Algorithms and Data Structures for Data Science

Trees 3

CS 277
Brad Solomon

October 18, 2021

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN
Department of Computer Science

Informal Early Feedback

The instructor is well-prepared for each class / recording

I feel that I can actively participate in lecture.

I feel that I can actively participate in class in general.

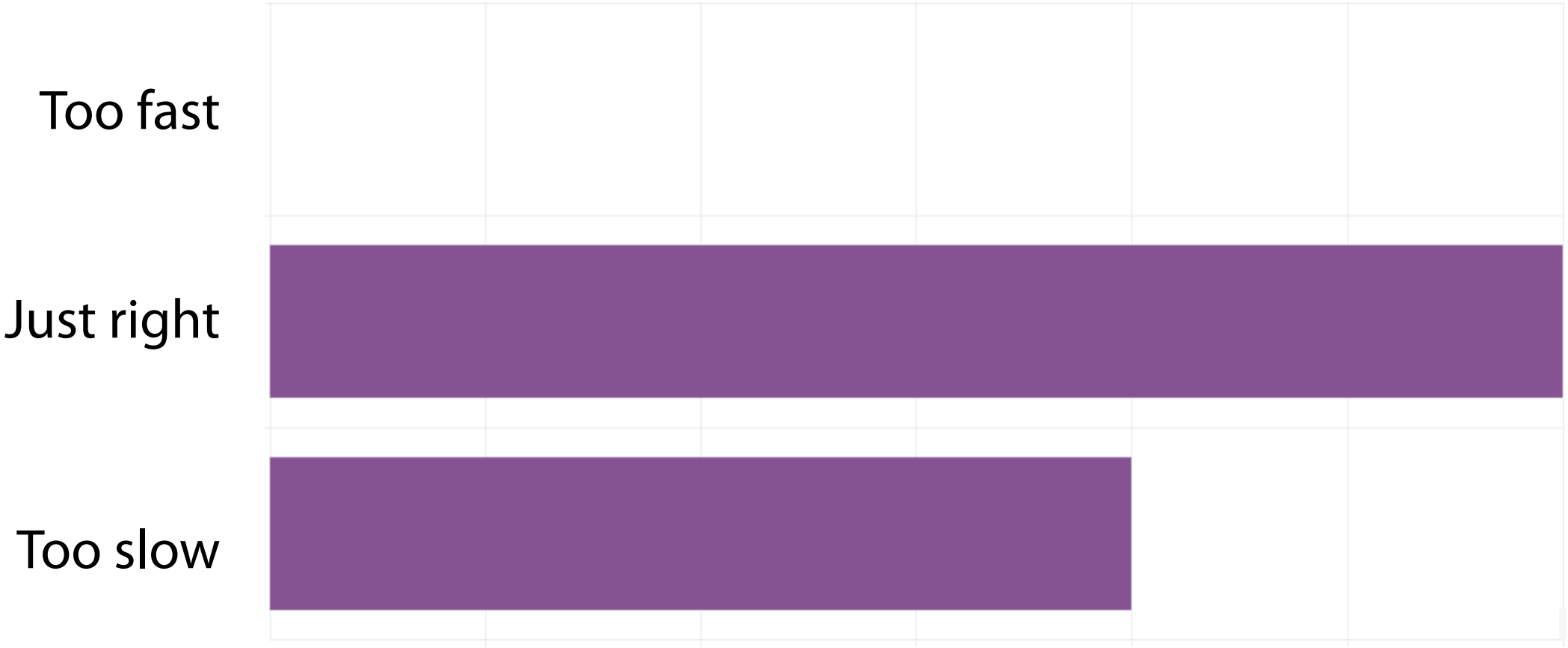
I receive helpful and complete answers to my questions.

Informal Early Feedback

How helpful have the lectures been for learning material?

How helpful have the assignments been for learning material?

Informal Early Feedback



What can make CS 277 better?

"requirements and prerequisites for the class can be defined better for a more united audience"

"Share some links at the end of the lecture so that we can explore topics in more detail"

Final Project Proposal Due Oct 20th

Proposal Deadline: October 20th

Submission: Commit proposal to GitHub.

`final_project/development/proposal._____`

Reminder: Next mp not released until October 20th!

Exam 2 October 29th - October 31st

Exam will be combination multiple choice and coding

Covers all content up until October 20th

Sign up for registration as early as October 14th

Coding question will use new workspace setup.

Learning Objectives

Review binary search trees (and dictionaries)

