Find the Node With the Max Difference in the Total Number of Descendants in its left Subtree and Right Subtree

```python
# Time : O(n)
# Space : O(h)

# Method 1
global_max = -1
res = None

def node_diff(root):
  if root is None:
    return 0
  left_total = node_diff(root.left)
  right_total = node_diff(root.right)
  global global_max
  global res
  if abs(left_total - right_total) > global_max:
    global_max = abs(left_total - right_total)
    res = root
  return left_total + right_total + 1

def get_max_diff(root):
  global res
  node_diff(root)
  return res

# Method 2
class ResultWrapper:
  def __init__(self):
    self.global_max = -1
    self.solution = None

def max_diff_node(root, res):
  if not root:
    return 0
  left_total = max_diff_node(root.left, res)
  right_total = max_diff_node(root.right, res)
  if abs(left_total - right_total) > res.global_max:
    res.global_max = abs(left_total - right_total)
    res.solution = root
```

```
  return left_total + right_total + 1

def max_diff(root):
  res = ResultWrapper()
  max_diff_node(root, res)
  return res.solution
```