트리문제풀이

작성 일시	@June 18, 2023 3:18 PM
자료	https://leetcode.com/problems/find-a-corresponding-node-of-a-binary-tree-in-a-clone-of-that-tree/
# 주차	4

1379. Find a Corresponding Node of a Binary Tree in a Clone of That Tree

적용이론

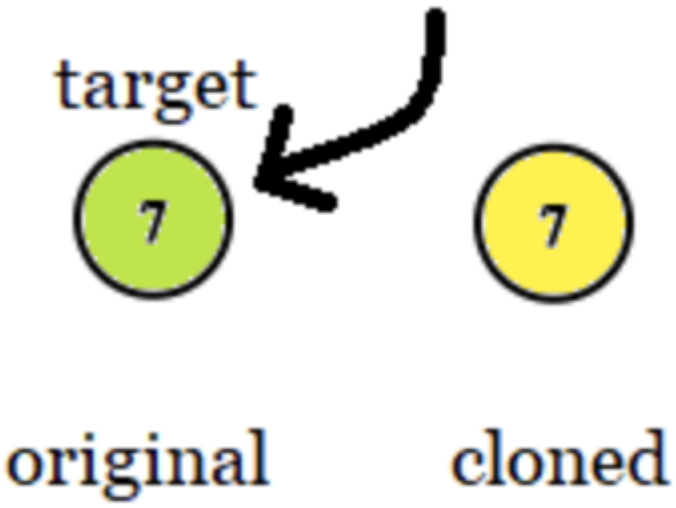
- Binary Tree(이진트리)
- Recursion(재귀)
- short circuit evaluation(단축평가)
- object comparison(객체비교)

```
/**
 * Definition for a binary tree node.
 * function TreeNode(val) {
 *     this.val = val;
 *     this.left = this.right = null;
 * }
 */
/**
 * @param {TreeNode} original
 * @param {TreeNode} cloned
 * @param {TreeNode} target
 * @return {TreeNode}
 */

var getTargetCopy = function (original, cloned, target) {
  if (original !== null) {
    if (JSON.stringify(original) === JSON.stringify(target)) {
      return cloned;
    }
    return (
      getTargetCopy(original.left, cloned.left, target) ||
```

```
      getTargetCopy(original.right, cloned.right, target)
    );
  }
};
```

Example 2:

