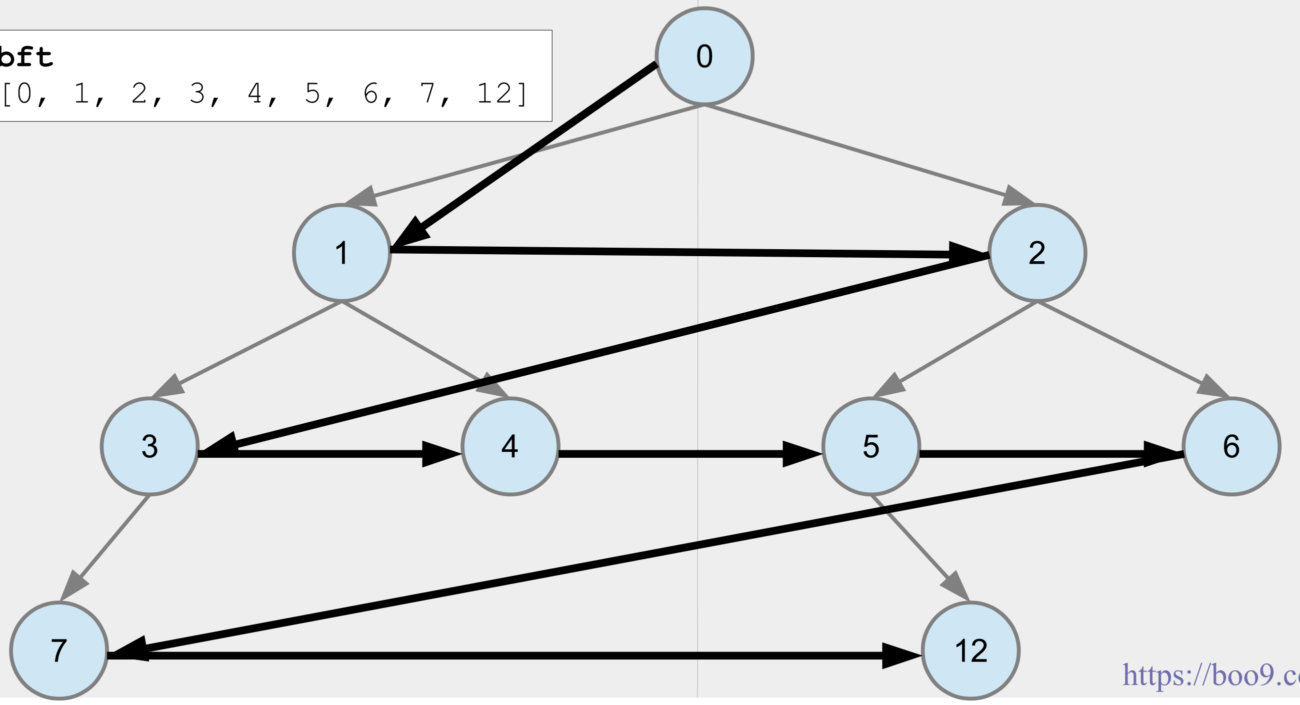
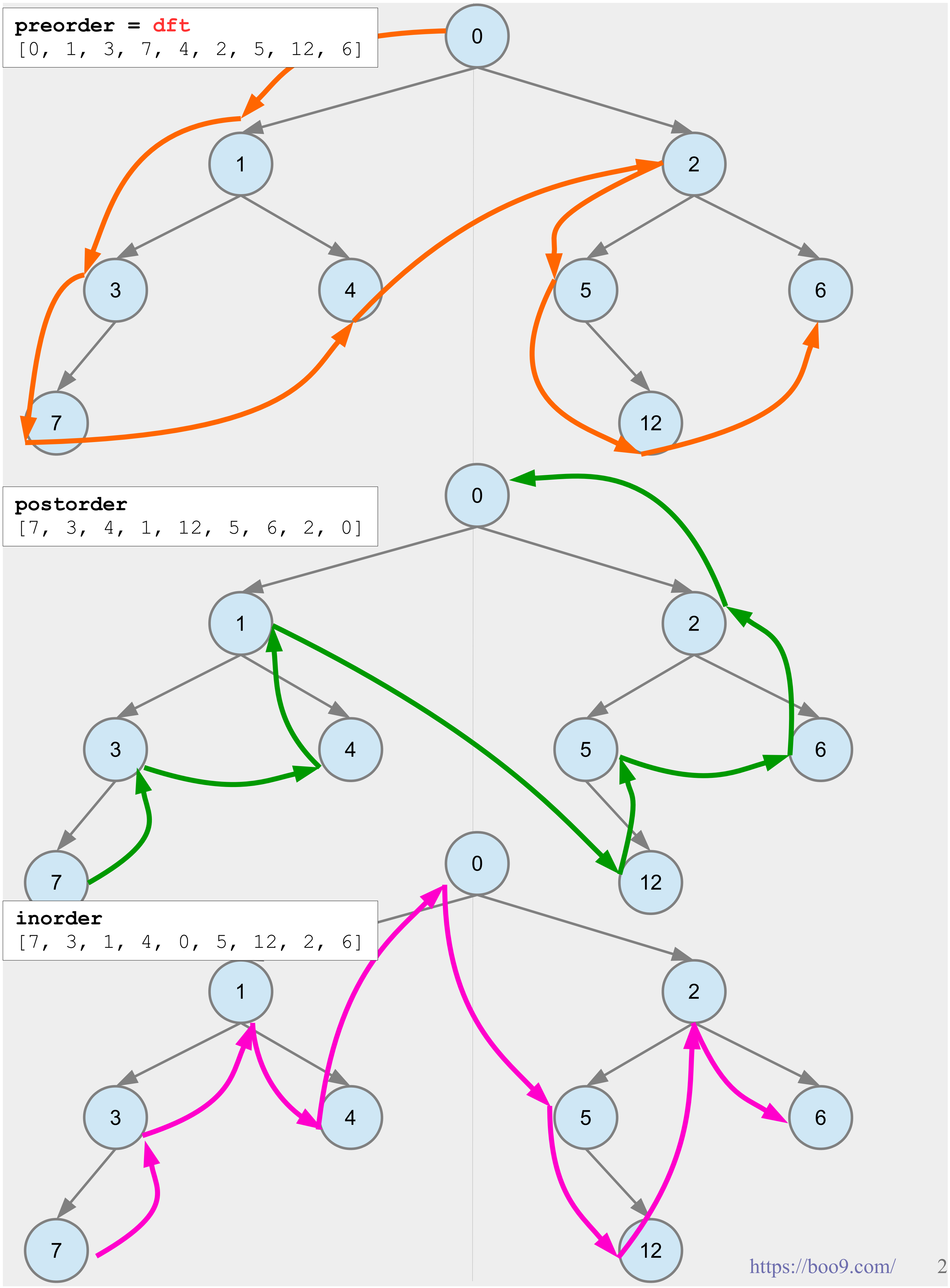
```
# Binary tree traversal
class TreeNode:
    def __init__(self, x):
        self.val = x
        self.left = None
        self.right = None


def bft(node, list_):
    q = [node]
    while len(q) > 0:
        n = q.pop(0)
        list_.append(n.val)
        if n.left: q.append(n.left)
        if n.right: q.append(n.right)

def preorder(node, list_):
    list_.append(node.val)
    if node.left: preorder(node.left, list_)
    if node.right: preorder(node.right, list_)

def postorder(node, list_):
    if node.left: postorder(node.left, list_)
    if node.right: postorder(node.right, list_)
    list_.append(node.val)

def inorder(node, list_):
    if node.left: inorder(node.left, list_)
    list_.append(node.val)
    if node.right: inorder(node.right, list_)

def traversal(root: TreeNode):
    if root == None: return []

    list_ = []
    bft(root, list_)
    return list_
```

```
# Convert a list-of-values to tree (array-based convention)
def list_to_tree(list_):
    if len(list_) == 0: return None

    # 1st pass - create the nodes
    node_list = [None] * len(list_)
    for i in range(len(list_)):
        if list_[i] != None:
            node_list[i] = TreeNode(list_[i])

    # 2nd pass - link
    for i in range(len(list_)):
        if node_list[i] != None:
            left_i = 2*i + 1
            right_i = left_i + 1
            if left_i<len(list_) and list_[left_i] != None:
                node_list[i].left = node_list[left_i]
            if right_i<len(list_) and list_[right_i] != None:
                node_list[i].right = node_list[right_i]

    return node_list[0]
# --------

