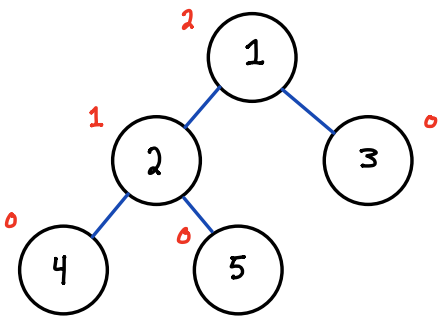
Calculating Height of a Binary Tree

Height of tree:

The height of a tree is the height of its root node.

Height of Node:

The height of a node is the number of edges on the longest path between that node and a leaf.



Height of Nodes

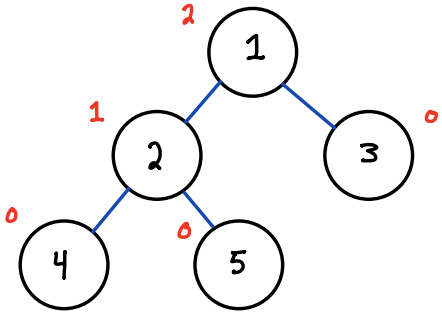
- Recursion to Solve this Problem

```
def height(node):
    if node is None:
        return -1
    left_height = height(node.left)
    right_height = height(node.right)

    return 1 + max(left_height, right_height)
```

Base Case

Recursively Call height function on right and left Sub-trees

Node:	Left_height	right_height



Step-by Step

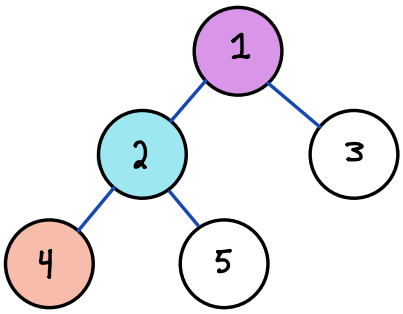
```
def height(node):
    if node is None:
        return -1
    left_height = height(node.left)
    right_height = height(node.right)

    return 1 + max(left_height, right_height)
```

Node isn't None → SKIP Line

Call first recursive function

Start @ root Node

Node:	Left_height	right_height
1		
2		
4		

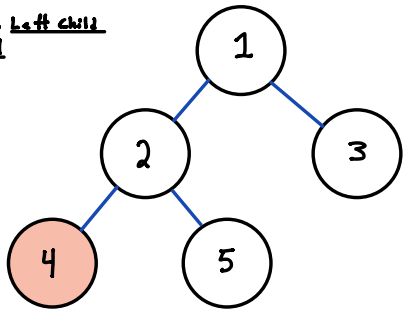


Make recursive call to Left Child of Node 4

Call height Function

```
def height(node):
    if node is None:
        return -1
    left_height = height(node.left)
    right_height = height(node.right)

    return 1 + max(left_height, right_height)
```

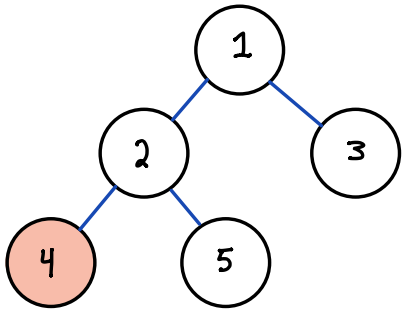
None since no Left Child at Node 4



Node:	Left_height	right_height
1		
2		
4		
None	-1	

Make recursive call to right child of Node 4

```
def height(node):
    if node is None:
        return -1
    left_height = height(node.left)
    right_height = height(node.right)

    return 1 + max(left_height, right_height)
```

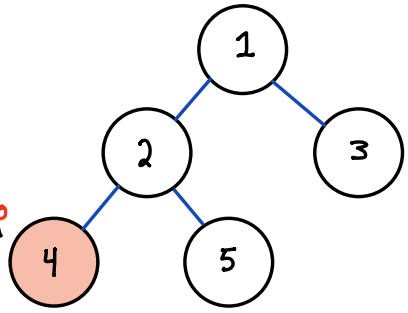


Node:	Left_height	right_height
1		
2		
4		
None	-1	-1

← finished two recursive calls for node 4

Since we finished two recursive calls for node 4

```
def height(node):
    if node is None:
        return -1
    left_height = height(node.left)
    right_height = height(node.right)

    return 1 + max(left_height, right_height)
```