Conceptualize BST insertion

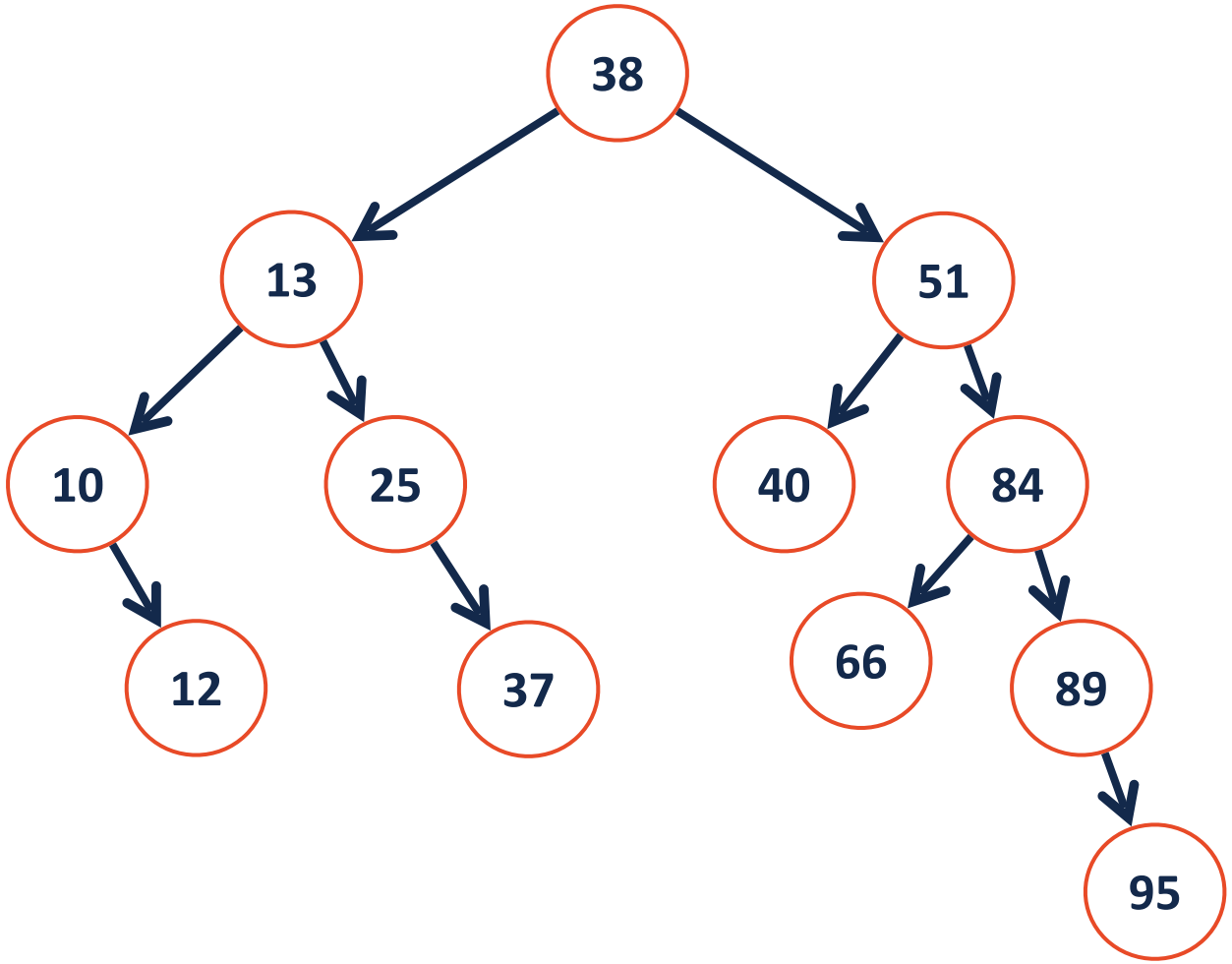
Implement BST remove

Binary Search Tree (BST)

A **BST** is a binary tree $T = treeNode(val, T_L, T_r)$ such that:

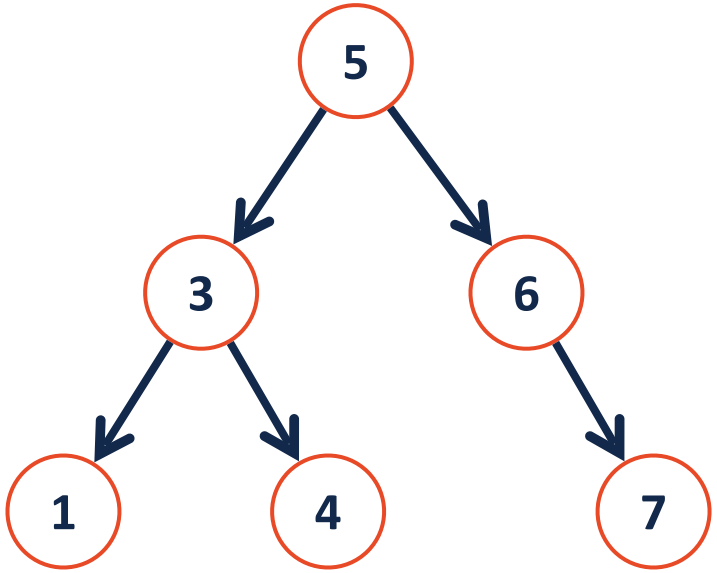
$$\forall n \in T_L,\ n.val < T.val$$

$$\forall n \in T_R,\ n.val > T.val$$



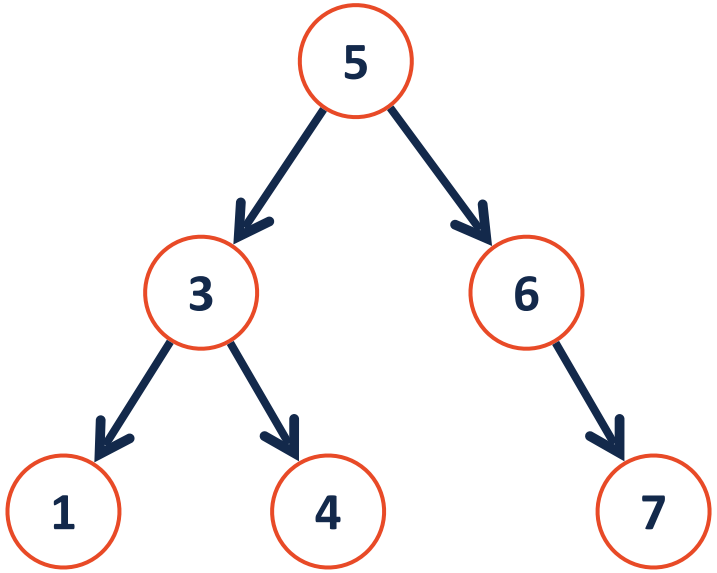
Binary Search Tree

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

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



BST Find

```
def find(self, key):
    n = self.find_helper(self.root, key)
    if n:
        return n.val
    else:
        return None

def find_helper(self,node,key):
    nkey = node.key
    if nkey > key:
        if node.left:
            return self.find_helper(node.left, key)
        else:
            return None
    elif nkey < key:
        if node.right:
            return self.find_helper(node.right, key)
        else:
            return None
    else:
        return node