# Tests ...
list_ = [0, 1, 2, 3, 4, 5, 6, 7, None, None, None, None, 12]
root = list_to_tree(list_)
print(root.right.left.right.val)
>> 12
```

bft
```
[0, 1, 2, 3, 4, 5, 6, 7, 12]
```





preorder = dft
```
[0, 1, 3, 7, 4, 2, 5, 12, 6]
```

postorder
```
[7, 3, 4, 1, 12, 5, 6, 2, 0]
```

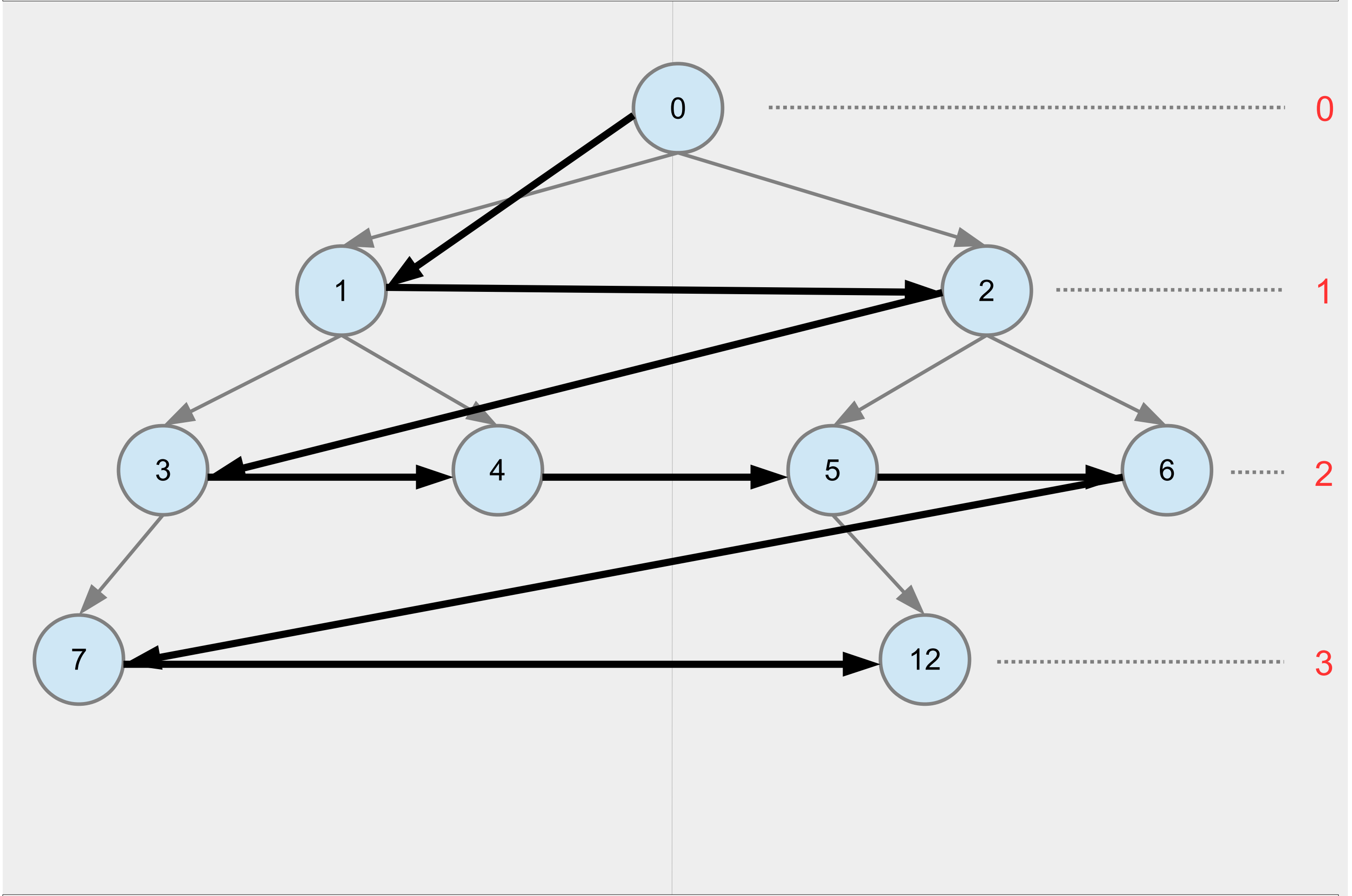
inorder
```
[7, 3, 1, 4, 0, 5, 12, 2, 6]
```

```
# Binary tree traversal w/ depth info


def bft_depth(node, depth, list_):
    q = [(node, depth)]
    while len(q) > 0:
        n, d = q.pop(0)
        list_.append((n.val, d))
        if n.left: q.append((n.left, d+1))
        if n.right: q.append((n.right, d+1))


