

Data Structure

자료구조



이진 트리의 연산



한국기술교육대학교
온라인평생교육원

학습내용

- 이진 트리의 순회
- 이진 트리의 연산들

학습목표

- 이진 트리의 순회 방법을 설명하고 구현할 수 있다.
- 이진 트리의 다양한 연산들을 설명하고 구현할 수 있다.

이진 트리의 순회



이진 트리의 연산

이진 트리의 순회

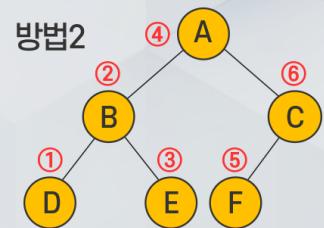
1 이진 트리의 표준 순회

● 순회(Traversal)란?

순회(Traversal)

- 트리에 속하는 모든 노드를 한번 씩 방문하는 것
- 선형 자료구조는 순회가 단순
- 트리는 다양한 방법이 있음

• 전위 순회, 중위 순회, 후위 순회, 레벨 순회 등



이진 트리의 연산

이진 트리의 순회

1 이진 트리의 표준 순회

1 전위 순회(Preorder Traversal) : VLR

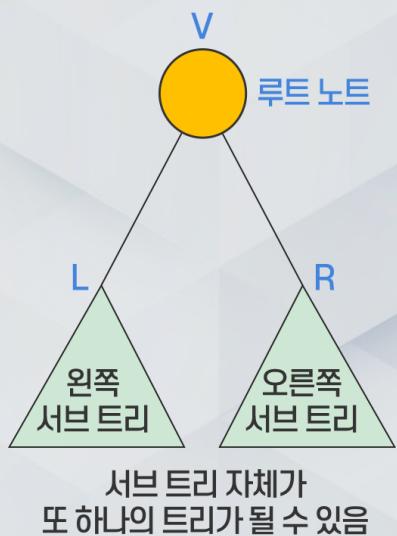
- 루트 → 왼쪽 서브 트리 → 오른쪽 서브 트리

2 중위 순회(Inorder Traversal) : LVR

- 왼쪽 서브 트리 → 루트 → 오른쪽 서브 트리

3 후위 순회(Postorder Traversal) : LRV

- 왼쪽 서브 트리 → 오른쪽 서브 트리 → 루트





이진 트리의 순회

이진 트리의 연산

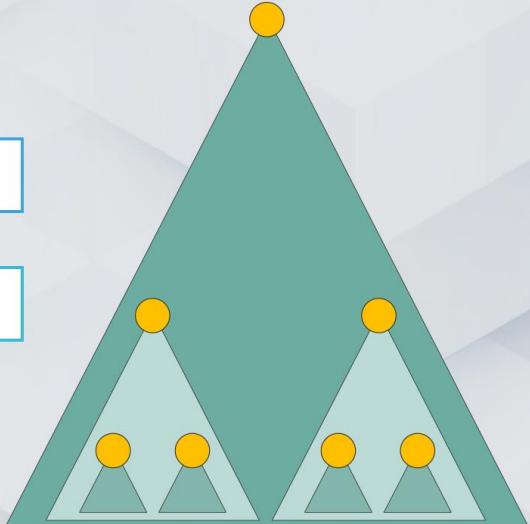
이진 트리의 순회

1 이진 트리의 표준 순회

전체 트리나 서브 트리나 구조는 동일

표준 순회는 순환 구조로 기술

- 순환이 진행될수록 문제의 크기는 줄어듦



이진 트리의 연산

이진 트리의 순회

2 표준 순회 알고리즘

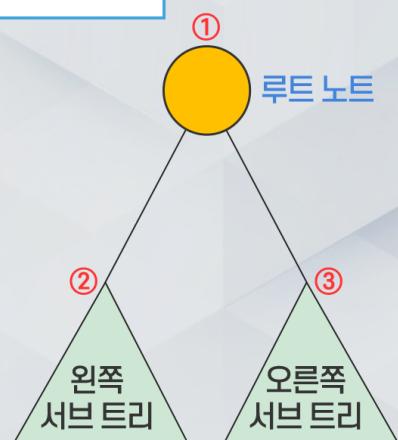
● 전위 순회

전위 순회

루트 \rightarrow 왼쪽 서브 트리 \rightarrow 오른쪽 서브 트리

응용 예

- 노드의 레벨 계산
 - 부모 노드(루트 노드)의 레벨이 계산되어야 자식 노드의 레벨이 계산
- 구조화된 문서 출력
 - 책의 목차





이진 트리의 순회

이진 트리의 연산

이진 트리의 순회

2 표준 순회 알고리즘

● 전위 순회 알고리즘

```
def preorder(n) : _____ # 전위 순회 함수
    if n is not None :
        print(n.data, end=' ')
        preorder(n.left) _____ # 루트 노드를 먼저 처리
        preorder(n.right) _____ # 왼쪽 서브 트리 처리
                                # 오른쪽 서브 트리 처리
```