Node:	Left_height	right_height
1		
2		
4		
None	-1	-1

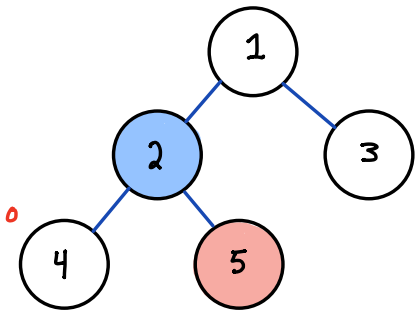
return 1 + max(left_height, right_height)
return 1 + max(-1, -1)
return 1 + -1

return 0

Recursive Call at node 2

```
def height(node):
    if node is None:
        return -1
    left_height = height(node.left)
    right_height = height(node.right)

    return 1 + max(left_height, right_height)
```

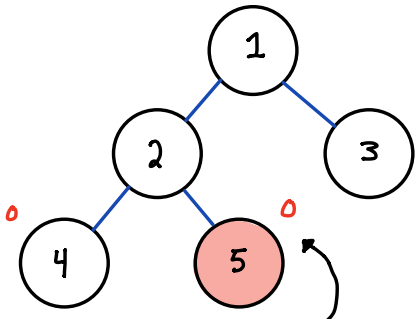
Node:	Left_height	right_height
1		
2		



Recursive Call at Node 5

```
def height(node):
    if node is None:
        return -1
    left_height = height(node.left)
    right_height = height(node.right)

    return 1 + max(left_height, right_height)
```

Node:	Left_height	right_height
1		
2		
5		
None	-1	-1



Both left and right child of 5 will return -1

- Then finish these two recursive calls

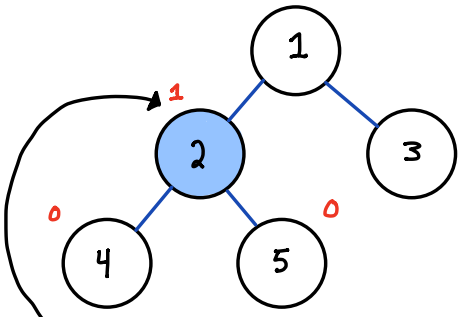
return 1 + max(left_height, right_height)
return 1 + max(-1, -1)
return 1 + -1

return 0

Go Back to Node 2

```
def height(node):
    if node is None:
        return -1
    left_height = height(node.left)
    right_height = height(node.right)

    return 1 + max(left_height, right_height)
```

Node:	Left_height	right_height
1		
2		



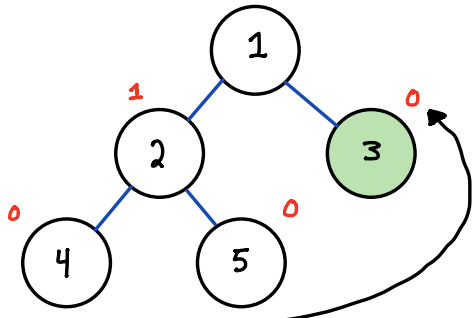
- We finished both Left and Right recursive calls for node 2, so we proceed to the return statement:

```
return 1 + max(left_height, right_height)
return 1 + max(0,0)
return 1 + 0
return 1
```

Now we go back to node 1, since we performed recursive call on the Left Height subtree, we will go to the right subtree of node 3

```
def height(node):
    if node is None:
        return -1
    left_height = height(node.left)
    right_height = height(node.right)

    return 1 + max(left_height, right_height)
```

Node:	Left_height	right_height
1	1	
3		
None	-1	-1



- Then finish these two recursive calls

return 1 + max(left_height, right_height)
return 1 + max(-1, -1)
return 1 + -1

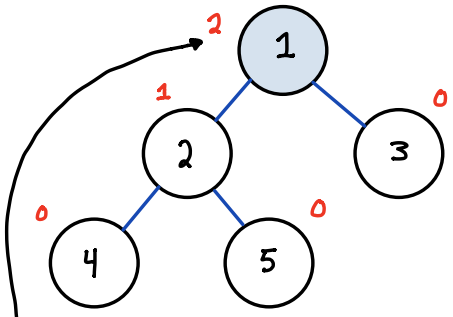
return 0

Now we go Back to node 1, we've completed both Left and Right recursive calls.

```
def height(node):
    if node is None:
        return -1
    left_height = height(node.left)
    right_height = height(node.right)

    return 1 + max(left_height, right_height)
```

Node:	Left_height	right_height
1	1	0

- Now we return 1 + max(r, l)



return 1 + max(left_height, right_height)
return 1 + max(1, 0)
return 1 + 1

return 2

Python Code:

```python
def height(self, node):
    if node is None:
        return -1
    left_height = self.height(node.left)
    right_height = self.height(node.right)

    return 1 + max(left_height, right_height)
```