def preorder_depth(node, depth, list_):
    list_.append((node.val, depth))
    if node.left: preorder_depth(node.left, depth+1, list_)
    if node.right: preorder_depth(node.right, depth+1, list_)
```

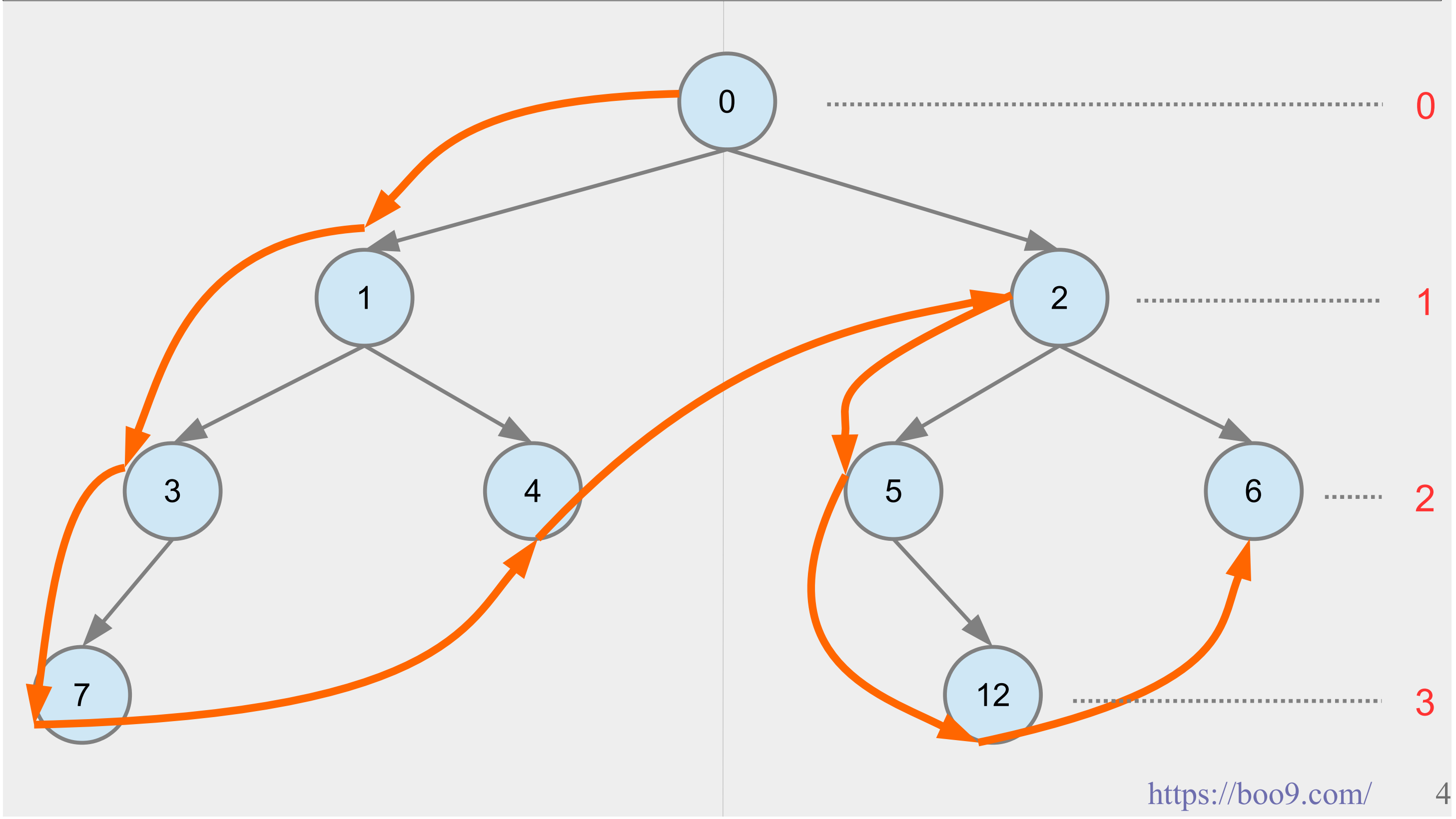
Similarly for postorder_depth and inorder_depth ...

```
def traversal_depth(root: TreeNode):
    if root == None: return []

    list_ = []
    bft_depth(root, 0, list_)
    return list_
```

```
bft_depth
[(0, 0), (1, 1), (2, 1), (3, 2), (4, 2), (5, 2), (6, 2), (7, 3), (12, 3)]
```



```
preorder_depth = dft_depth
[(0, 0), (1, 1), (3, 2), (7, 3), (4, 2), (2, 1), (5, 2), (12, 3), (6, 2)]
```


0
1
2
3
4
5
6
7
12
0
1
2
3