이진 트리의 연산

이진 트리의 순회

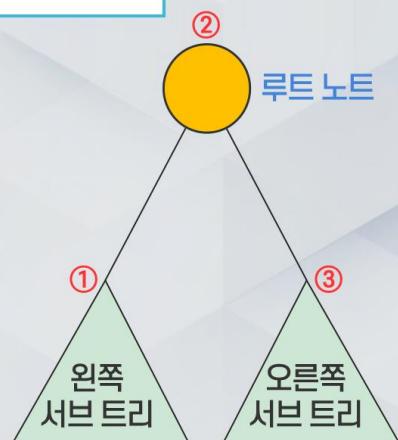
2 표준 순회 알고리즘

● 중위 순회

중위 순회

왼쪽 서브 트리 → 루트 → 오른쪽 서브 트리

예 정렬





이진 트리의 순회

이진 트리의 연산

이진 트리의 순회

2 표준 순회 알고리즘

● 중위 순회 알고리즘

```
def inorder(n) : _____ # 중위 순회 함수
    if n is not None :
        inorder(n.left) _____ # 왼쪽 서브 트리 처리
        print(n.data, end=' ') _____ # 루트 노드를 처리
        inorder(n.right) _____ # 오른쪽 서브 트리 처리
```

이진 트리의 연산

이진 트리의 순회

2 표준 순회 알고리즘

● 후위 순회

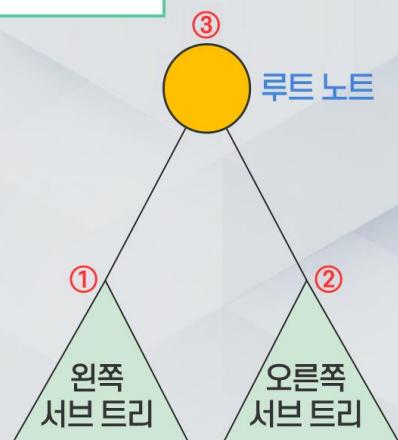
후위 순회

왼쪽 서브 트리 → 오른쪽 서브 트리 → 루트

응용 예

- 폴더 용량 계산

- 컴퓨터 폴더



이진 트리의 순회



이진 트리의 연산

이진 트리의 순회

2 표준 순회 알고리즘

◎ 후위 순회 알고리즘

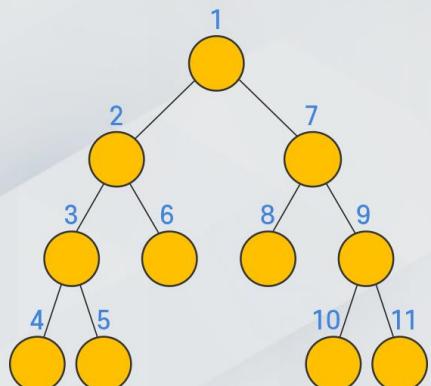
```
def postorder(n) : _____ # 후위 순회 함수
    if n is not None :
        postorder(n.left) _____ # 왼쪽 서브 트리 처리
        postorder(n.right) _____ # 오른쪽 서브 트리 처리
        print(n.data, end=' ') _____ # 루트 노드를 처리
```