```



Dictionary ADT

Data is often organized into key/value pairs:

Word → Definition

Course Number → Lecture/Lab Schedule

Node → Edges

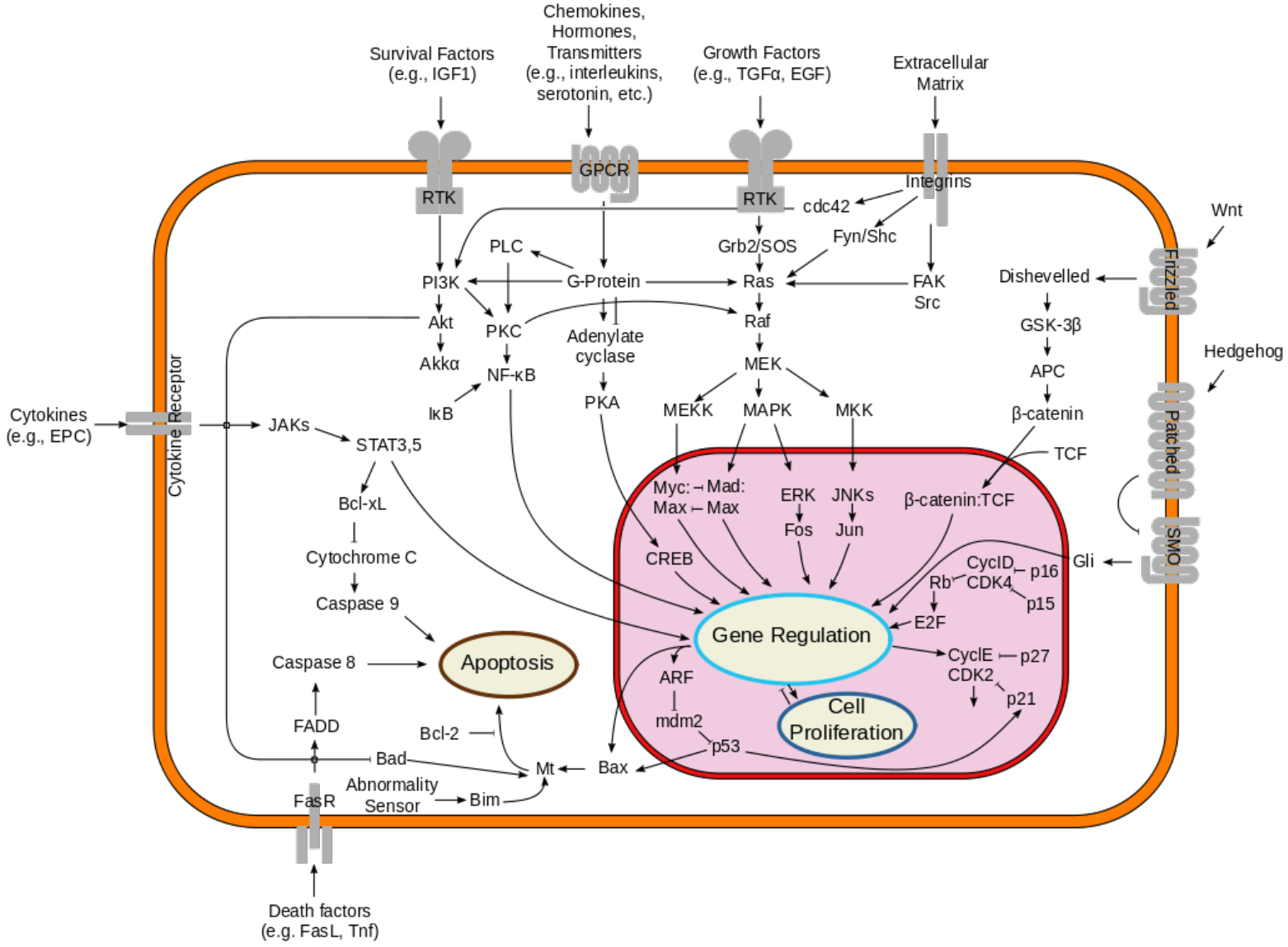
Flight Number → Arrival Information

URL → HTML Page

lab_bst

geneID ➔ Expression

geneID	length	Expression
367	529	4.5
163	785	4.8
18	428	6.6
737	799	1.6
502	278	4.9
941	484	47.7
219	456	3.0
491	473	4.6
953	704	30.6
739	819	30.2
361	298	82.2
524	736	6.8
702	570	66.9
466	414	9.9
820	383	53.8
546	588	33.0
643	961	3.8



Dictionaries in Python

```
pyDictionary = {}

pyDictionary[5]="A"
pyDictionary[3]="B"
pyDictionary[6]="C"
pyDictionary[1]="D"
pyDictionary[4]="E"

if 6 in pyDictionary:
    print(pyDictionary[6])
if 8 in pyDictionary:
    print(pyDictionary[8])
```

