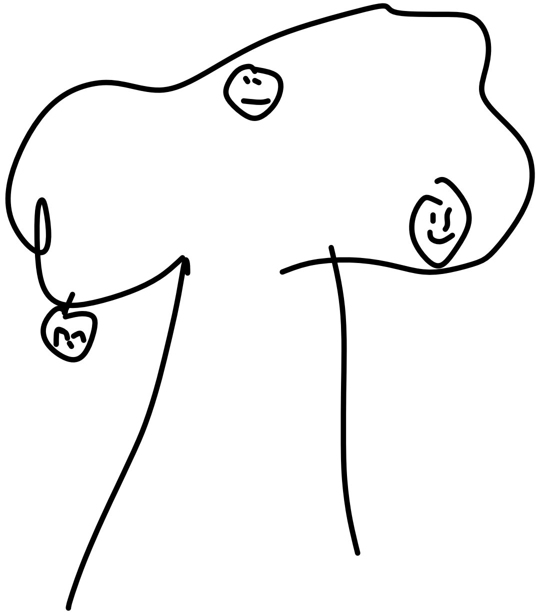
Algorithms and Data Structures for Data Science

lab_trees

CS 277
Brad Solomon

March 1, 2024



UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science

Learning Objectives

Explore structure and use of a binary tree

Practice building programs by applying small functions to solve more complex problems

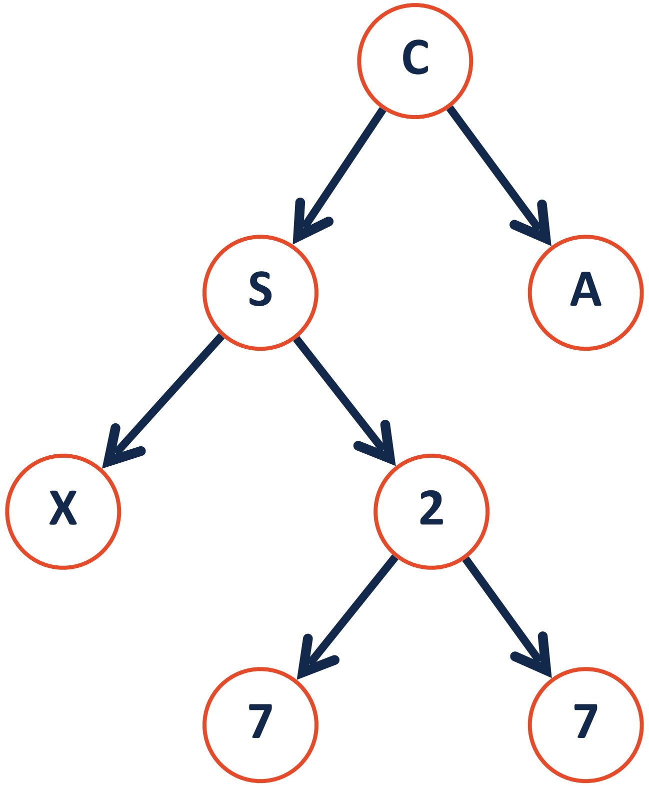
(Binary) Tree Recursion

A **binary tree** is a tree T such that:

$$T = None$$

or

$$T = treeNode(val, T_L, T_R)$$

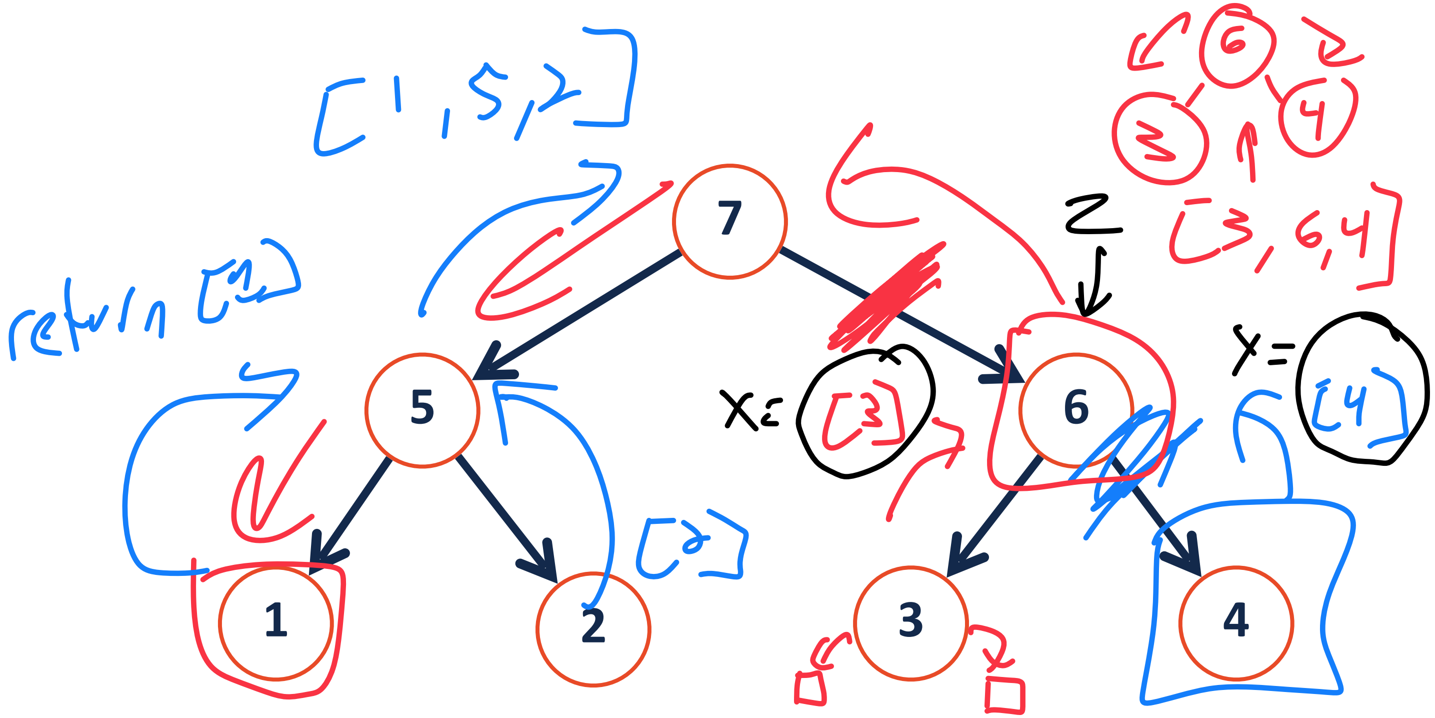


```
class treeNode:
    def __init__(self, val, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
```

```
class binaryTree:
    def __init__(self):
        self.root = None
```

In-Order Traversal

1) Recurse left
2) Get current nodes value
3) Recurse right



Goal: Return a list

def (Node):   "None

↳ Node.left = None.left

will crash!

Base Case   if leaf → return [val]

if Node is None → return []

How do I know I am a leaf?

O(1)

Recursion

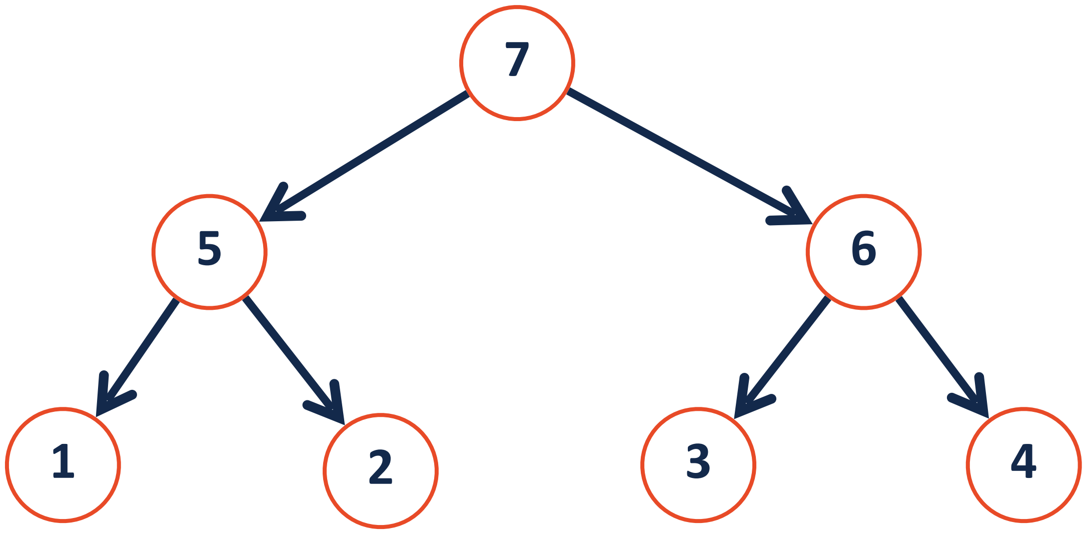
Combining   Given return from left + right return the sub problem soln

