maximum depth of any node
- **Descendant** of a node: child, grandchild, grand-grandchild, etc.

- **Subtree**: tree consisting of a node and its descendants



subtree

Revisit Tree

인하대학교 INHA UNIVERSITY

Tree

- 재귀적으로 연결되는 형태의 자료구조이다.
  - 노드 관점에서, 부모 노드 아래 자식 노드들이 연결되고, 자식 노드 아래 또다른 자식 노드들이 연결됨
  - Tree 관점에서, tree안에 subtree들이 연결되어 있음
- 일반적인 tree는 자식의 수가 제한되지 않음
  - 자식의 수가 제한되지 않기 위해, linked list나 vector 등의 자료구조를 이용해서 child 들의 위치를 표시
  - 알고리즘이나 자료구조 등에서는 **자식 노드가 최대 2개인 이진트리**를 많이 씀
    - 자식의 수가 제한되지 않으면, 메모리 낭비가 큼

이진트리 (Binary Tree)

- 자식 노드가 최대 2개이다.
- 자식 노드가 2개다보니 left, right 로 구분한다

Binary tree

- Node : 본인의 값을 가지고 있고, child를 나타내는 변수 2개 있음
- is_leaf : 해당 node가 leaf인지 확인
- height : 해당 node의 heigh를 return

```python
class Node: # Tree Node class
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None

def is_leaf(root): # Check if the node is a leaf node
    return root.left is None and root.right is None

def height(position): # Calculate the height of the tree
    if position is None: # Check if the node is None
        return -1
    if is_leaf(position):
        return 0
    else:
        height_left = height(position.left)
        height_right = height(position.right)
        if height_left > height_right:
            return height_left + 1
        else:
            return height_right + 1
```

Exercise #1

인하대학교 INHA UNIVERSITY

지금까지 구현한 이진트리의 문제점

- Depth를 구할 수 없음
- 특정 노드의 부모를 확인할 수 없음
- 물론, 현재 구조로 구할 수 있는 내용이긴 하다. 하지만, 너무 비효율적…

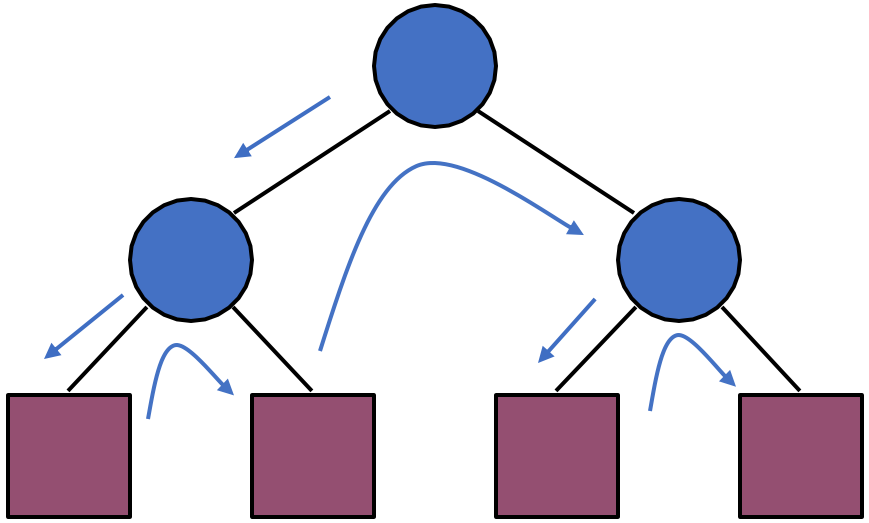
Depth와 특정 node의 parent를 확인할 수 있는 tree를 만드시오

- 힌트 : node에 변수 하나 추가하세요.