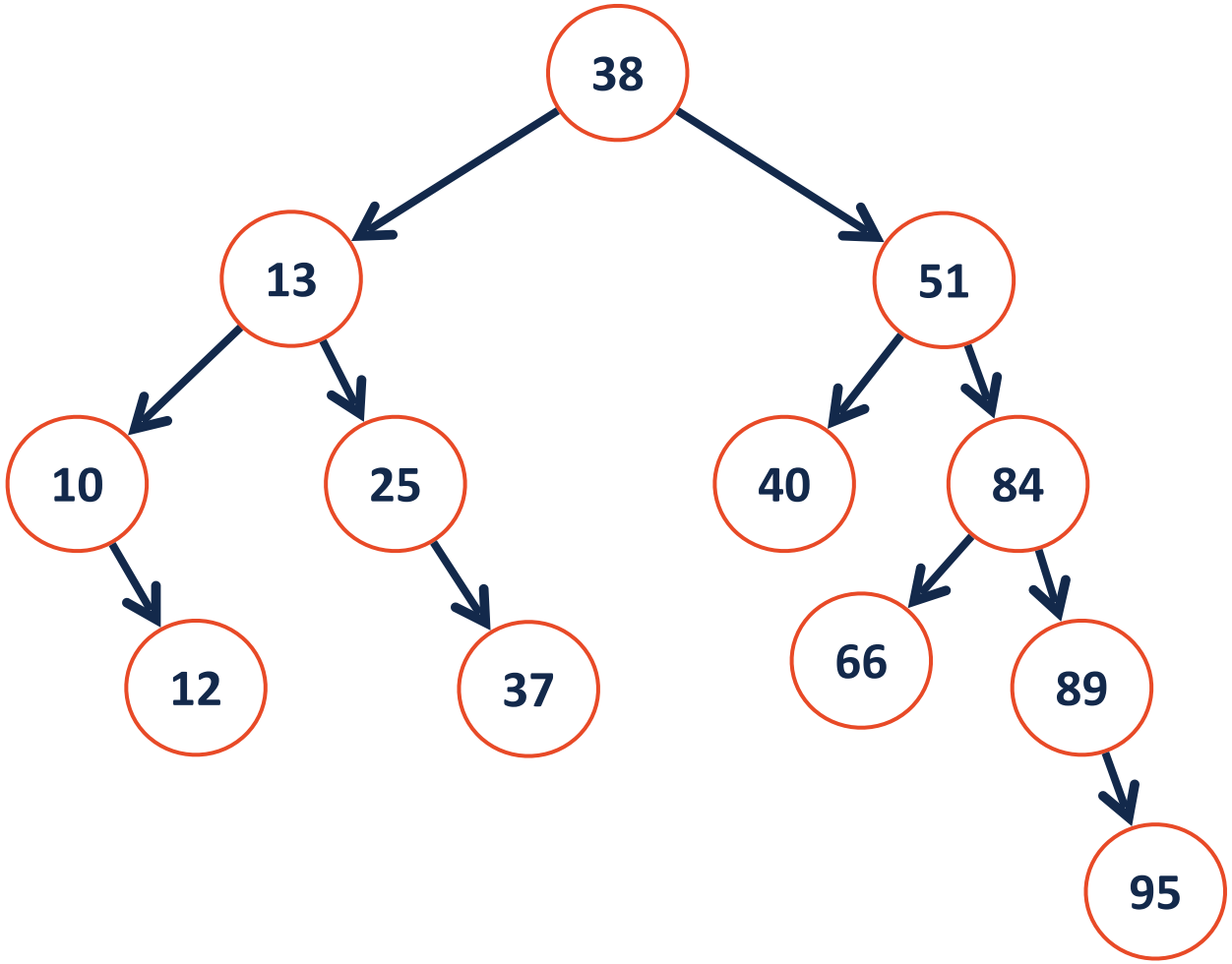
Key	Value

Dictionaries in Python

```
bt = bst()

bst.insert(5, "A")
bst.insert(3, "B")
bst.insert(6, "C")
bst.insert(1, "D")
bst.insert(4, "E")


bst.find(6)

bst.find(8)
```