Post-Order Traversal

1) Recurse left

2) Recurse right

3) Get current nodes value

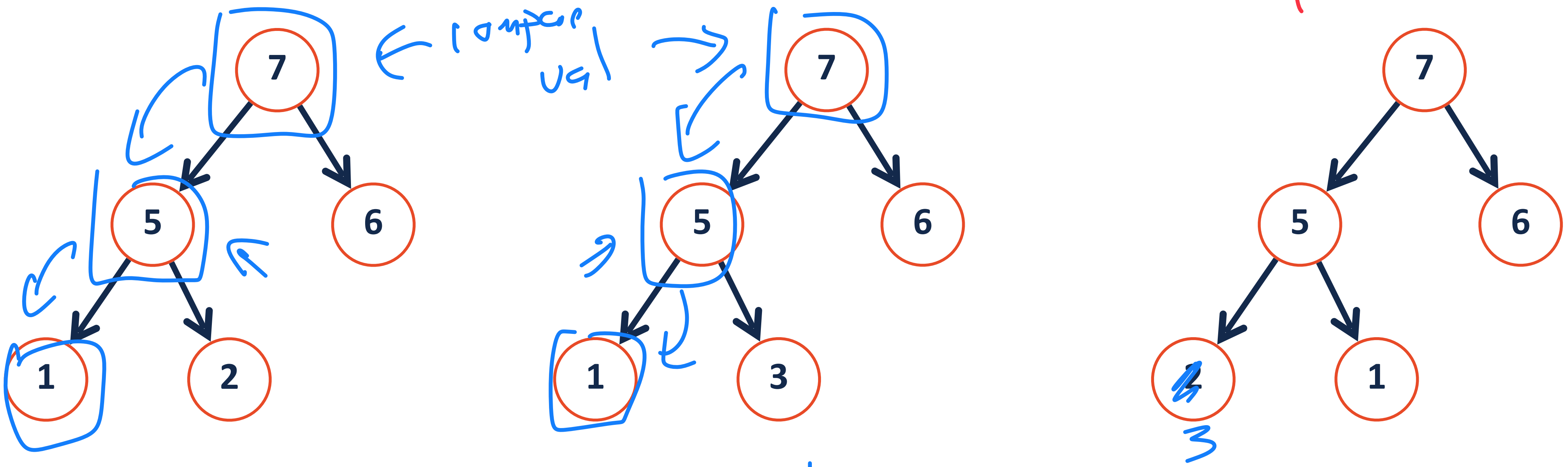


Are Equal

How can we tell if two trees are equal?

[1, 5, 3, 7, 6]

[~~2~~ 3, 5, 1, 7, 6]



1) Check node1.val == node2.val

2) Check if left child exists -> recurse left

3) Repeat right

Are Equal

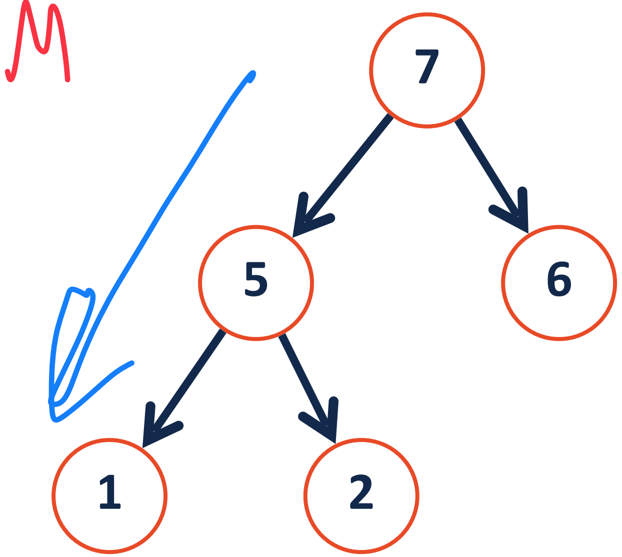
One traversal doesn't work.

