길찾기 게임

전무로 승진한 라이언은 기분이 너무 좋아 프렌즈를 이끌고 특별 휴가를 가기로 했다.

내친김에 여행 계획까지 구상하던 라이언은 재미있는 게임을 생각해냈고 역시 전무로 승진할만한 인재라고 스스로에게 감탄했다.
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.
라이언은 아래와 같은 특별한 규칙으로 트리 노드들을 구성한다.

① 트리를 구성하는 모든 노드의 x, y 좌표 값은 정수이다.
② 모든 노드는 서로 다른 x값을 가진다.
③ 같은 레벨(level)에 있는 노드는 같은 y 좌표를 가진다.
④ 자식 노드의 y 값은 항상 부모 노드보다 작다.
⑤ 임의의 노드 V의 왼쪽 서브 트리(left subtree)에 있는 모든 노드의 x값은 V의 x값보다 작다.
⑥ 임의의 노드 V의 오른쪽 서브 트리(right subtree)에 있는 모든 노드의 x값은 V의 x값보다 크다.

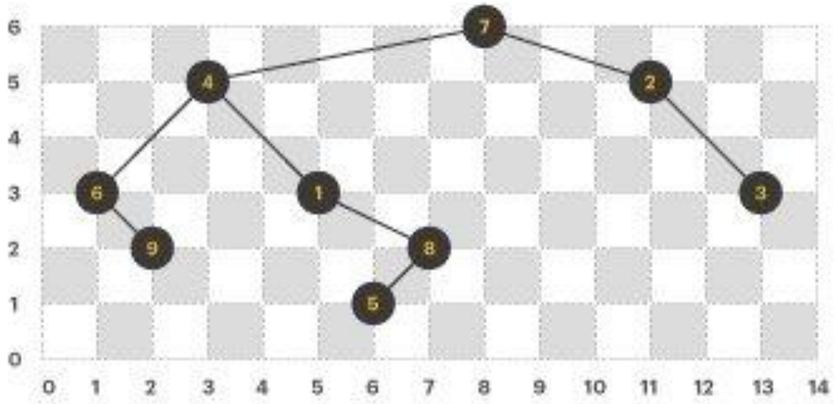
… 중략 …

이진트리를 구성하는 노드들의 좌표가 담긴 배열 nodeinfo가 매개변수로 주어질 때,
노드들로 구성된 이진트리를 전위 순회, 후위 순회한 결과를 2차원 배열에 순서대로 담아 return 하도록 solution 함수를 완성하자.

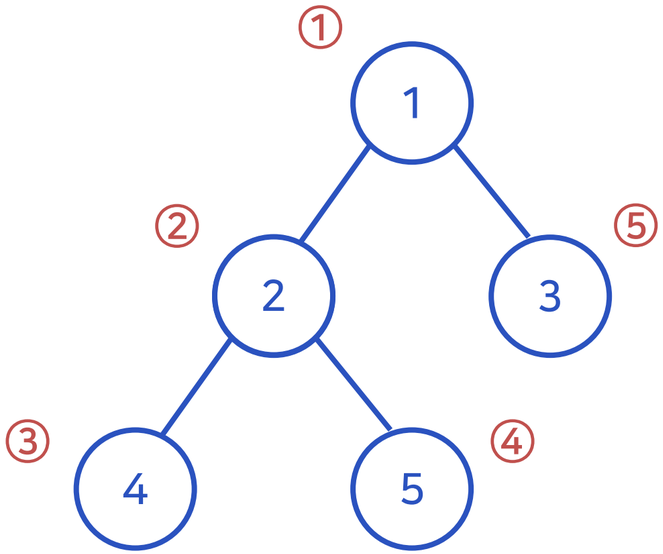
길찾기 게임

[제한 조건]

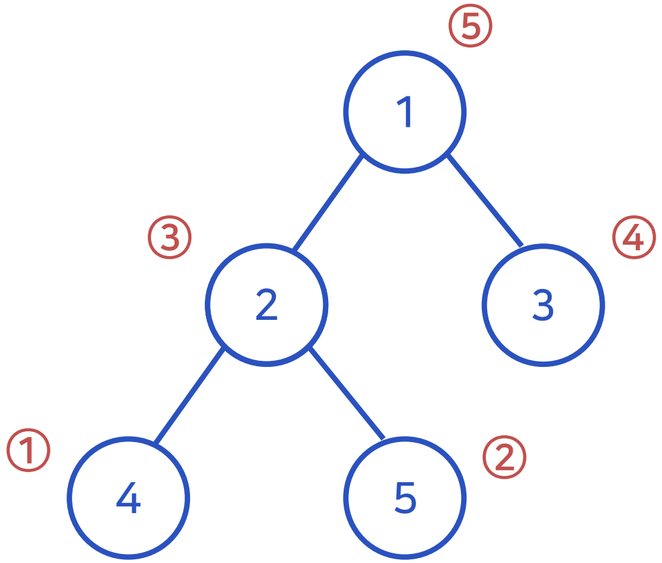
- nodeinfo는 이진트리를 구성하는 각 노드의 좌표가 1번 노드부터 순서대로 들어있는 2차원 배열이다.
  - nodeinfo의 길이는 1 이상 10,000 이하이다.
  - nodeinfo[i] 는 i + 1번 노드의 좌표이며, [x축 좌표, y축 좌표] 순으로 들어있다.
  - 모든 노드의 좌표 값은 0 이상 100,000 이하인 정수이다.
  - 트리의 깊이가 1,000 이하인 경우만 입력으로 주어진다.
  - 모든 노드의 좌표는 문제에 주어진 규칙을 따르며, 잘못된 노드 위치가 주어지는 경우는 없다.



전위 순회(Pre-order)



후위 순회(Post-order)



```python
import sys
sys.setrecursionlimit(10**6) # 런타임에러 방지

# Tree 클래스 생성하기
def solution(nodeinfo):
    # 이진 트리에 데이터 저장
    root = None

    for i in range(len(nodeinfo)):
        nodeinfo[i].append(i+1)

    # y좌표를 기준으로 내림차순 정렬
    nodeinfo = sorted(nodeinfo, key= lambda x:x[1], reverse=True)

    # enumerate 함수로 for문 반복
    for i,node in enumerate(nodeinfo):
        newTree = Tree()
        newTree.index = node[2]
        newTree.data = node

        if root == None:
            root = newTree
        else:
            curTree = root

        while 1:
            # 새로 삽입하려는 데이터가 원래 데이터보다 크다면 오른쪽에 삽입
            if curTree.data[0] < newTree.data[0]:
                # 현재 노드의 오른쪽 노드가 null이라면
                if curTree.right == None:
                    curTree.right = newTree
                    newTree.parent = curTree
                    break
                # 현재 노드의 오른쪽 노드를 현재 노드로 다시 정의한다
                else:
                    curTree = curTree.right

            # 새로 삽입하려는 데이터가 원래 데이터보다 작다면 왼쪽에 삽입
            else:
                # 현재 노드의 왼쪽 노드가 null이라면
                if curTree.left == None:
                    curTree.left = newTree
                    newTree.parent = curTree
                    break
                # 현재 노드의 왼쪽 노드를 현재 노드로 다시 정의한다
                else:
                    curTree = curTree.left

    # 최종 순회 후 결과값을 저장
    answer = [preOrder(root,[]), postOrder(root,[])]
    return answer

    return vector
```

THANK YOU