Chapter. 07

고급 탐색 알고리즘

기초 문제풀이

FAST CAMPUS
ONLINE
유형별 문제풀이

강사. 나동빈

Chapter. 07

고급 탐색 알고리즘(기초 문제풀이)

FAST CAMPUS
ONLINE
나동빈 강사.

I 혼자 힘으로 풀어 보기

트리 순회: https://www.acmicpc.net/problem/1991

문제 제목: 트리 순회

문제 난이도: 하(Easy)

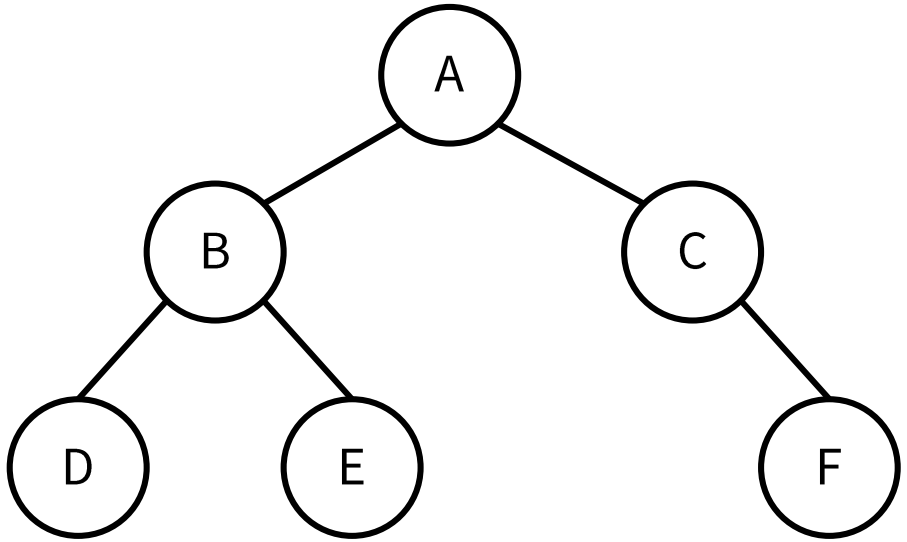
문제 유형: 트리, 구현

추천 풀이 시간: 20분

트리 순회: https://www.acmicpc.net/problem/1991

문제 풀이 핵심 아이디어

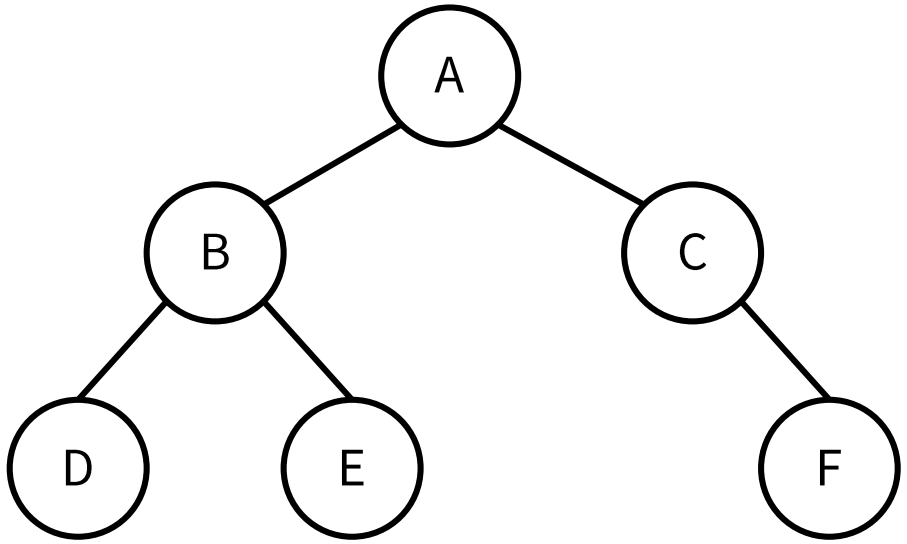
- 전위 순회: 루트 → 왼쪽 자식 → 오른쪽 자식
- 중위 순회: 왼쪽 자식 → 루트 → 오른쪽 자식
- 후위 순회: 왼쪽 자식 → 오른쪽 자식 → 루트