LMR

A

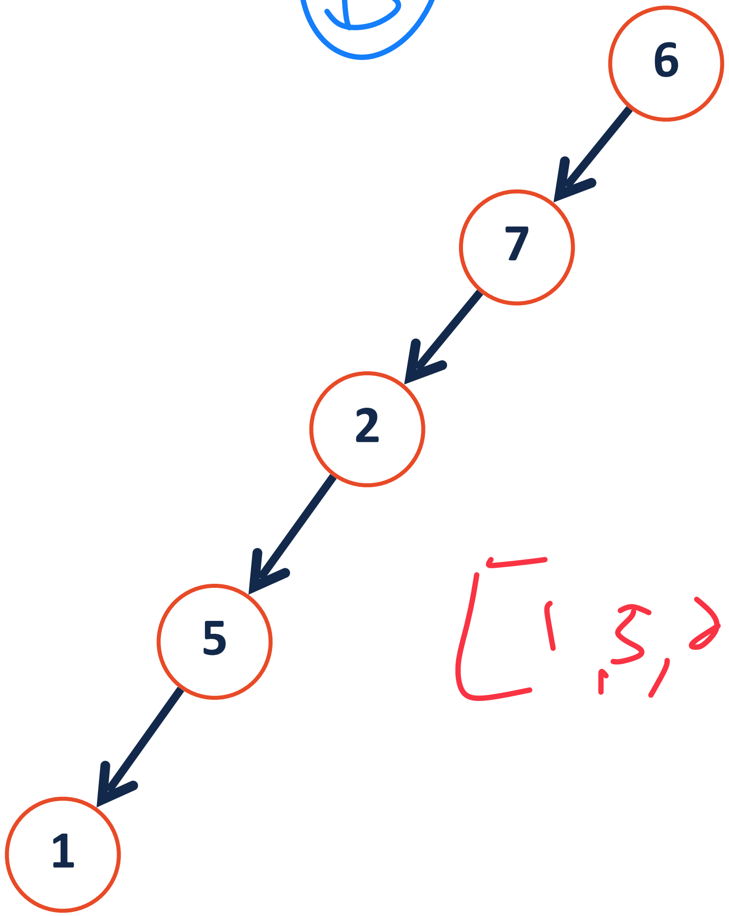
[1 2 5 6 7]

LRM



[1, 5, 2, 7, 6]

B



[1, 5, 2, 7, 6]

[1, 5, 2, 7, 6]

pathToNode()