DFS(Depth First Search)

DFS란, **루트에서 시작**해서 한 쪽 방향의 가장 끝까지 탐색한 후, 돌아와서 다음 방향으로 넘어가며 탐색하는 방법

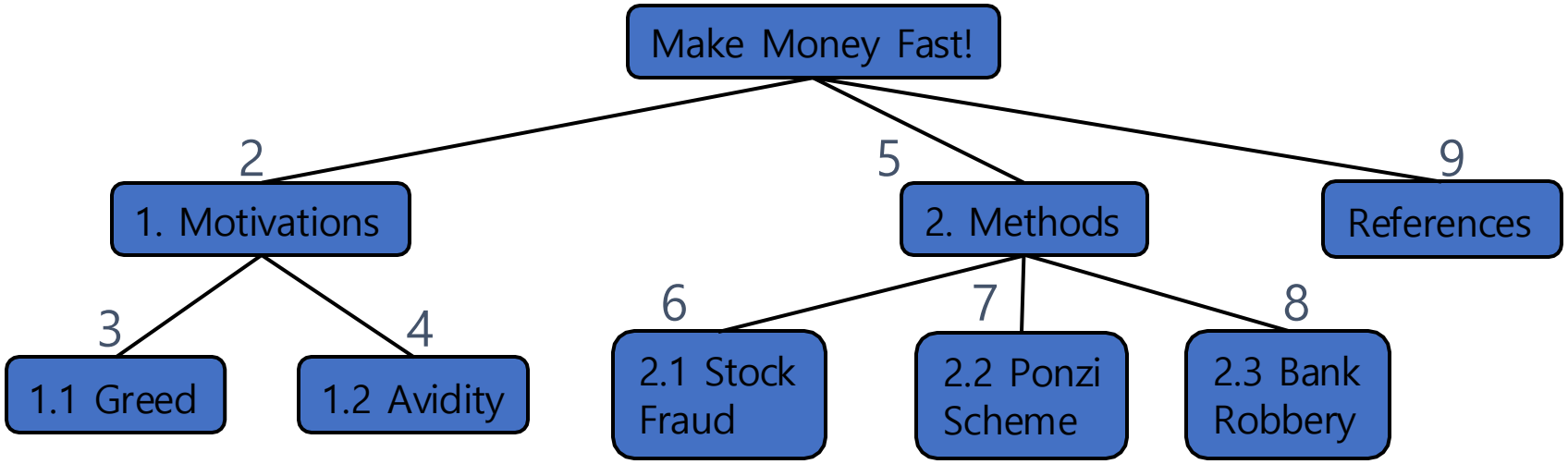
- 갈 수 있는 가장 깊은 노드까지 우선 출력하면 됨
- preorder, postorder, inorder는 모두 DFS 방식



Preorder Traversal

- 전위 순회(preorder traversal): 자신, 자식 순서로 방문하는 순회
- A traversal visits the nodes of a tree in a systematic manner
- In a preorder traversal, a node is visited before its descendants
- Application: print a structured document

Algorithm *preOrder*(*v*)
    visit(*v*)
    for each child *w* of *v*
        preorder (*w*)