트리 순회: https://www.acmicpc.net/problem/1991

I 문제 풀이 핵심 아이디어

- 전위 순회: **루트 출력** → 왼쪽 자식 → 오른쪽 자식

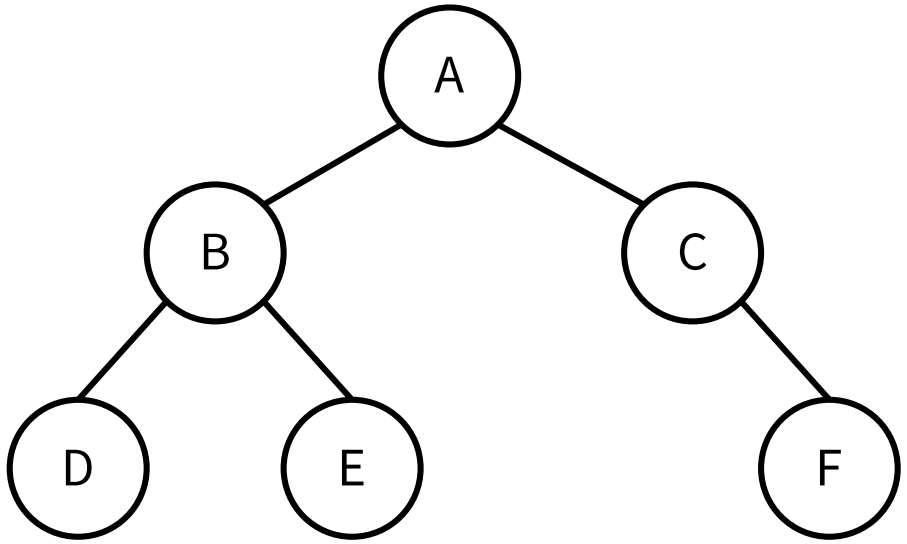


순회 결과: A – B – D – E – C – F

트리 순회: https://www.acmicpc.net/problem/1991

I 문제 풀이 핵심 아이디어

- 중위 순회: 왼쪽 자식 → **루트 출력** → 오른쪽 자식

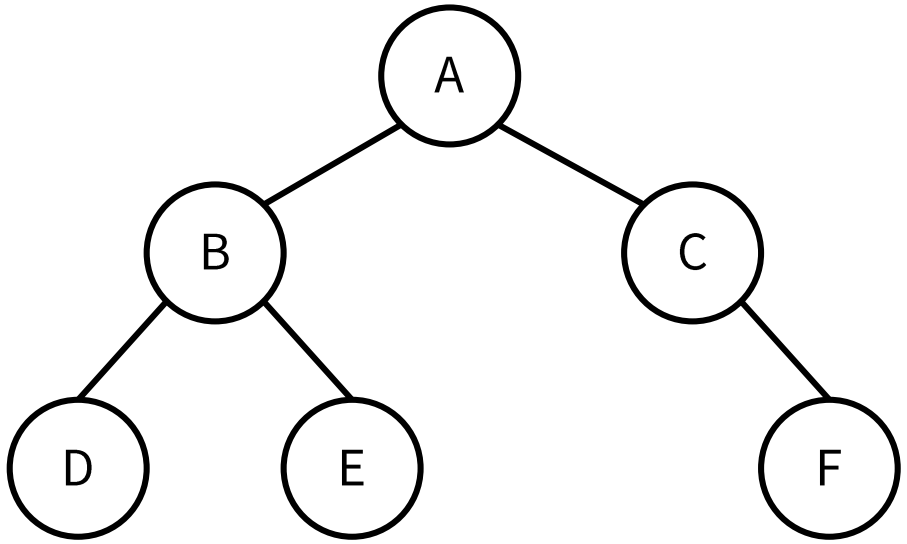


순회 결과: D – B – E – A – C – F

트리 순회: https://www.acmicpc.net/problem/1991

I 문제 풀이 핵심 아이디어

- 후위 순회: 왼쪽 자식 → 오른쪽 자식 → **루트 출력**



순회 결과: D – E – B – F – C – A

트리 순회: https://www.acmicpc.net/problem/1991

| 소스코드

```python
class Node:
    def __init__(self, data, left_node, right_node):
        self.data = data
        self.left_node = left_node
        self.right_node = right_node

def pre_order(node):
    print(node.data, end='')
    if node.left_node != '.':
        pre_order(tree[node.left_node])
    if node.right_node != '.':
        pre_order(tree[node.right_node])

def in_order(node):
    if node.left_node != '.':
        in_order(tree[node.left_node])
    print(node.data, end='')
    if node.right_node != '.':
        in_order(tree[node.right_node])
```

```python
def post_order(node):
    if node.left_node != '.':
        post_order(tree[node.left_node])
    if node.right_node != '.':
        post_order(tree[node.right_node])
    print(node.data, end='')

n = int(input())
tree = {}
for i in range(n):
    data, left_node, right_node = input().split()
    tree[data] = Node(data, left_node, right_node)

pre_order(tree['A'])
print()
in_order(tree['A'])
print()
post_order(tree['A'])
```

I 혼자 힘으로 풀어 보기

트리의 높이와 너비: https://www.acmicpc.net/problem/2250

문제 제목: 트리의 높이와 너비

문제 난이도: 중(Medium)