`pathToNode(<t>, 2)`

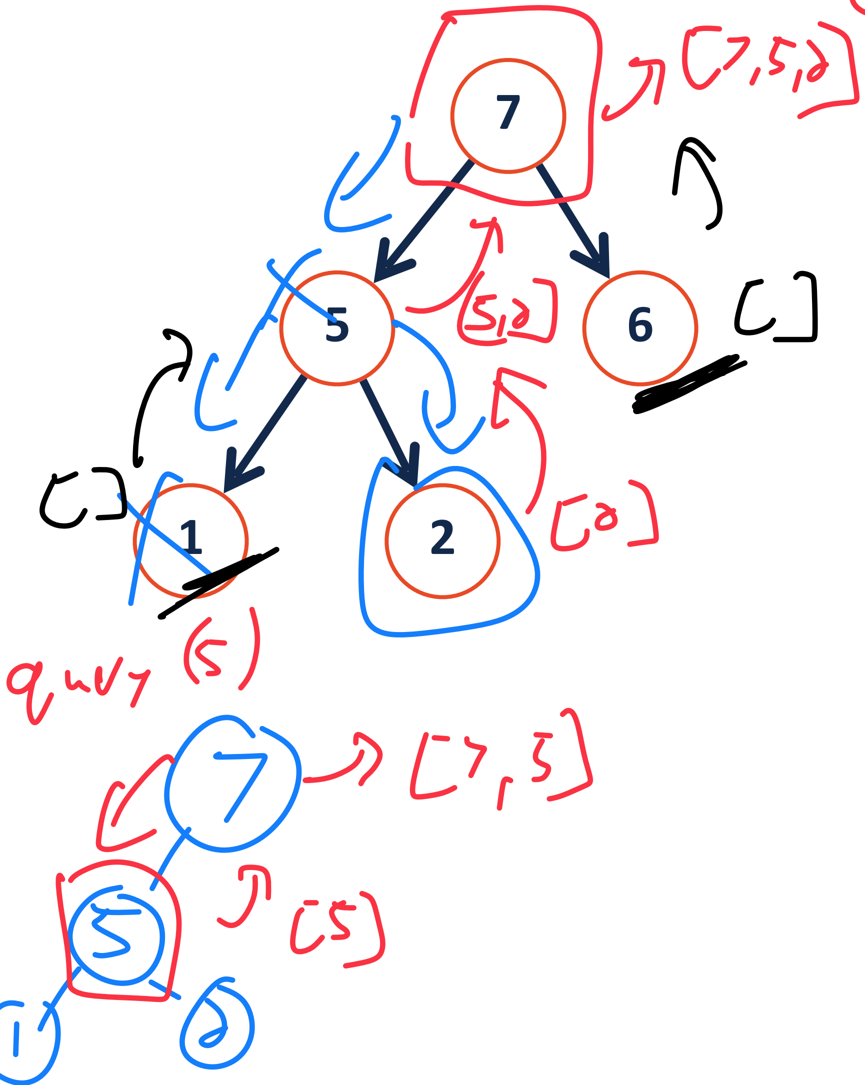
val ↑

How can we return the path to a specific node?

↳ In class how to search a tree

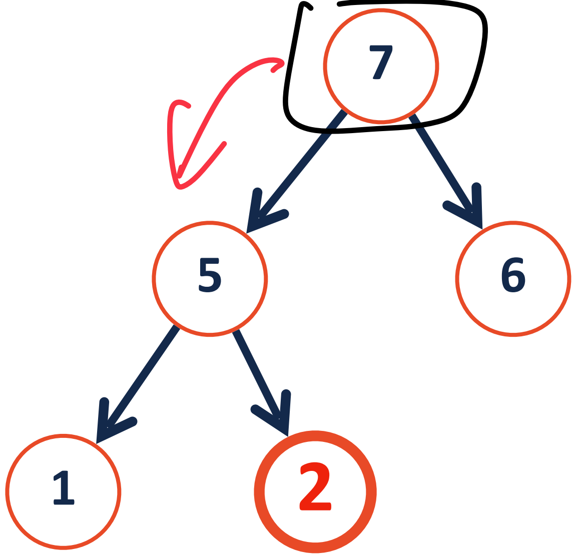
↳ A recursive traversal

Base Case: If node value is query or If leaf



[7,5,2]

pathToNode Visualized



def pathToNode(<7>~~t:~~, 2)

<7> is not 2

[7, 5, 2]

def pathToNode(<5>~~t:~~ 2)

<5> is not 2

[5, 2]

def pathToNode(<6>~~t:~~, 2)

return []

return [2]

def pathToNode(<1>~~t:~~, 2)

Is a leaf that is not 2!

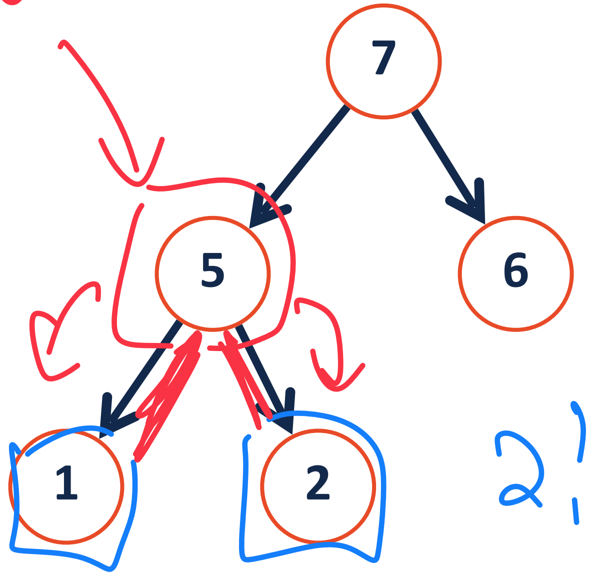
def pathToNode(<2>~~t:~~, 2)