이진 트리의 연산

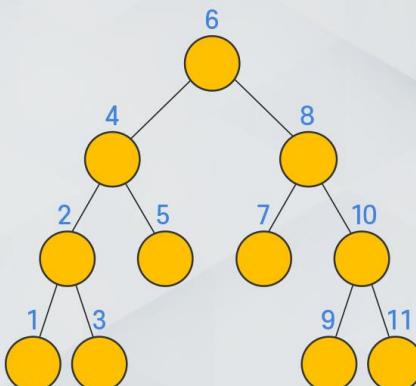
이진 트리의 순회

2 표준 순회 알고리즘

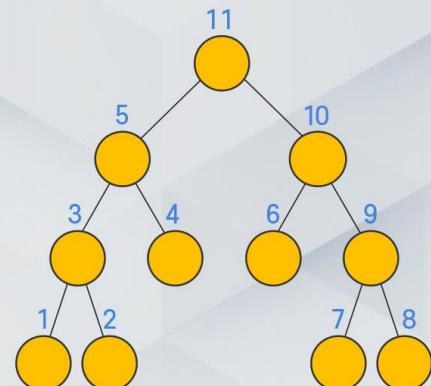
◎ 순회 방법에 따른 노드 방문 순서



전위 순회



중위 순회



후위 순회

이진 트리의 순회



이진 트리의 연산

이진 트리의 순회

3 이진 트리의 레벨 순회

레벨 순회

노드를 레벨 순으로 검사하는 순회 방법

레벨 순회의 특징

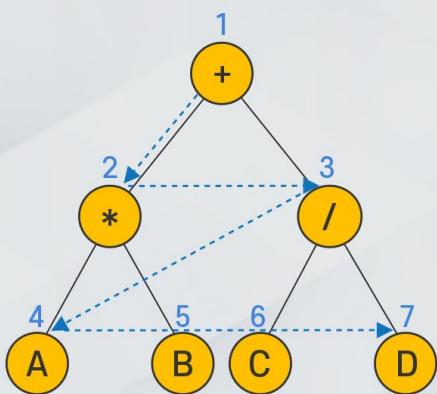
- 트리의 전체 구조를 이해하기 쉬움
- 큐를 사용해 구현
- 순환을 사용하지 않음

이진 트리의 연산

이진 트리의 순회

3 이진 트리의 레벨 순회

● 레벨 순회의 예



이진 트리의 순회

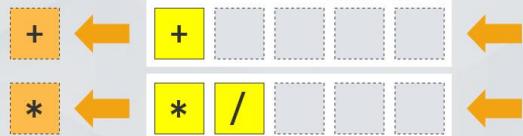
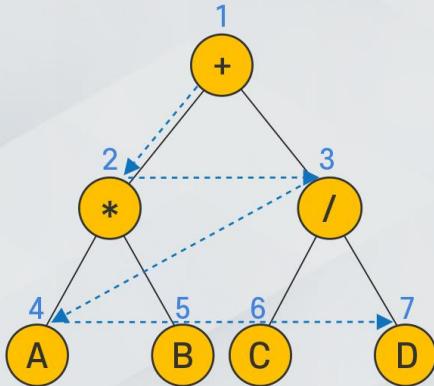


이진 트리의 연산

이진 트리의 순회

3 이진 트리의 레벨 순회

● 레벨 순회의 예

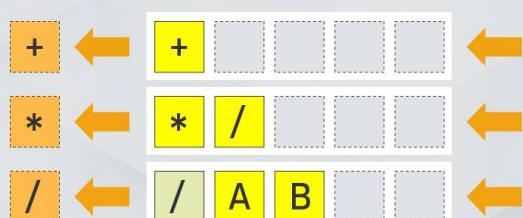
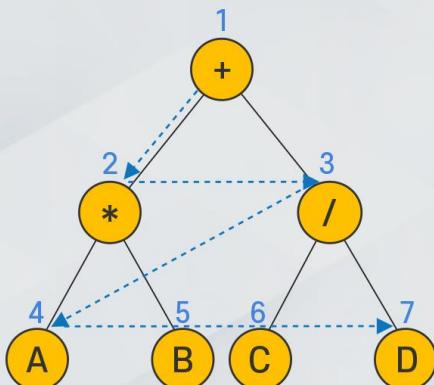


이진 트리의 연산

이진 트리의 순회

3 이진 트리의 레벨 순회

● 레벨 순회의 예



이진 트리의 순회

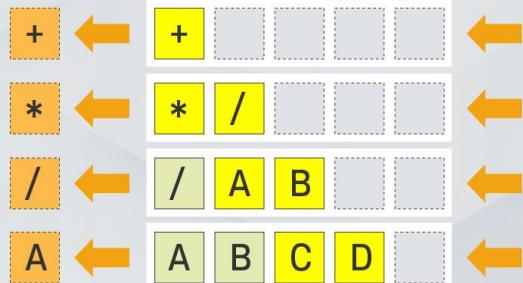
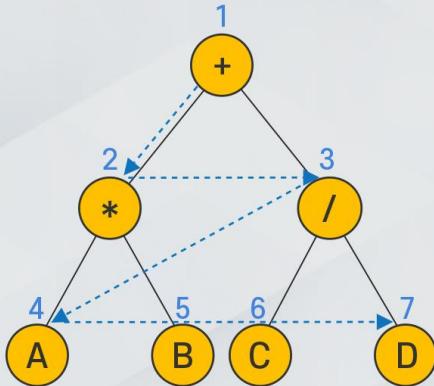


