다양한 자료구조 문제 풀이

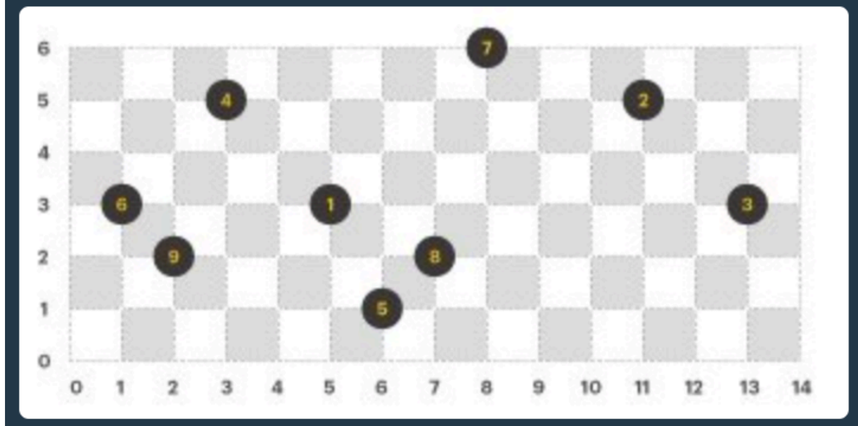
23.12.4 송이

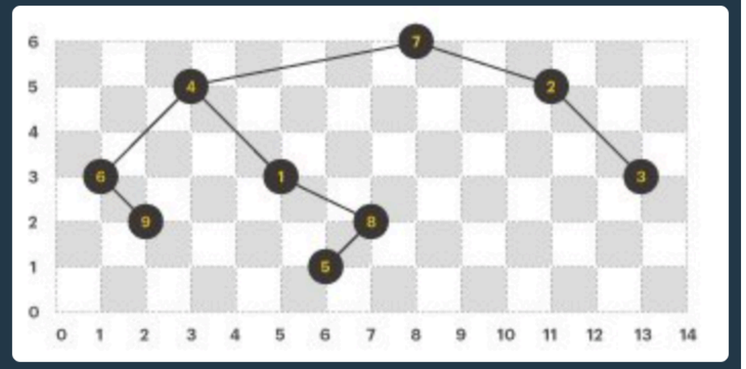
문제 47번
이해하기

라이언이 구상한(그리고 아마도 라이언만 즐거울만한) 게임은, 카카오 프렌즈를 두 팀으로 나누고, 각 팀이 같은 곳을 다른 순서로 방문하도록 해서 먼저 순회를 마친 팀이 승리하는 것이다.

그냥 지도를 주고 게임을 시작하면 재미가 덜해지므로, 라이언은 방문할 곳의 2차원 좌표 값을 구하고 각 장소를 이진트리의 노드가 되도록 구성한 후, 순회 방법을 힌트로 주어 각 팀이 스스로 경로를 찾도록 할 계획이다.

라이언의 규칙에 맞게 이진트리의 노드만 좌표 평면에 그리면 다음과 같다. (이진트리의 각 노드에는 1부터 N까지 순서대로 번호가 붙어있다.)