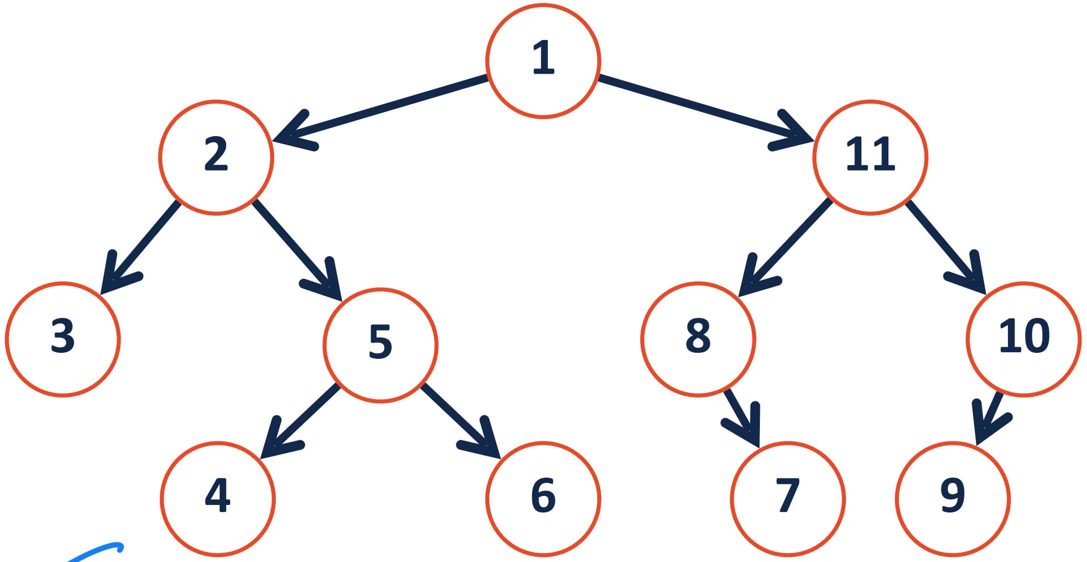
I found node!

distBetweenNodes()

I want to compute the smallest distance between nodes.

lowest common anlestor





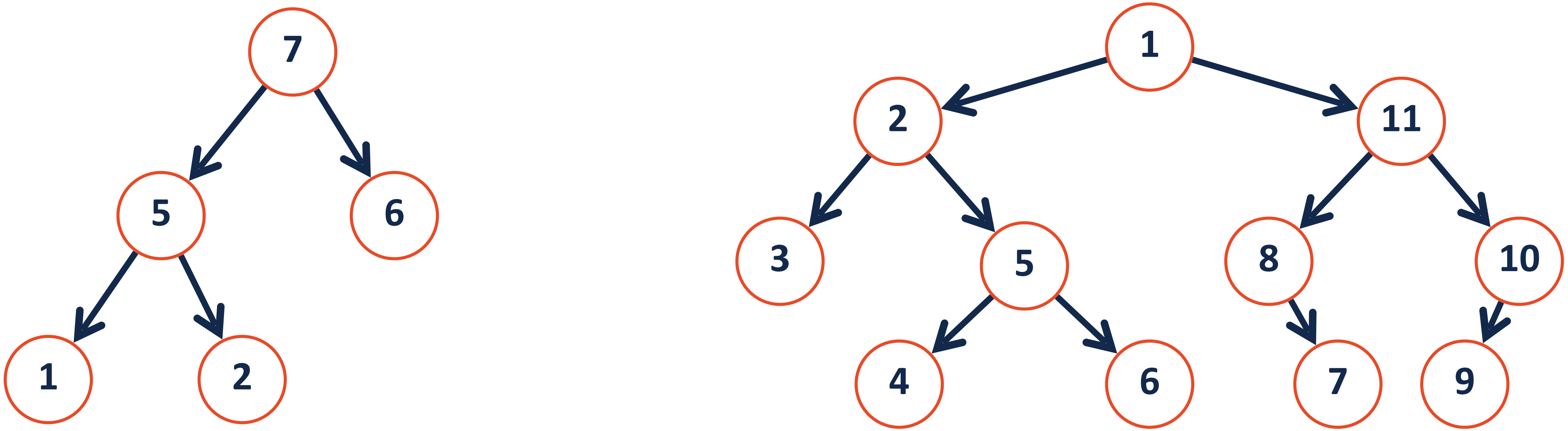
!!

path(1) - [7, 5, 1]

path(2) - [7, 5, 2]