이진 트리의 연산

이진 트리의 순회

3 이진 트리의 레벨 순회

● 레벨 순회의 예

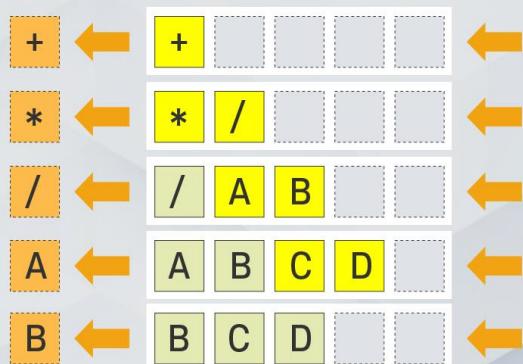
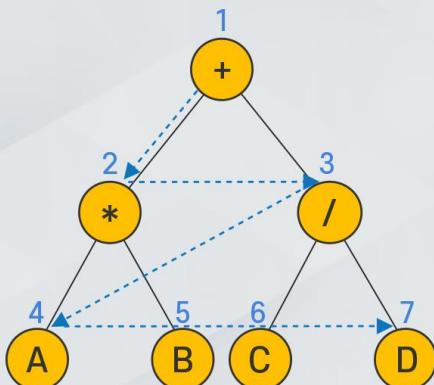


이진 트리의 연산

이진 트리의 순회

3 이진 트리의 레벨 순회

● 레벨 순회의 예



이진 트리의 순회

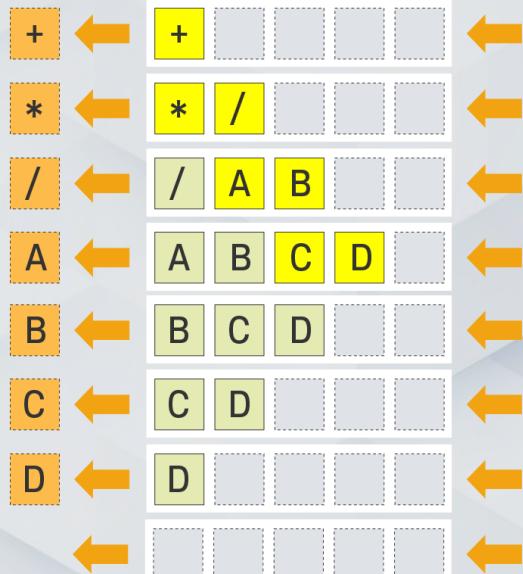
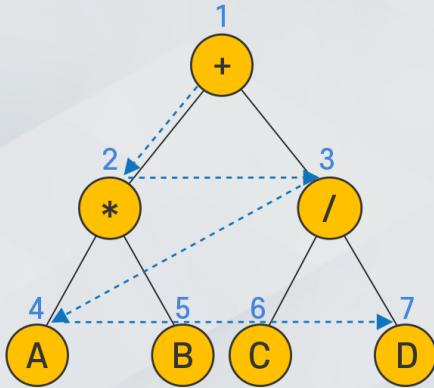


이진 트리의 연산

이진 트리의 순회

3 이진 트리의 레벨 순회

● 레벨 순회의 예



이진 트리의 연산

이진 트리의 순회

3 이진 트리의 레벨 순회

● 레벨 순회 전략

- 1 맨 처음에는 루트 노드만 큐에 있음
- 2 큐에서 노드를 꺼내고 출력
- 3 꺼낸 노드의 왼쪽과 오른쪽 노드를 순서대로 큐에 삽입
- 4 이때, 노드가 없으면 삽입 안함
- 5 큐가 공백이 될 때 까지 진행