BST Insert

`insert(33)`

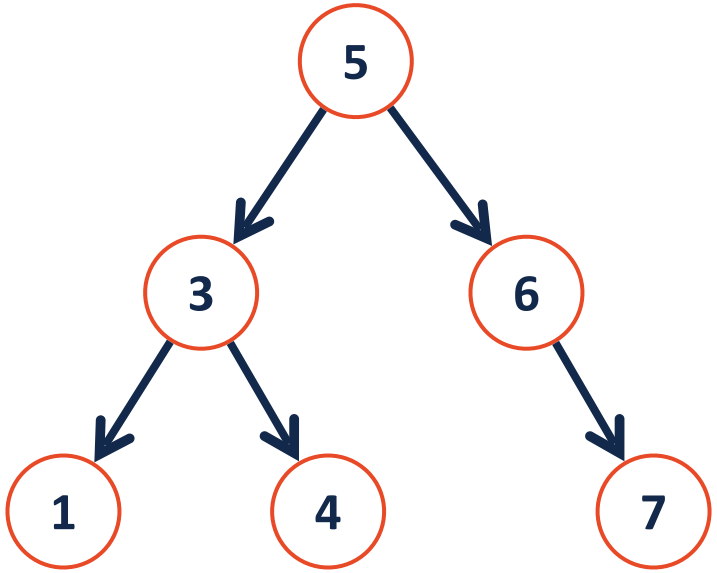


BST Insert

Base Case

Recursive Step

Combining

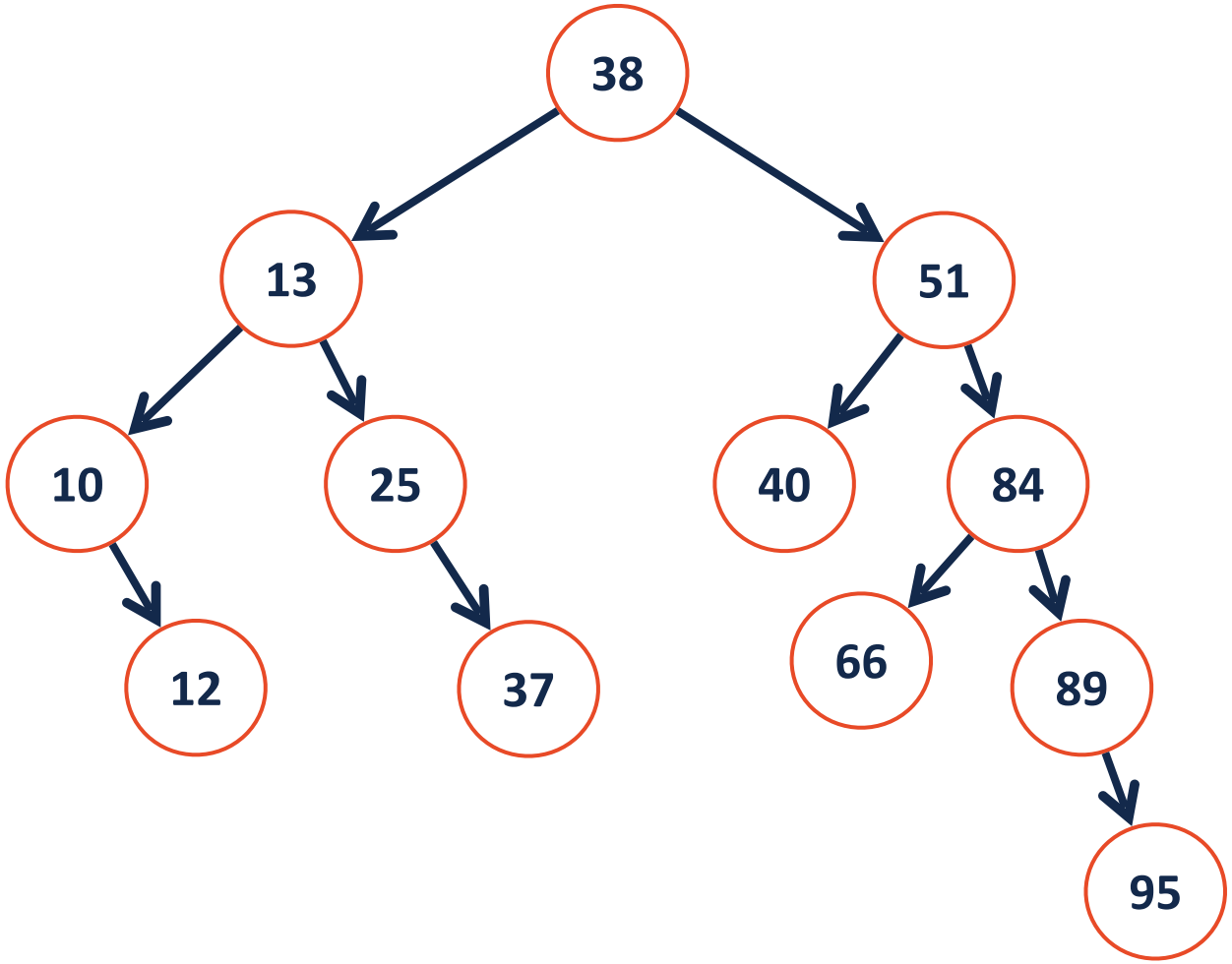


BST Insert

What binary would be formed by inserting the following sequence of integers: `[3, 7, 2, 1, 4, 8, 0]`