문제 유형: 트리, 구현

추천 풀이 시간: 50분

트리의 높이와 너비: https://www.acmicpc.net/problem/2250

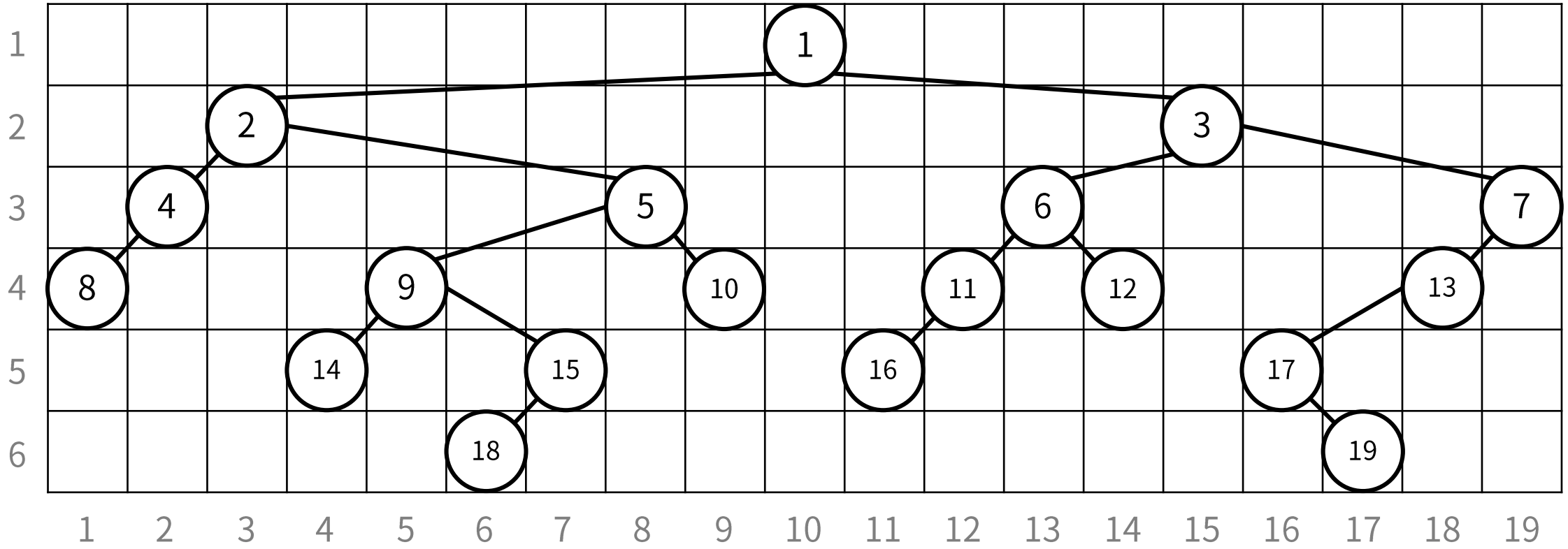
I 문제 풀이 핵심 아이디어

- 중위 순회를 이용하면 X축을 기준으로 왼쪽부터 방문한다는 특징이 있습니다.
- 이 문제는 중위 순회 알고리즘을 이용하고, 추가적으로 Level 값을 저장하도록 하여 문제를 해결할 수 있습니다.

트리의 높이와 너비: https://www.acmicpc.net/problem/2250

I 문제 풀이 핵심 아이디어

- **중위 순회 결과**: 8 – 4 – 2 – 14 – 9 – 18 – 15 – 5 – 10 – 1 – 16 – 11 – 6 – 12 – 3 – 17 – 19 – 13 – 7



트리의 높이와 너비: https://www.acmicpc.net/problem/2250

| 소스코드

```python
class Node:
    def __init__(self, number, left_node, right_node):
        self.parent = -1
        self.number = number
        self.left_node = left_node
        self.right_node = right_node

def in_order(node, level):
    global level_depth, x
    level_depth = max(level_depth, level)
    if node.left_node != -1:
        in_order(tree[node.left_node], level + 1)
    level_min[level] = min(level_min[level], x)
    level_max[level] = max(level_max[level], x)
    x += 1
    if node.right_node != -1:
        in_order(tree[node.right_node], level + 1)

n = int(input())
tree = {}
level_min = [n]
level_max = [0]
root = -1
x = 1
level_depth = 1

for i in range(1, n + 1):
    tree[i] = Node(i, -1, -1)
    level_min.append(n)
    level_max.append(0)

for _ in range(n):
    number, left_node, right_node = map(int, input().split())
    tree[number].left_node = left_node
    tree[number].right_node = right_node
    if left_node != -1:
        tree[left_node].parent = number
    if right_node != -1:
        tree[right_node].parent = number

for i in range(1, n + 1):
    if tree[i].parent == -1:
        root = i

in_order(tree[root], 1)

result_level = 1
result_width = level_max[1] - level_min[1] + 1
for i in range(2, level_depth + 1):
    width = level_max[i] - level_min[i] + 1
    if result_width < width:
        result_level = i
        result_width = width

print(result_level, result_width)
```