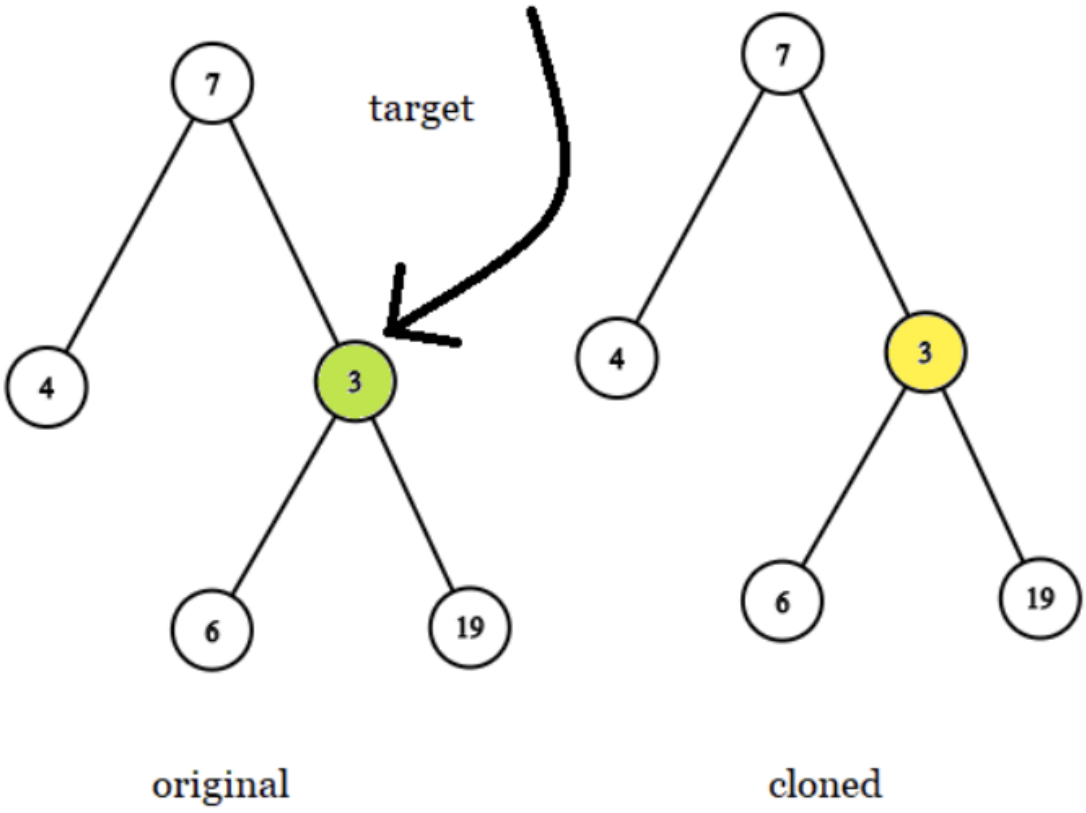



```
var getTargetCopy = function (original, cloned, target) {
  if (original !== null) {
    if (JSON.stringify(original) === JSON.stringify(target)) {
      return cloned;
    }
    return (
      getTargetCopy(original.left, cloned.left, target) ||
      getTargetCopy(original.right, cloned.right, target)
    );
  }
};
```

1. getTargetNode(original, cloned, target) 함수 호출
   original node = 7
   cloned node = 7
   target node = 7
2. 현재 노드가 Null이면 다음 로직 돌필요 없음
3. 현재 노드가 타겟노드와 같다면

4. 현재 노드의 복제노드 리턴

Example 1:



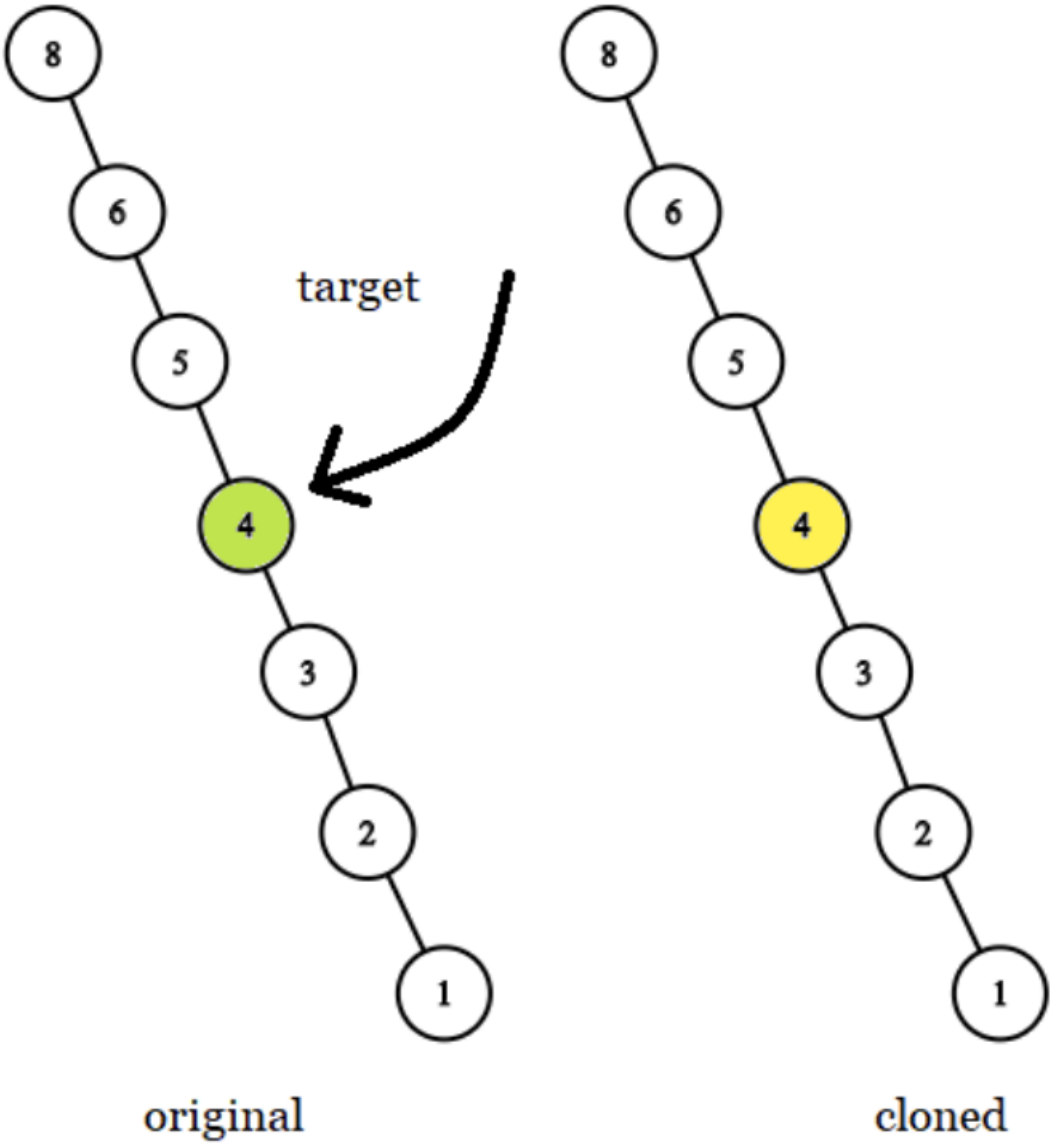

```
var getTargetCopy = function (original, cloned, target) {
  if (original !== null) {
    if (JSON.stringify(original) === JSON.stringify(target)) {
      return cloned;
    }
    return (
      getTargetCopy(original.left, cloned.left, target) ||
      getTargetCopy(original.right, cloned.right, target)
    );
  }
};
```

1. getTargetNode(original, cloned, target) 함수 호출
   original node = 7

cloned node = 7
target node = 3

2. 현재 노드가 Null이면 다음 로직 돌필요 없음
3. 현재 노드와 타겟노드가 같지 않으므로
4. 다음 로직 돈다.
5. 현재노드의 왼쪽 노드값 먼저 검사
   return getTargetNode(left) || getTargetNode(right)
   return getTargetNode(left-left) || getTargetNode(left-right)
   → return undefined || getTargetNode(right)
6. 왼쪽 노드 값은 falsy 하므로 오른쪽 노드 검사
   return undefined || getTargetNode(right)
   → return undefined || **cloned**;

Example 3:

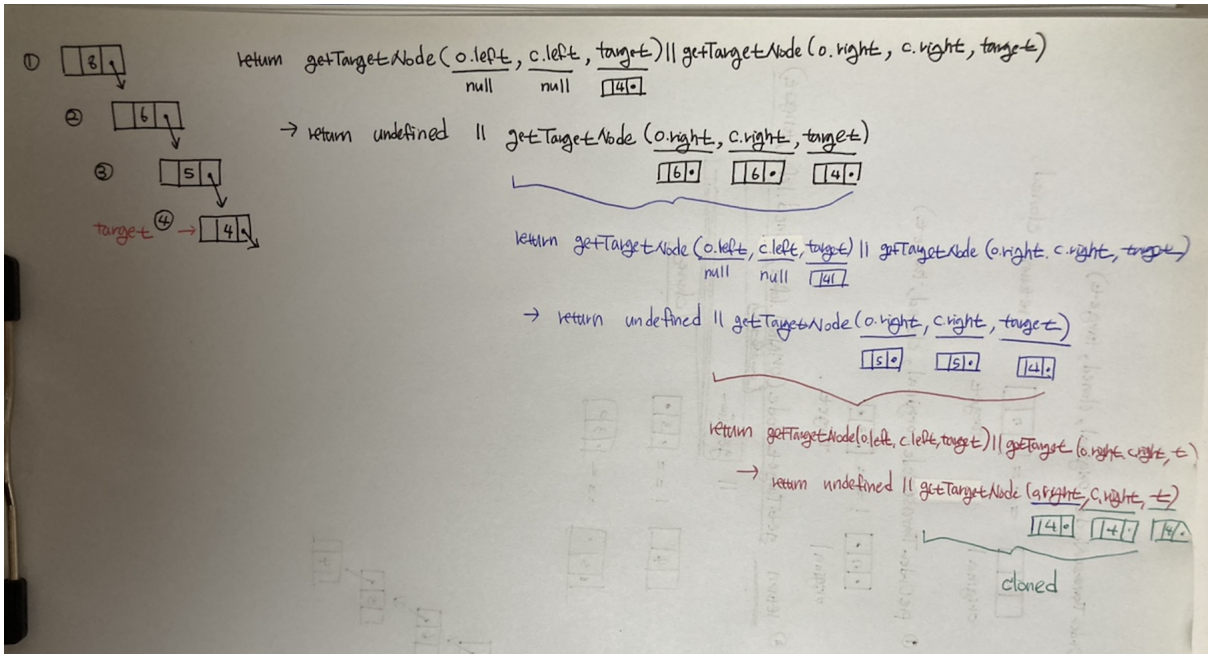



```
var getTargetCopy = function (original, cloned, target) {
  if (original !== null) {
    if (JSON.stringify(original) === JSON.stringify(target)) {
      return cloned;
    }
    return (
      getTargetCopy(original.left, cloned.left, target) ||
      getTargetCopy(original.right, cloned.right, target)
    );
  }
};
```

1. getTargetNode(original, cloned, target) 함수 호출
   original node = 8

cloned node = 8
target node = 4

2. 현재 노드가 Null이면 다음 로직 돌필요 없음
3. 현재 노드와 타겟노드가 같지 않으므로
4. 다음 로직 돈다.
5. 현재노드의 왼쪽 노드값 먼저 검사
   return getTargetNode(left) || getTargetNode(right)
   → return undefined || getTargetNode(right)
6. 왼쪽 노드 값은 falsy 하므로 오른쪽 노드 검사
   return undefined || getTargetNode(right) // 6
   return getTargetNode(right-left) || getTargetNode(right-right) // 5
   →return undfined || getTargetNode(right-right)
   return getTargetNode(right-right-left) || getTargetNode(right-right-right);



