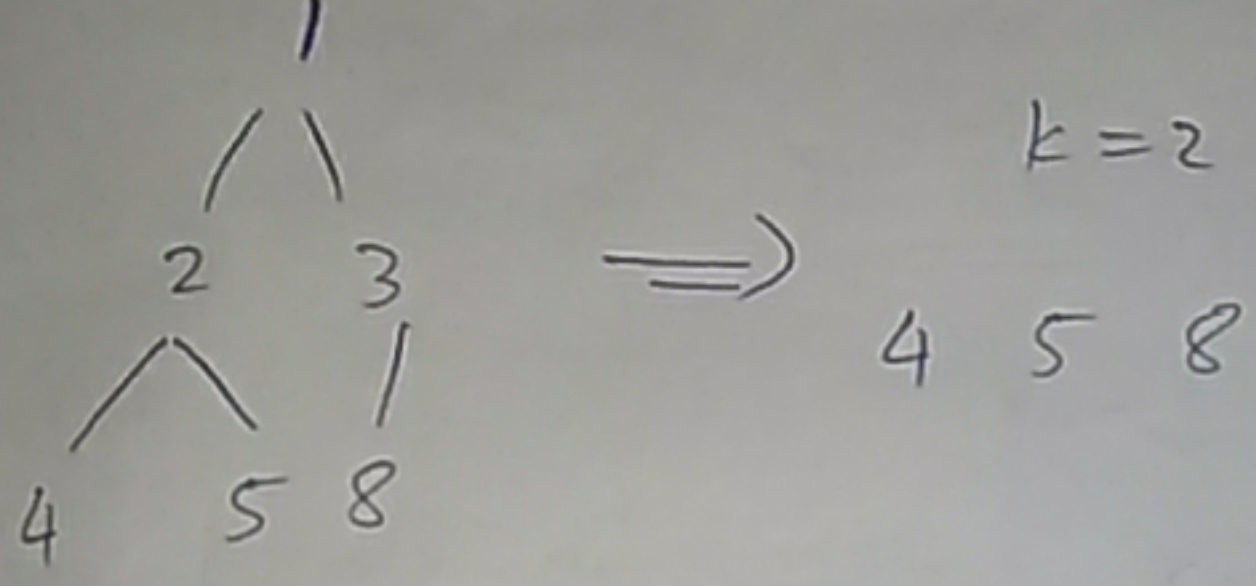
day 20

Print nodes at k distance from root



Implemenations:

Class Node:

```
def __init__(self, data):
    self.data = data
    self.left = None
    self.right = None

def printkdistance(root, k):
    if root is None:
        return
    if k==0:
        print(root.data)
```

else:

Print kdistance (root.left, k-1)

Print kdistance (root.right, k-1)

Input:

root = Node(1)

root.left = Node(2)

root.right = Node(3)

root.left.left = Node(4)

root.left.right = Node(5)

root.right.left = Node(8)

Print kdistance (root, 2)

Output:

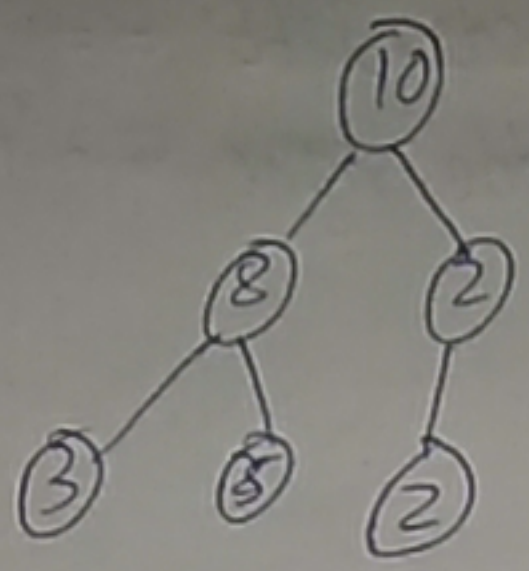
4 5 8

Time Complexity:

O(n)

Children Sum property in Binary tree

For everynode, data value must be equal to sum of data values in left and right children. Consider data value as 0 for Null children.



Traverse the given binary tree. for each node check (reawsively) if the node and both its children satisty the children sum properity. If so then ret else return false.

Implementation:

```
class Node:
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None

def sumproperty(node):
    left_data = 0
    right_data = 0
```

```
If (node == None or (node.left == None and
                     node.right == None)):

    return 1

else:
    if (node.left != None)
        left_data = Node.left.data

    if (right.node != None)
        right_data = node.right.data

    if ((node.data == left_data + right_data) and
                Sumproperty (node.left) and
                Sum property (node.right)):

        return 1

    else:

        return 0
```

Input:

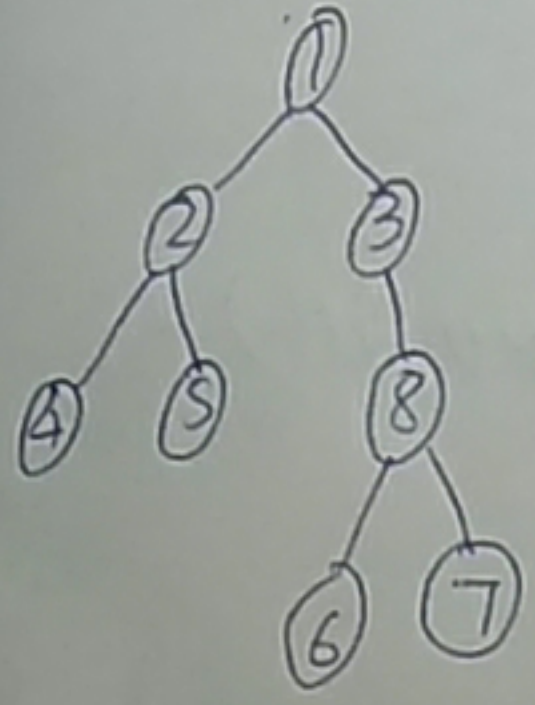
```
root = Node(10)

root.left = Node(8)

root.right = Node(2)

root.left.left = Node(3)

root.left.right = Node(5)

root.right.right = Node(2)
```

```
If (sumProperty (root)):
    print ("the given tree statisfies the "children
                sum property")
else:
    print ("the given tree does not satisfies the
                children sum property
```

output :

the given tree statisfies the Children Sum property

Maximum width of a binary tree



$\Rightarrow$ width of level 1 is 1
width of level 2 is 2
width of level 3 is 3
width of level 4 is 2

So the maximum width of tree is 3

```
class Node:
    def __init__(self,data)
        self.data = data
        self.right = None
        self.left = None

def getmaxwidth(root):
    if root is None:
        return 0
    q = []
    maxwidth = 0
    q.insert(0, root)
    while (q != []):
        count = len(q)
        maxwidth = max(count, maxwidth)
        while (count is not 0):
            count = count -1
            temp = q[-1]
            q.pop()
            if temp.left is not None:
                q.insert(0, temp.left)
            if temp.right is not None:
                q.insert(0, temp.right)
```

return maxwidth

input:

```
root = Node(1)
root.left = Node(2)
root.right = Node(3)
root.left.left = Node(4)
root.left.right = Node(5)
root.right.right = Node(6)
print(getmaxwidth(root))
```

output:

3