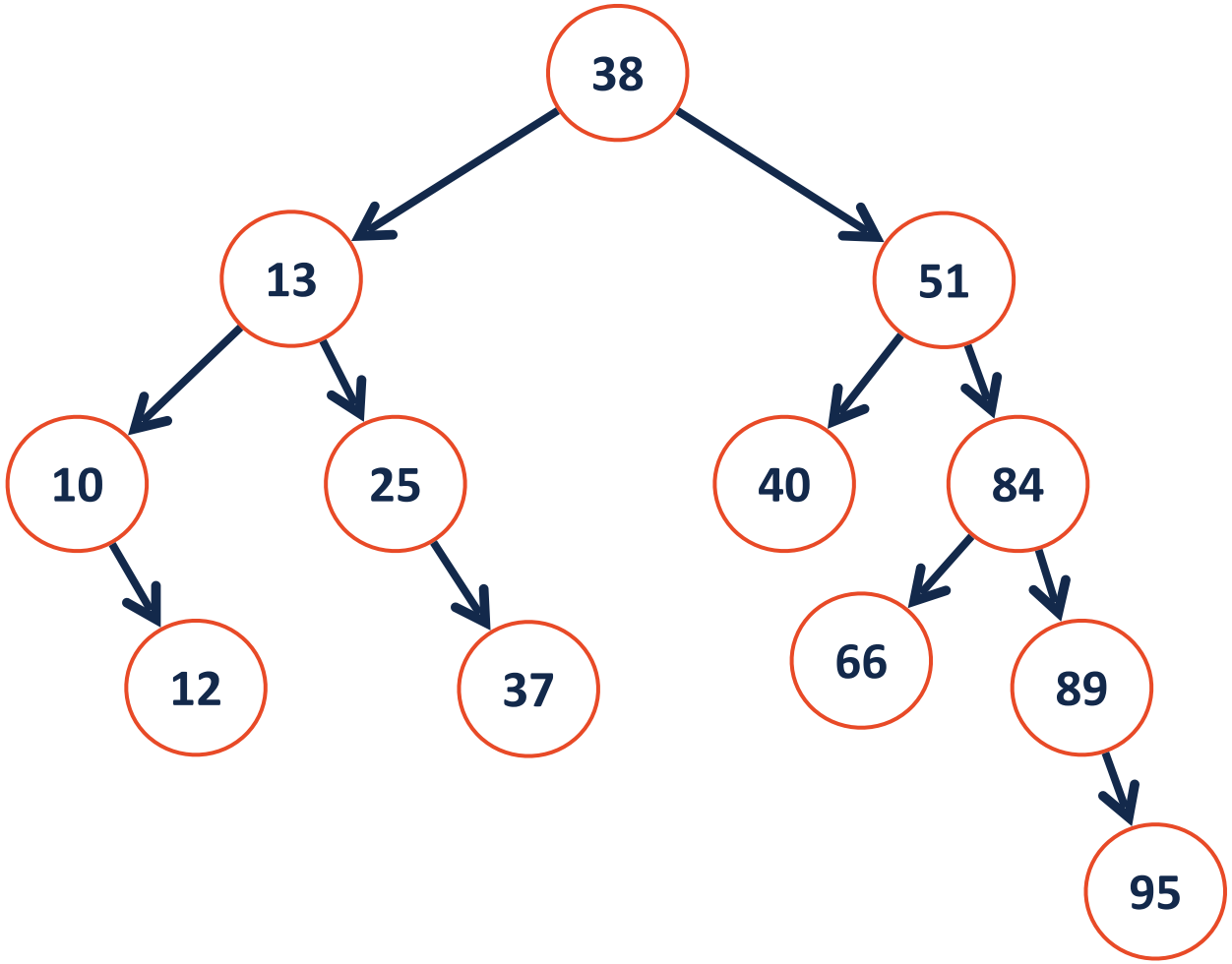
BST Remove

`remove(40)`



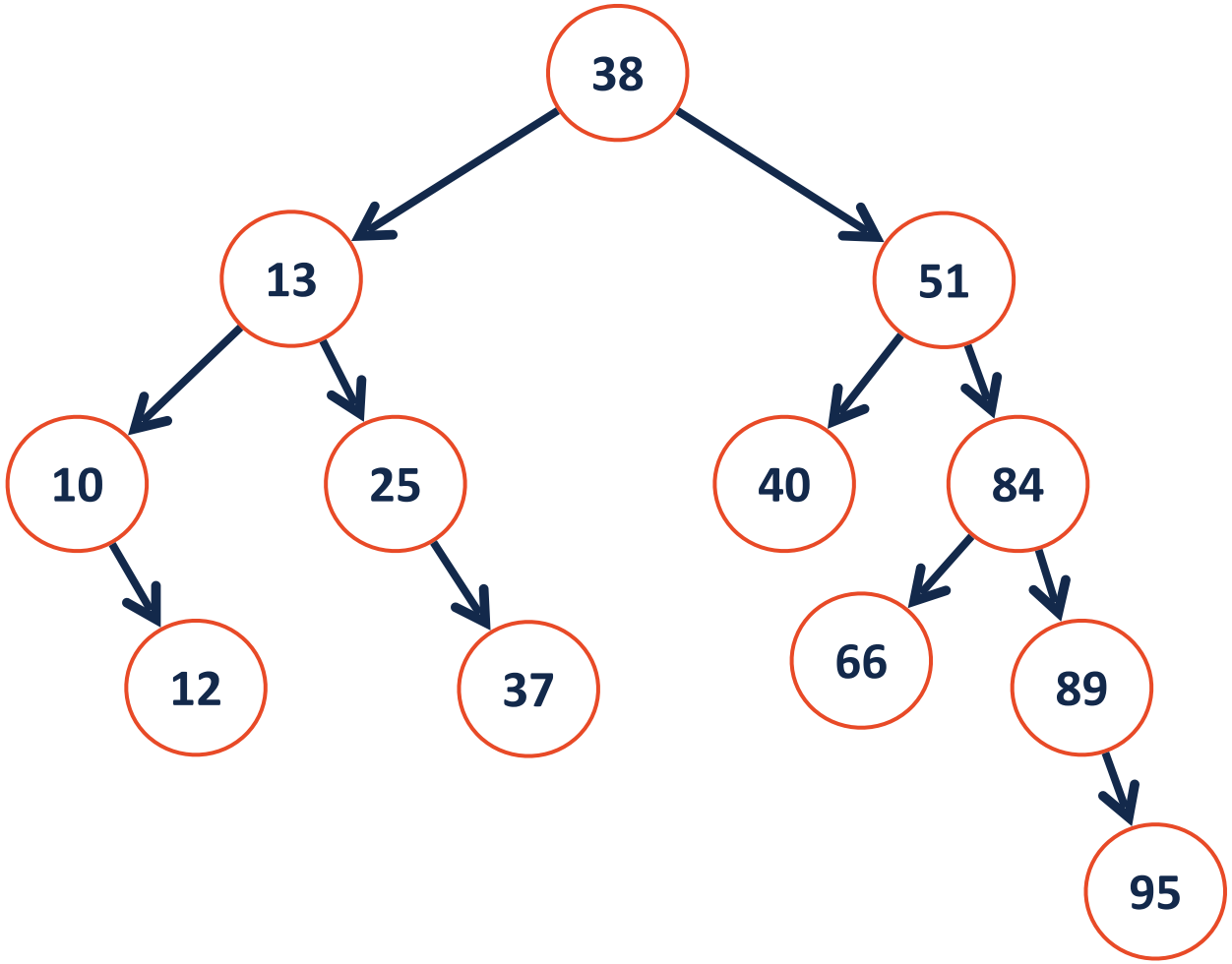
BST Remove

`remove(25)`



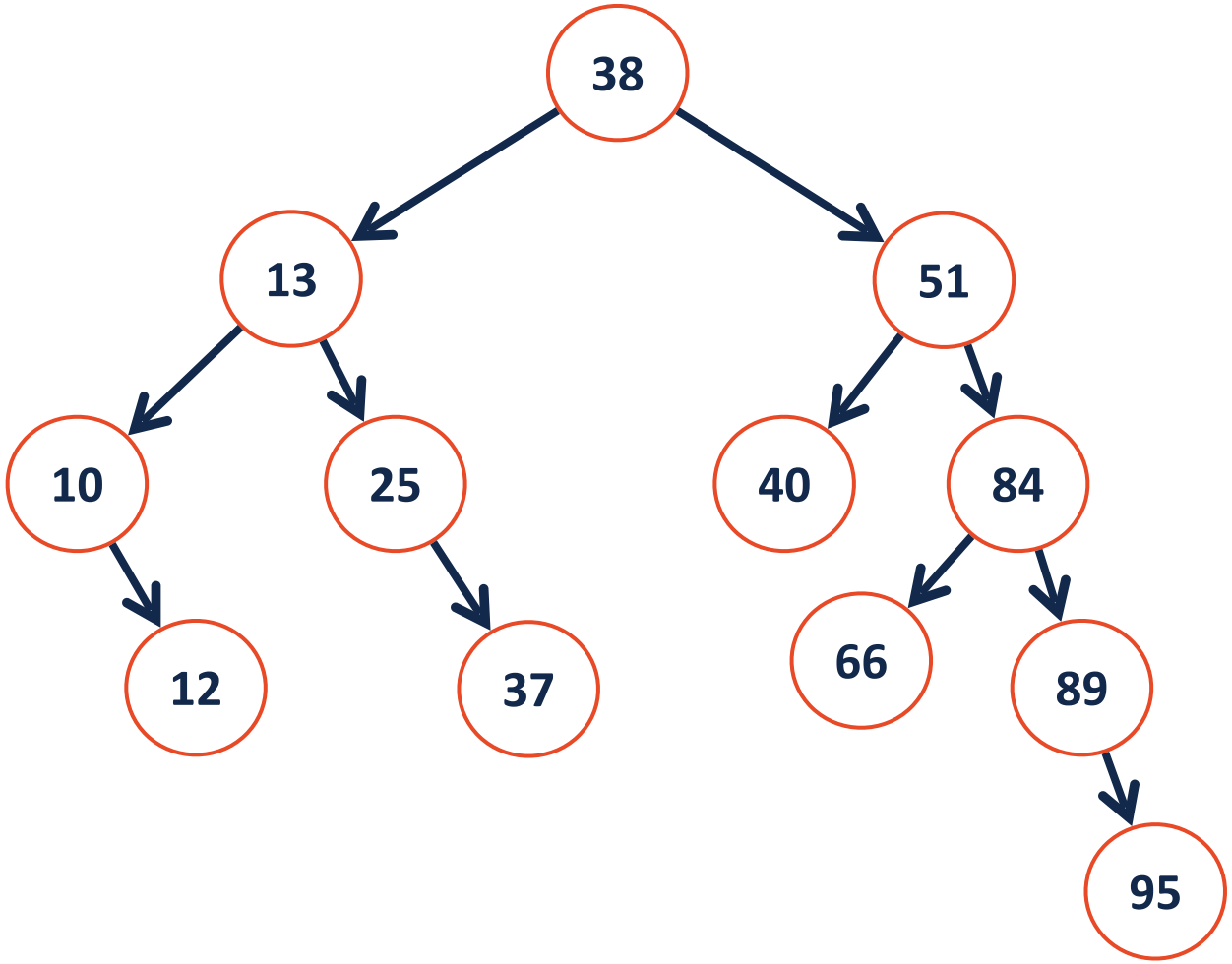
BST Remove

`remove(13)`