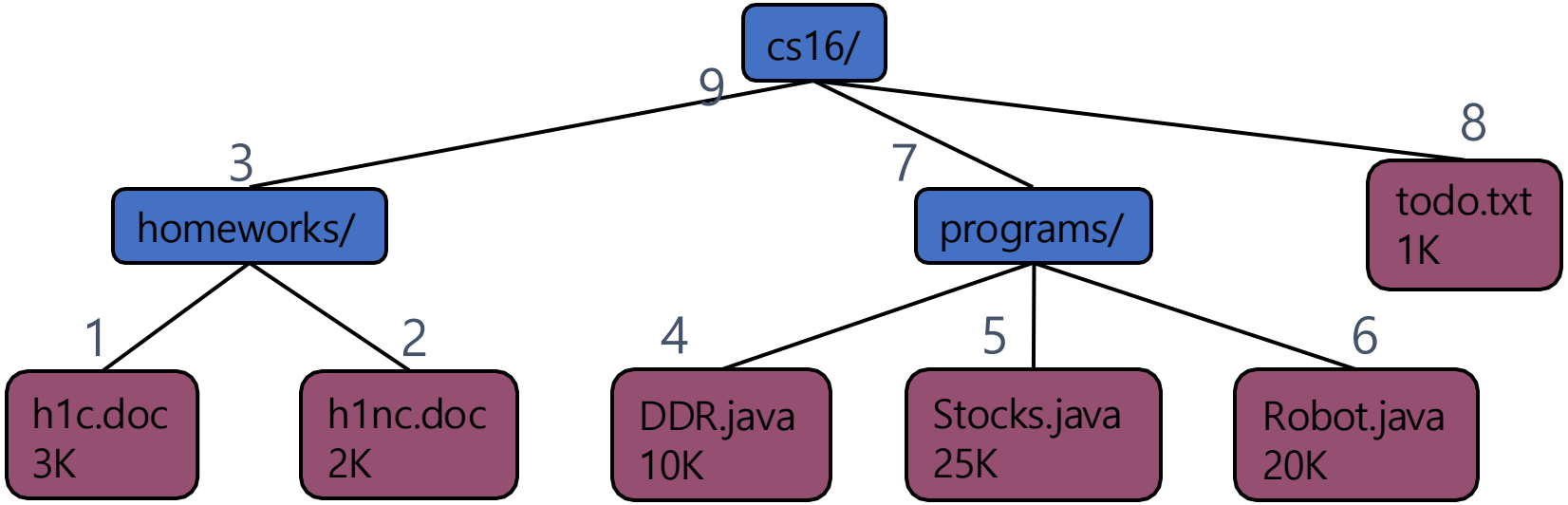


Postorder Traversal

- 후위 순회(postorder traversal): 자식, 자신 순서로 방문하는 순회

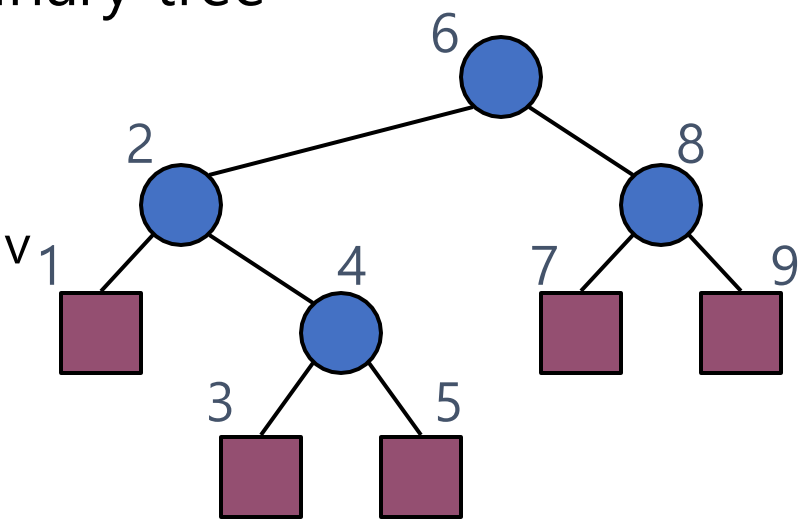
- In a postorder traversal, a node is visited after its descendants
- Application: compute space used by files in a directory and its subdirectories

Algorithm *postOrder*(*v*)
  for each child *w* of *v*
    postOrder (*w*)
  visit(*v*)



Inorder Traversal

- 중위 순회(inorder traversal): 왼쪽 자손, 자신, 오른쪽 자손 순서로 방문하는 순회
- In an inorder traversal a node is visited after its left subtree and before its right subtree
- Application: draw a binary tree
  - x(v) = inorder rank of v
  - y(v) = depth of v



Algorithm *inOrder*(*v*)
  if *v* **has a left child**
    inOrder (*left* (*v*))
  visit(*v*)
  if *v* **has a right child**
    inOrder (*right* (*v*))

Summarize Traversal

인하대학교 INHA UNIVERSITY

Tree 탐색 요약

자손을 순회해야 하면, 가장 왼쪽 자식 노드에서 가장 오른쪽 자식 노드로 탐색하는 방향

- 전위 순회(preorder traversal): 자신, 자식 순서로 방문하는 순회
- 후위 순회(postorder traversal): 자식, 자신 순서로 방문하는 순회

이진 트리에서의 탐색

- 전위 순회(preorder traversal): 자신, 왼쪽 자손, 오른쪽 자손 순서로 방문하는 순회
- 후위 순회(postorder traversal): 왼쪽 자손, 오른쪽 자손, 자신 순서로 방문하는 순회
- 중위 순회(inorder traversal): 왼쪽 자손, 자신, 오른쪽 자손 순서로 방문하는 순회

