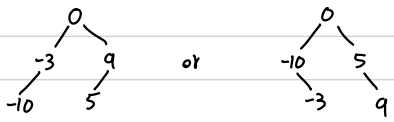
① 이해

· 정렬된 배열을 이진 탐색 트리(BST)로 변환하라. (높이 균형적인 BST)

ex) [-10, -3, 0, 5, 9]



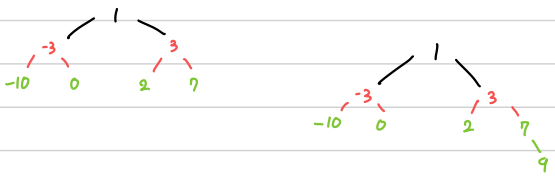
② 계획 "재귀"

· 높이 균형적인 트리를 만들기 위해서는 배열의 제일 중간값이 루트가 돼야함"

· 가운데를 기준으로 왼/오른쪽으로 각자 BST를 만들어 아래→위 서브트리를 붙혀가며 전체 BST 완성!!

· 노드가 1, 2개 일 때 BST 만드는 코드 작성 → 노드 개수가 넘어갈 경우 다시 왼/오른쪽으로 나눠 재귀

ex) [-10, -3, 0, 1, 2, 3, 7] / [-10, -3, 0, 1, 2, 3, 7, 9]



③ 풀이코드 (1) "재귀"

```python
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def sortedArrayToBST(self, nums: List[int]) -> Optional[TreeNode]:
        if len(nums) == 1 : return TreeNode(nums[0])
        if len(nums) == 2 :
            node = TreeNode(nums[0])
            node.right = TreeNode(nums[1])
            return node

        center = len(nums)//2

        node = TreeNode(nums[center])

        node.left = self.sortedArrayToBST(nums[:center])
        node.right = self.sortedArrayToBST(nums[center+1:])

        return node
```

→ 트리 노드가 1개일 경우 BST 리턴

→ 트리 노드가 2개일 경우 BST 리턴

왼쪽, 오른쪽 서브트리 → BST로

③ 풀이코드 (2)

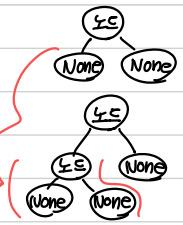
재귀

(더 간단)

```
# Definition for a binary tree node.
# class TreeNode:
#     def __init__(self, val=0, left=None, right=None):
#         self.val = val
#         self.left = left
#         self.right = right
class Solution:
    def sortedArrayToBST(self, nums: List[int]) -> Optional[TreeNode]:
        if not nums : return None → 트리가 빈 트리 일때만 처리

        center = len(nums)//2

        node = TreeNode(nums[center])

        node.left = self.sortedArrayToBST(nums[:center])
        node.right = self.sortedArrayToBST(nums[center+1:])

        return node
```

- 풀이코드 (1)과 풀이 방식은 재귀로 같으나 - 트리 노드가 1개인 경우

  트리 노드가 2개인 경우



위의 방식으로 처리할 수 있기 때문에 트리의 노드가 1, 2개 인경우로 나눌 필요 없이 트리가 빈 트리인 경우만 생각해주면 된다.

④ 비고

- 동일한 계산이 반복될 경우 "재귀"를 이용한 풀어 본 적 생각!!

- 풀이코드 (1) → (2) 처럼 풀이를 간소화 할 수 있는지 다시 생각해보기!!