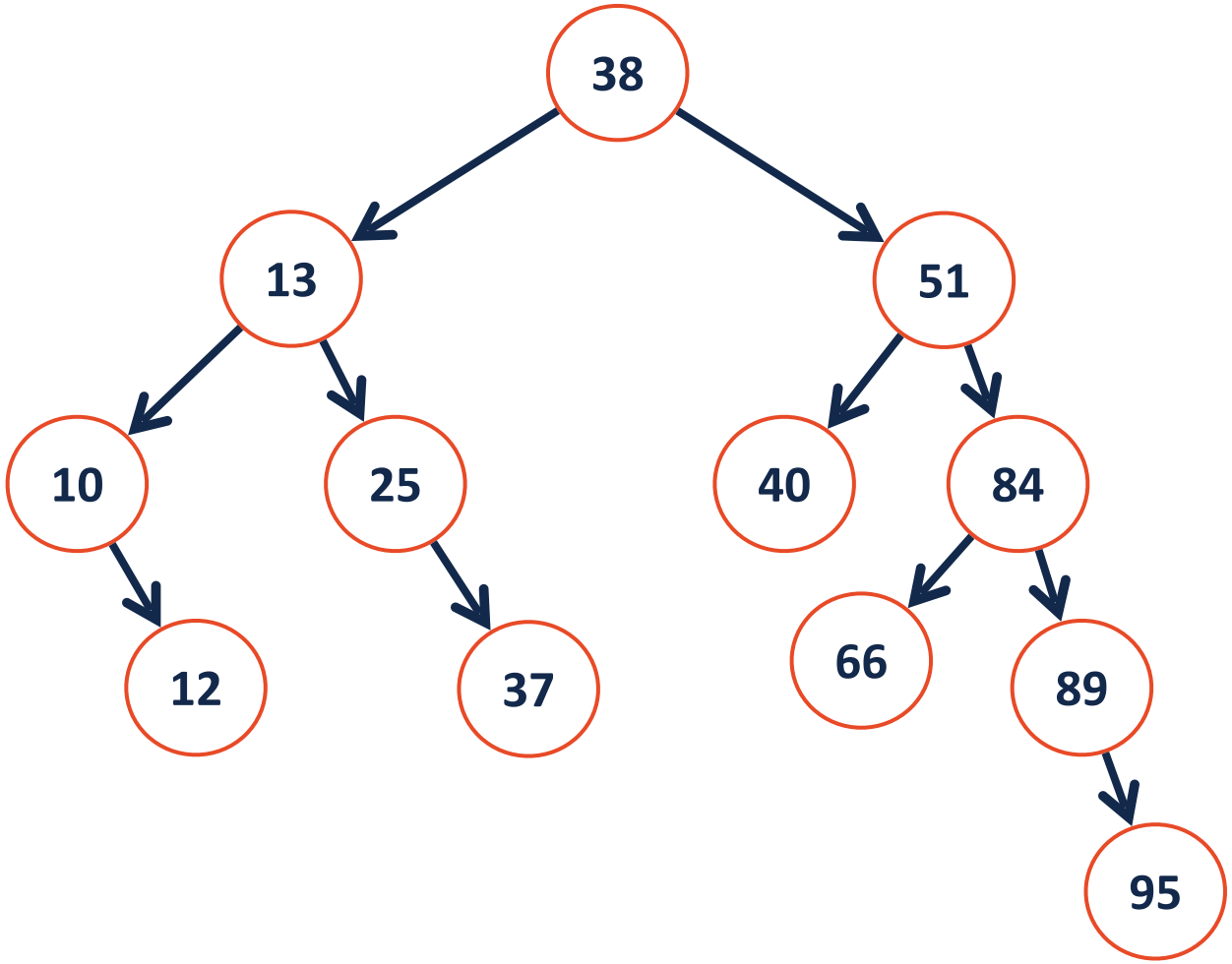
In-Order Predecessor

```
def findIOP(self, node):

    if node.left == None:
        return None

    curr = node.left

    while curr.right:
        curr = curr.right
    return curr
```