이진 트리의 순회



이진 트리의 연산

이진 트리의 순회

3 이진 트리의 레벨 순회

● 레벨 순회 알고리즘

```
def levelorder(root) :  
    queue = CircularQueue()  
    queue.enqueue(root)  
    while not queue.isEmpty() :  
        n = queue.dequeue()  
        if n is not None :  
            print(n.data, end=' ')  
            queue.enqueue(n.left)  
            queue.enqueue(n.right)  
  
# 레벨 순회 함수  
# 공백 큐 준비  
# 처음에는 큐에 루트를 삽입  
# 큐가 공백이 아닐 때 까지  
# 노드를 꺼냄  
# 공백 노드가 아니면  
# 출력(처리)  
# 왼쪽 자식을 먼저 큐에 삽입  
# 오른쪽 자식을 다음으로 큐에 삽입
```

이진 트리의 순회

이진 트리 테스트(링크 표현법)

```

41
42     def testBinaryTree() :
43         print("===== 이진트리 테스트 =====")
44         d = TNode('D', None, None)
45         e = TNode('E', None, None)
46         b = TNode('B', d, e)
47         f = TNode('F', None, None)
48         c = TNode('C', f, None)
49         root = TNode('A', b, c)
50
51         print('\n    In-Order : ', end='')
52         inorder(root)
53         print('\n    Pre-Order : ', end='')
54         preorder(root)
55         print('\n    Post-Order : ', end='')
56         postorder(root)
57         print('\nLevel-Order : ', end='')
58         levelorder(root)
59         print()
60
61
62

```

실습 단계

이진 트리 테스트(링크 표현법)

프로그램 실행결과

이진 트리의 연산들



이진 트리의 연산



이진 트리의 연산들

1 이진 트리와 관련된 연산들

간단한 연산들

- 노드 개수 : 트리의 모든 노드의 수를 세는 연산
- 단말 노드의 수 : 트리의 모든 단말 노드의 수를 세는 연산
- 트리의 높이 : 트리의 높이를 계산

좀 더 복잡한 연산

- 이진 트리가 완전 이진 트리인가?
- 임의의 노드의 레벨을 계산하라.
- 이진 트리가 균형 잡혀 있는가?
- 루트부터 모든 자식 노드까지의 경로의 길이는?
- 좌우로 대칭된 트리를 구하라.

이진 트리의 연산



이진 트리의 연산들

2 노드의 수

아이디어

- 좌우 서브 트리의 노드 수를 먼저 구하고 1을 더한 값
 - 후위 순회
- 좌우 서브 트리는 원래의 트리보다 작음
 - 문제의 크기가 줄어듦
- 종료 조건
 - 서브 트리가 공백 트리이면 → 노드의 수 0을 반환
- 순환 적용

이진 트리의 연산들



이진 트리의 연산

이진 트리의 연산들

2 노드의 수

● 알고리즘

```
def count_node(n) :           # 트리의 노드 수 계산 함수
    if n is None :            # 공백 노드이면(종료 조건)
        return 0               # 0을 반환
    else :                    # 공백 노드가 아니면(순환 호출)
        return 1 + count_node(n.left) + count_node(n.right)
```

이진 트리의 연산

이진 트리의 연산들

3 단말 노드의 수

아이디어

- 노드 개수 연산과 방법이 유사함
- 종료 조건
 - 노드가 공백이면 0 반환
 - 노드가 단말이면 1 반환
- 공백도 아니고 단말 노드도 아니면
 - 좌우 서브 트리의 결과를 단순히 더해서 반환
 - 순환 호출



이진 트리의 연산들

이진 트리의 연산

이진 트리의 연산들

3 단말 노드의 수

● 알고리즘

```

def count_leaf(n) :           # 트리의 단말 노드 수 계산 함수
    if n is None :
        return 0               # 공백 노드이면 → 0 반환
    elif n.isLeaf() :
        return 1               # 단말 노드이면 → 1 반환
    else :
        return count_leaf(n.left) + count_leaf(n.right)           # 순환 호출
    
```

