Problem 1490: Clone N-ary Tree

Problem Information

Difficulty: Medium
Acceptance Rate: 83.16%
Paid Only: Yes
Tags: Hash Table, Tree, Depth-First Search, Breadth-First Search

Problem Description

Given a `root` of an N-ary tree, return a [**deep copy**](https://en.wikipedia.org/wiki/Object_copying#Deep_copy) (clone) of the tree.

Each node in the n-ary tree contains a val (`int`) and a list (`List[Node]`) of its children.

class Node { public int val; public List<Node> children; }

Nary-Tree input serialization is represented in their level order traversal, each group of children is separated by the null value (See examples).

Example 1:

Input: root = [1,null,3,2,4,null,5,6] **Output:** [1,null,3,2,4,null,5,6]

Example 2:

Input: root = [1,null,2,3,4,5,null,null,6,7,null,8,null,9,10,null,null,11,null,12,null,13,null,null,14] **Output:** [1,null,2,3,4,5,null,null,6,7,null,8,null,9,10,null,null,11,null,12,null,13,null,null,14]

Constraints:

* The depth of the n-ary tree is less than or equal to `1000`. * The total number of nodes is between `[0, 104]`.

Follow up: Can your solution work for the [graph problem](https://leetcode.com/problems/clone-graph/)?

Code Snippets

C++:

```
/*
// Definition for a Node.
class Node {
public:
int val;
vector<Node*> children;

Node() {}

Node(int _val) {
val = _val;
}

Node(int _val, vector<Node*> _children) {
val = _val;
children = _children;
}
};
*/

class Solution {
public:
Node* cloneTree(Node* root) {

}
};
```

Java:

```
/*
// Definition for a Node.
```

```
class Node {
public int val;
public List<Node> children;


public Node() {
children = new ArrayList<Node>();
}

public Node(int _val) {
val = _val;
children = new ArrayList<Node>();
}

public Node(int _val,ArrayList<Node> _children) {
val = _val;
children = _children;
}
};
*/

class Solution {
public Node cloneTree(Node root) {

}
}
```

Python3:

```
"""
# Definition for a Node.
class Node:
def __init__(self, val: Optional[int] = None, children:
Optional[List['Node']] = None):
self.val = val
self.children = children if children is not None else []
"""

class Solution:
def cloneTree(self, root: 'Node') -> 'Node':
```