Lowest Common Ancestor

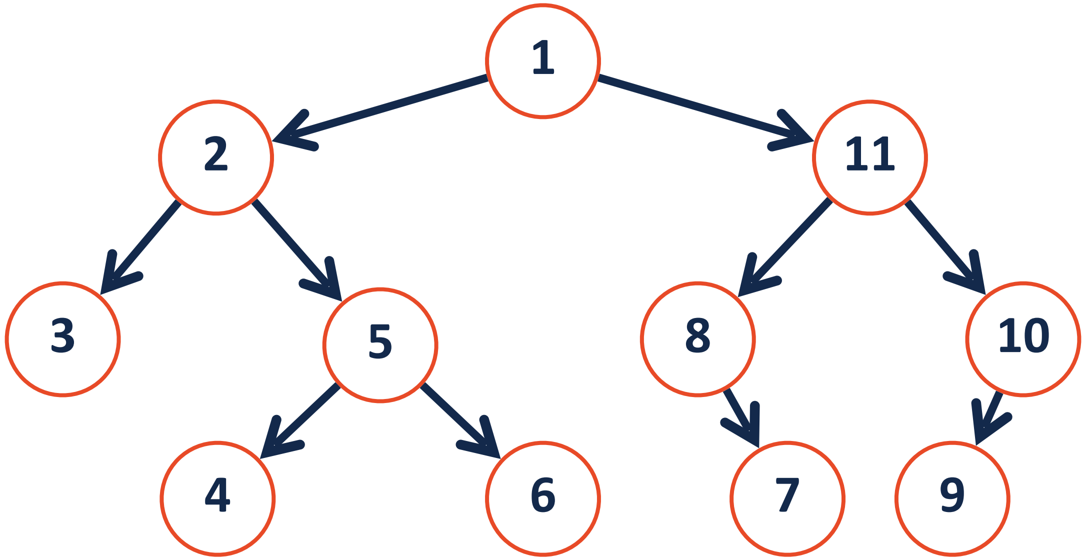
The **LCA** is the mutual parent of both nodes with the greatest depth.



distBetweenNodes()

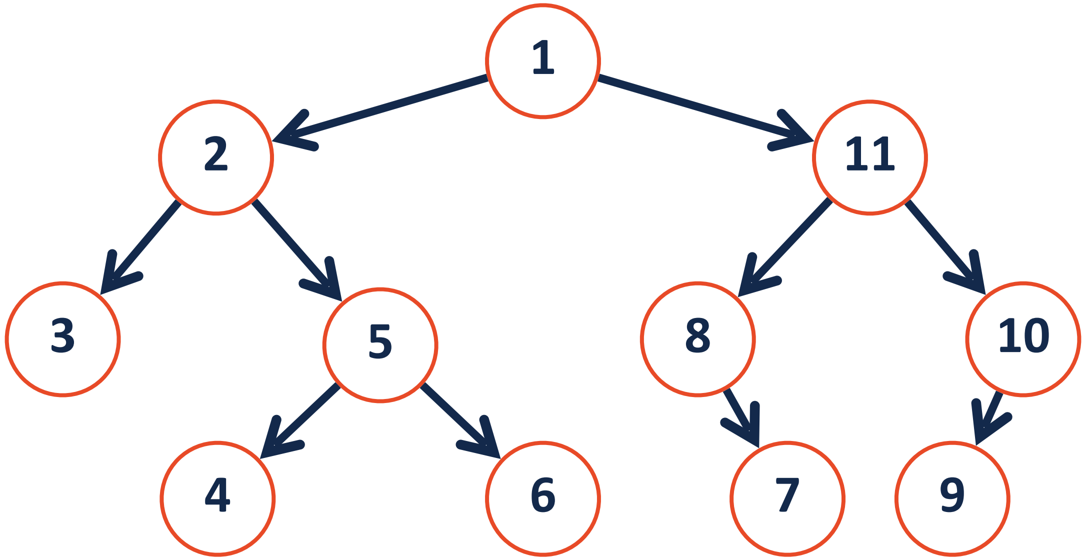
`dbn(<t>, 5, 7)`

I want to compute the smallest distance between nodes.



distBetweenNodes() `dbn(<t>, 3, 6)`

I want to compute the smallest distance between nodes.


1
2
11
3
5
8
10
4
6
7
9