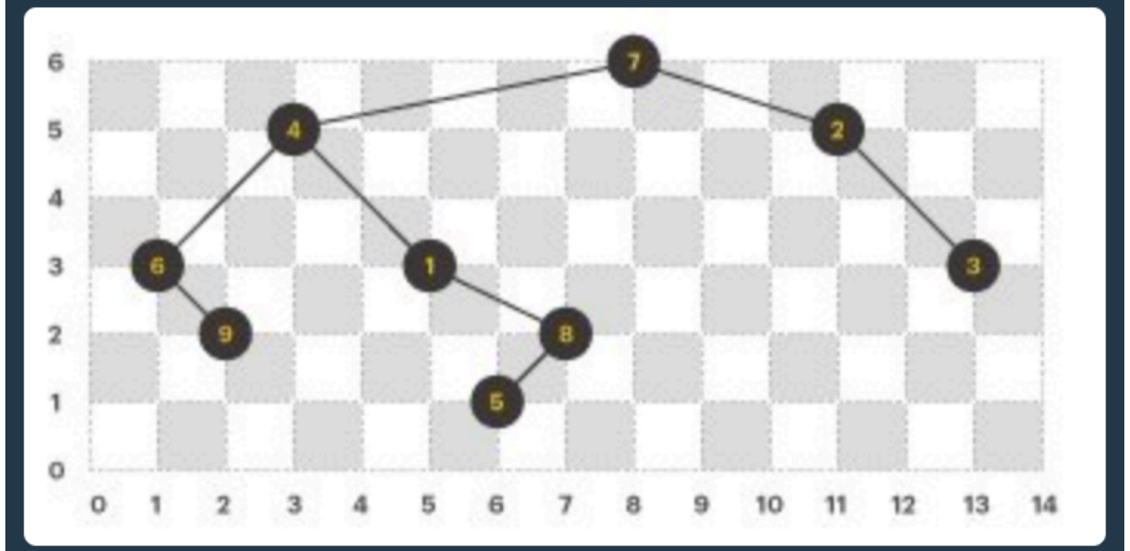


라이언은 아래와 같은 특별한 규칙으로 트리 노드들을 구성한다.

- 트리를 구성하는 모든 노드의 x, y 좌표 값은 정수이다.
- 모든 노드는 서로 다른 x값을 가진다.
- 같은 레벨(level)에 있는 노드는 같은 y 좌표를 가진다.
- 자식 노드의 y 값은 항상 부모 노드보다 작다.
- 임의의 노드 V의 왼쪽 서브 트리(left subtree)에 있는 모든 노드의 x값은 V의 x값보다 작다.
- 임의의 노드 V의 오른쪽 서브 트리(right subtree)에 있는 모든 노드의 x값은 V의 x값보다 크다.

문제 47번

아이디어 소개



위 이진트리에서 전위 순회(preorder), 후위 순회(postorder)를 한 결과는 다음과 같고, 이것은 각 팀이 방문해야 할 순서를 의미한다.

- 전위 순회 : 7, 4, 6, 9, 1, 8, 5, 2, 3
- 후위 순회 : 9, 6, 5, 8, 1, 4, 3, 2, 7

제한사항

- nodeinfo는 이진트리를 구성하는 각 노드의 좌표가 1번 노드부터 순서대로 들어있는 2차원 배열이다.
  - nodeinfo의 길이는 `1` 이상 `10,000` 이하이다.
  - nodeinfo[i] 는 i + 1번 노드의 좌표이며, [x축 좌표, y축 좌표] 순으로 들어있다.
  - 모든 노드의 좌표 값은 `0` 이상 `100,000` 이하인 정수이다.
  - 트리의 깊이가 `1,000` 이하인 경우만 입력으로 주어진다.
  - 모든 노드의 좌표는 문제에 주어진 규칙을 따르며, 잘못된 노드 위치가 주어지는 경우는 없다.

입출력 예

nodeinfo	result
[[5,3],[11,5],[13,3],[3,5],[6,1],[1,3],[8,6],[7,2],[2,2]]	[[7,4,6,9,1,8,5,2,3],[9,6,5,8,1,4,3,2,7]]