1161. Maximum Level Sum of a Binary Tree

적용이론

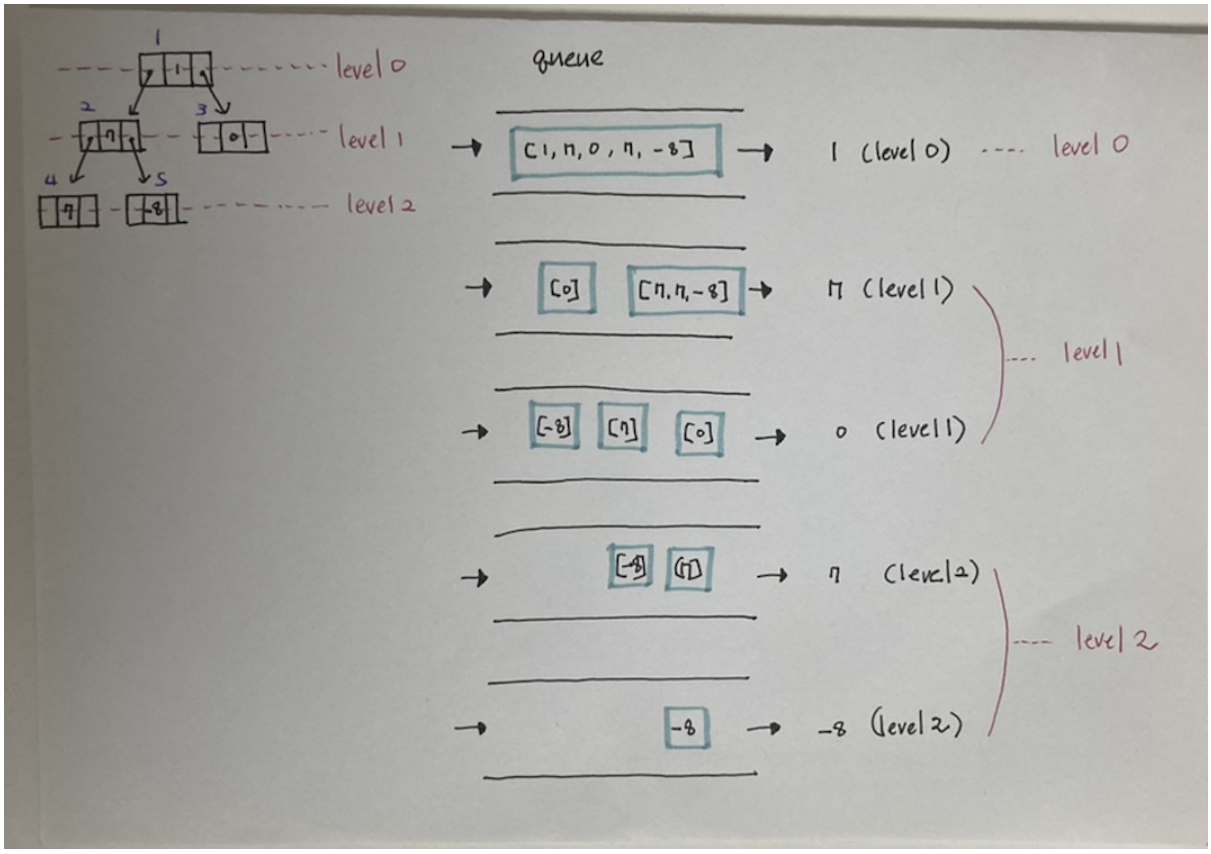
- Binary Tree(이진트리)

- Breadth First Traversal(너비 우선 순회)
- Level Order Traversal
- Queue

```
/**
 * @param {TreeNode} root
 * @return {number}
 */
var maxLevelSum = function (root) {
  // array of the sum of the values each level
  const levelSum = levelOrder(root).map((el) =>
    el.reduce((acc, cur) => acc + cur, 0)
  );

  // get the max value of array levelSum
  const max = Math.max(...levelSum);

  // get the index of the max value
  const indexOfMaxSum = levelSum.indexOf(max) + 1;
  return indexOfMaxSum;
};
function levelOrder(root) {
  if (!root) return [];
  let result = [];
  let queue = [root];
  while (queue.length !== 0) {
    let subarr = [];
    const n = queue.length;
    for (let i = 0; i < n; i++) {
      let node = queue.pop();
      subarr.push(node.val);
      if (node.left) queue.unshift(node.left);
      if (node.right) queue.unshift(node.right);
    }
    result.push(subarr);
  }
  // root = [1,7,0,7,-8,null,null]
  // result = [[1], [7,0], [7,-8]]
  return result;
}
```


level 0
level 1
level 2
queue
[1, ㄲ, 0, ㄲ, -8]
1 (level 0)
level 0
[0]
[ㄲ, ㄲ, -8]
ㄲ (level 1)
level 1
[-8]
[ㄲ]
[0]
0 (level 1)
[-8]
[ㄲ]
ㄲ (level 2)
level 2
-8
-8 (level 2)