Binary tree traversal

- 전위 순회(preorder traversal):
  자신, 왼쪽 자손, 오른쪽 자손

- 후위 순회(postorder traversal):
  왼쪽 자손, 오른쪽 자손, 자신

- 중위 순회(inorder traversal):
  왼쪽 자손, 자신, 오른쪽 자손

```python
class Node: # Tree Node class
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None


def PreorderTraversal(root): # Root, Left, Right
    if root: # Check if the node is not None
        print(root.data, end=' ')
        PreorderTraversal(root.left)
        PreorderTraversal(root.right)

def InorderTraversal(root): # Left, Root, Right
    if root: # Check if the node is not None
        InorderTraversal(root.left)
        print(root.data, end=' ')
        InorderTraversal(root.right)

def PostorderTraversal(root): # Left, Right, Root
    if root: # Check if the node is not None
        PostorderTraversal(root.left)
        PostorderTraversal(root.right)
        print(root.data, end=' ')
```

Exercise # 2

인하대학교 INHA UNIVERSITY

무한한 child가 있는 tree를 만드시오

무한한 child가 있는 tree를 만들고, 이에 대한 전위탐색과 후위탐색을 구현하시오.

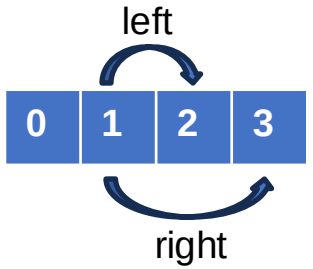
Array based binary tree

Array based binary tree

자식 노드는 2개밖에 없는데, 노드를 하나하나 가리키면서 생성해야 할까?

배열의 index로 접근하면 어떨까?

- index 0는 비우고, index 1을 root로 한다.
- 왼쪽 자식은 index *2 , 오른쪽 자식은 index * 2 + 1의 위치에 저장한다.
- 배열 기반이다보니, 생성할 수 있는 노드의 수가 제한된다.



Index 0을 root로 하면 안되는가?

문제는 없다.

- 왼쪽 자식은 index *2 + 1 , 오른쪽 자식은 index * 2 + 2의 위치에 저장한다.
- Index 0을 비우는 이유는, 나중에 정렬알고리즘 등을 구현할 때 해당 공간을 사용하기도 하기 때문

Array based binary tree

- setLeftChild, setRightChild :
  parent 의 child를 설정

- getValue : 해당 index가 가리키는
  node의 값 return

- getLeftChild, getRightChild, getParent :
  left child, right child, paren의 값 return

```python
class Tree:
    def __init__(self, n, rootValue): # Initialize the tree with the given capacity
        self.capacity = n
        self.array = array.array('h', [0]*self.capacity)
        self.array[1] = rootValue # The first element is the root node

    def setLeftChild(self, parentIndex, childValue):
        # Check if the left child index is within bounds and the parent index is valid
        if parentIndex * 2 >= self.capacity or parentIndex < 1:
            return
        self.array[parentIndex * 2] = childValue # Set the left child

    def setRightChild(self, parentIndex, childValue):
        # Check if the right child index is within bounds and the parent index is valid
        if parentIndex * 2 + 1 >= self.capacity or parentIndex < 1:
            return
        self.array[parentIndex * 2 + 1] = childValue

    def getValue(self, index):
        # Check if the index is within bounds
        if index < 1 or index >= self.capacity:
            return None
        return self.array[index]

    def getLeftChild(self, index):
        # Check if the left child index is within bounds and the parent index is valid
        if index * 2 >= self.capacity or index < 1:
            return None
        return self.array[index * 2]

    def getRightChild(self, index):
        # Check if the right child index is within bounds and the parent index is valid
        if index * 2 + 1 >= self.capacity or index < 1:
            return None
        return self.array[index * 2 + 1]

    def getParent(self, index):
        # Check if the child index is within bounds
        if index <= 1 or index >= self.capacity:
            return None
        return self.array[index // 2] # Return the parent node value
```