문제 47
접근방법

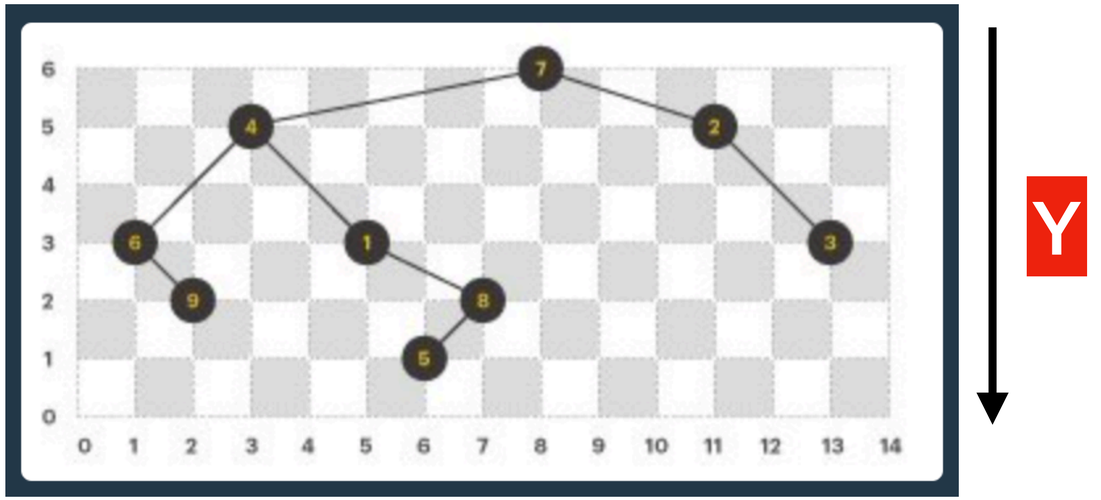
- 주어진 조건에 맞게 트리 구성하기
- 전위 순회 / 후위 순회 함수를 만들기
- 노드를 기준에 맞게 정렬하고 트리를 생성하기
- 각 순회의 결과를 합치고 결과를 반환하기

문제 47

노드를 기준에 맞게 정렬하고 트리를 생성하기

- nodeinfo[i] 는 i + 1번 노드의 좌표이며, [x축 좌표, y축 좌표] 순으로 들어있다.

nodeinfo
[[5,3],[11,5],[13,3],[3,5],[6,1],[1,3],[8,6],[7,2],[2,2]]




```
[[5, 3, 1], [11, 5, 2], [13, 3, 3], [3, 5, 4], [6, 1, 5], [1, 3, 6], [8, 6, 7], [7, 2, 8], [2, 2, 9]]
```

[x,y,순서]

```
[[8, 6, 7], [11, 5, 2], [3, 5, 4], [5, 3, 1], [13, 3, 3], [1, 3, 6], [7, 2, 8], [2, 2, 9], [6, 1, 5]]
```

y 값으로 sorting

문제 47

노드를 기준에 맞게 정렬하고 트리를 생성하기

```
def solution(nodeinfo):
    nodeinfo = [[*info , idx+1] for idx, info in enumerate(nodeinfo)]
    nodeinfo = sorted(nodeinfo, key=lambda x : x[1] , reverse = True)


    root = node(nodeinfo[0])
    for info in nodeinfo[1:]:
        addnode(root, info)
```

문제 47

트리 구성하기

노드 클래스 선언

```
class Node:
    def __init__(self, info):
        self.number = info[2]
        self.data = info[:2]
        self.right = None
        self.left = None
```

[x,y,순서]

트리 구조에 노드를 추가하는 함수

```
def addnode(root, info):
    if info[0] > root.data[0]:
        if not root.right : root.right = node(info)
        else: addnode(root.right,info)
    elif info[0] < root.data[0]:
        if not root.left: root.left = node(info)
        else:addnode(root.left,info)
```

값이 더 크면 오른쪽에 추가

오른쪽 노드가 None이라면 새로운 node를 만들고 추가

값이 작으면 왼쪽에 추가

문제 47
순회 함수 만들기

```
def preorder(root, order):
    if root !=None:
        order.append(root.number)
        preorder(root.left,order)
        preorder(root.right,order)
```

```
def postorder(root, order):
    if root !=None:
        preorder(root.left,order)
        preorder(root.right,order)
        order.append(root.number)
```

문제 47

각 순회의 결과를 합쳐 결과 반환하기

```
preorderList = []
preorder(root,preorderList)


postorderList = []
postorder(root,postorderList)


return [preorderList,postorderList]
```