이진 트리의 연산

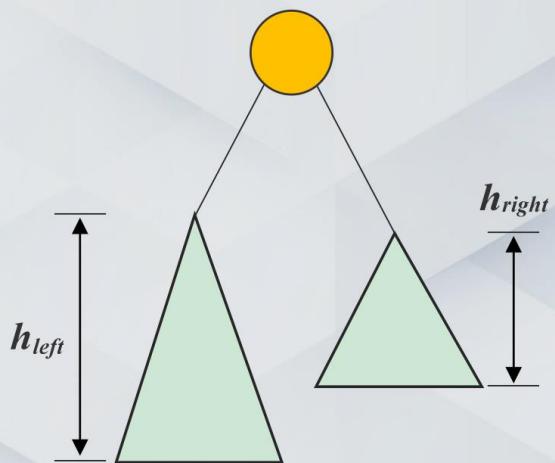
이진 트리의 연산들

4 트리의 높이

아이디어

- 좌우 서브 트리의 높이를 먼저 구함
- 최종 트리의 높이는 더 높은 자식 트리의 높이 +1
- 후위 순회

$$h = \max(h_{left}, h_{right}) + 1$$





이진 트리의 연산들

이진 트리의 연산

4 트리의 높이

● 알고리즘

```

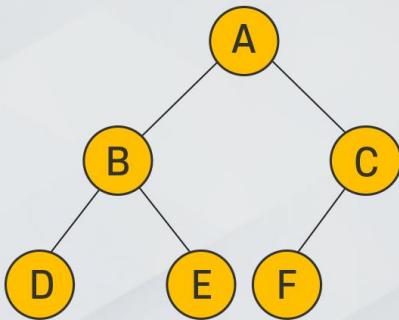
def calc_height(n) :           # 트리의 높이 계산 함수
    if n is None :
        return 0                 # 공백 노드이면 → 높이 0 반환
    hLeft = calc_height(n.left)   # 왼쪽 서브트리의 높이 계산
    hRight = calc_height(n.right) # 오른쪽 서브트리의 높이 계산
    if (hLeft > hRight) :
        return hLeft + 1
    else:                      # 더 높은 값 + 1 반환
        return hRight + 1

```

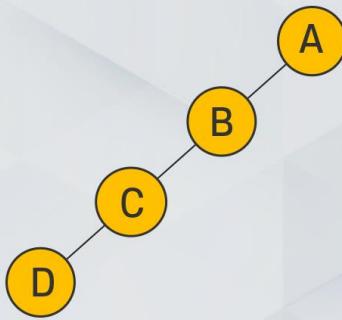
이진 트리의 연산

4 트리의 높이

● 테스트 프로그램



Tree 1



Tree 2

이진 트리의 연산들



파일(F) 편집(E) 보기(V) 프로젝트(P) 디버그(D) 텍스트(S) 블록(R) 도구(T) 확장(X) 찾(W) 도움말(H) 글씨(Ctrl+Q) 교재-파이썬

BinaryTree.py BinTreeRep.py MapChain.py

```

138     # 배열 표기법
139     tree1 = [None, 'A', 'B', 'C', 'D', 'E', 'F']
140     tree2 = [None, 'A', 'B', None, 'C', None, None, 'D']
141
142     # 링크 표기법
143     d = TNode('D', None, None)
144     e = TNode('E', None, None)
145     f = TNode('F', None, None)
146     b = TNode('B', d, e)
147     c = TNode('C', f, None)
148     tree1 = TNode('A', b, c)
149
150     print(" 노드의 개수 = %d개" % count_node(tree1))
151     print(" 단말의 개수 = %d개" % count_leaf(tree1))
152     print(" 트리의 높이 = %d" % calc_height(tree1))
153     print()
154
155     d = TNode('D', None, None)
156     c = TNode('C', d, None)
157     b = TNode('B', c, None)
158     tree2 = TNode('A', b, None)
159

```

206% 문제가 겹쳐되지 않을 때

출석 출석 보기 선택(S)

도구 목록 영업장 출석

실습 단계

a b c d가 순서적으로 왼쪽으로 내려가는 형태로 링크 표현법으로 나타낼 수 있음

TNode 클래스에서 isLeaf라는 멤버 함수를 만들었음

두개 중 하나라도 만족하지 않으면 False를 반환

멤버 함수는 현재 노드의 left와 right를 체크해서 둘 다 None인 경우에 True를 반환

else까지 오면 이 노드는 공백 노드도 아니고 자식이 반드시 하나는 있는 노드가 됨

더하기 1을 해줄 필요가 없음

왼쪽 서브 트리의 단말 노드의 개수에 오른쪽 서브 트리의 단말 노드의 개수를 더해서 그 합을 반환

왼쪽 서브 트리가 높으면 그 높은 것에 1을 더함

알고리즘의 실행 결과