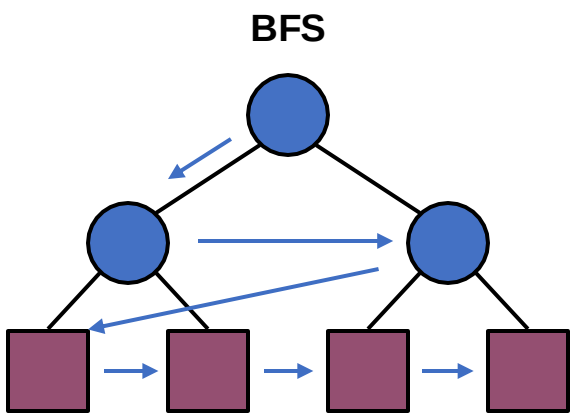
Exercise #3

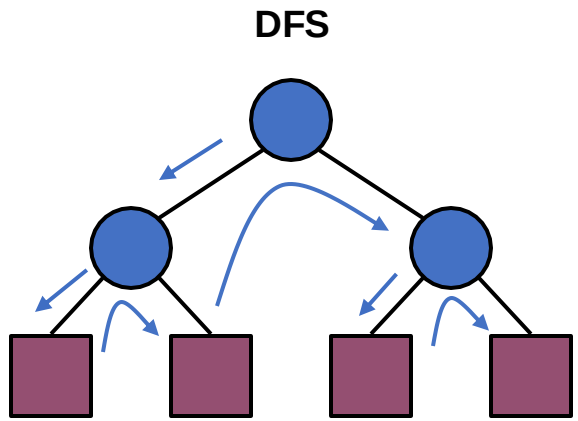
인하대학교 INHA UNIVERSITY

BFS(Breadth First Search)

너비우선탐색(BFS)를 구현하여라. BFS는 루트에서 가까운 노드부터 멀리 있는 노드 순서로 탐색하는 방식

- depth가 동일한 node들을 먼저 출력하면 됨





Problem #1

인하대학교 INHA UNIVERSITY

문제

이진 트리가 주어졌을 때, 트리의 경계를 다음과 같은 규칙으로 순회하여 출력하시오.

- 루트 노드에서 시작하여 트리의 왼쪽 경계를 위에서 아래로 출력한다. 단, 리프 노드는 제외한다.
- 모든 리프 노드를 왼쪽에서 오른쪽으로 출력한다.
- 트리의 오른쪽 경계를 아래에서 위로 출력한다. 단, 리프 노드는 제외한다.

주어진 트리의 경계 순회를 출력하는 class를 작성하시오.

메소드 설명

- boundaryTraversal(root) : 이진 트리의 루트 노드를 입력받아 경계 순회 결과를 리스트 형태로 반환한다.

Problem #1

인하대학교 INHA UNIVERSITY

입력 예시

```
#         20
#        /  \
#       8    22
#      / \     \
#     4   12    25
#        /  \
#       10   14
root = Node(20)
root.leftChild = Node(8)
root.rightChild = Node(22)
root.leftChild.leftChild = Node(4)
root.leftChild.rightChild = Node(12)
root.leftChild.rightChild.leftChild = Node(10)
root.leftChild.rightChild.rightChild = Node(14)
root.rightChild.rightChild = Node(25)

boundaryTraversal(root)
```

출력 예시

```
20 8 4 10 14 25 22
```

Problem #2

인하대학교 INHA UNIVERSITY

문제

이진 트리가 주어졌을 때, 트리 내 두 리프 노드 사이의 경로 중 합이 최대가 되는 경로를 찾아 그 합을 출력하는 method를 작성하시오. 경로의 합은 경로에 포함된 모든 노드의 값을 합산한 값이다.

메소드 설명

- maxPathSum(Node root): 이진 트리의 루트 노드를 입력 받아 두 리프 노드 사이의 경로 중 최대 합을 반환한다.

Problem #2

인하대학교 INHA UNIVERSITY

입력 예시

```
#          1
#        /   \
#      -2     3
#      / \   / \
#     8  -1  4  -5


root = Node(1)
root.leftChild = Node(-2)
root.rightChild = Node(3)
root.leftChild.leftChild = Node(8)
root.leftChild.rightChild = Node(-1)
root.rightChild.leftChild = Node(4)
root.rightChild.rightChild = Node(-5)

print(findMaxPathSum(root))
```

출력 예시