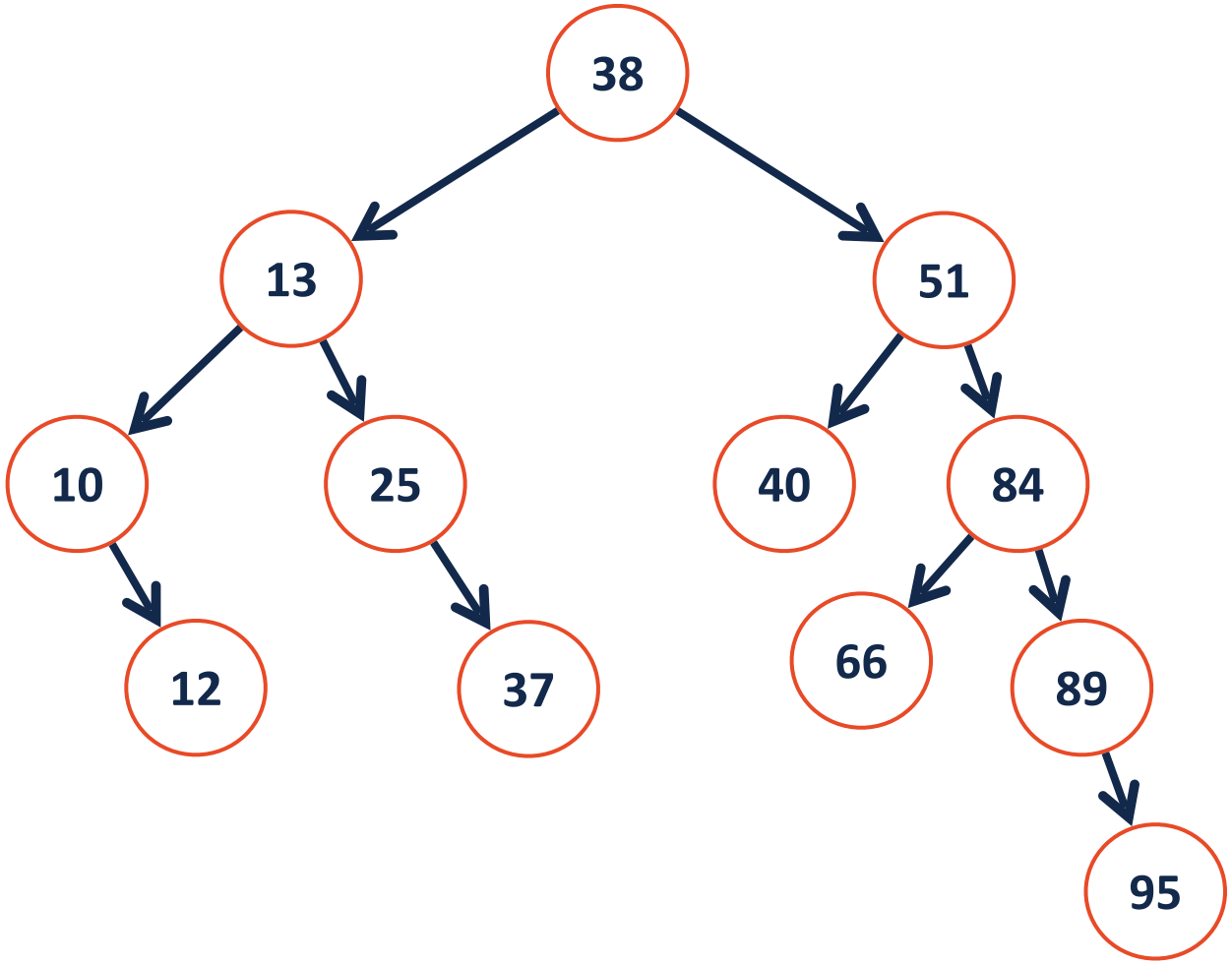
BST Remove

`remove(51)`



In-Order Successor

```
def findIOS(self, node):


    if ___________________ == None:
        return None


    curr = ___________________


    while ____________________________:


        ________________________


    return curr
```



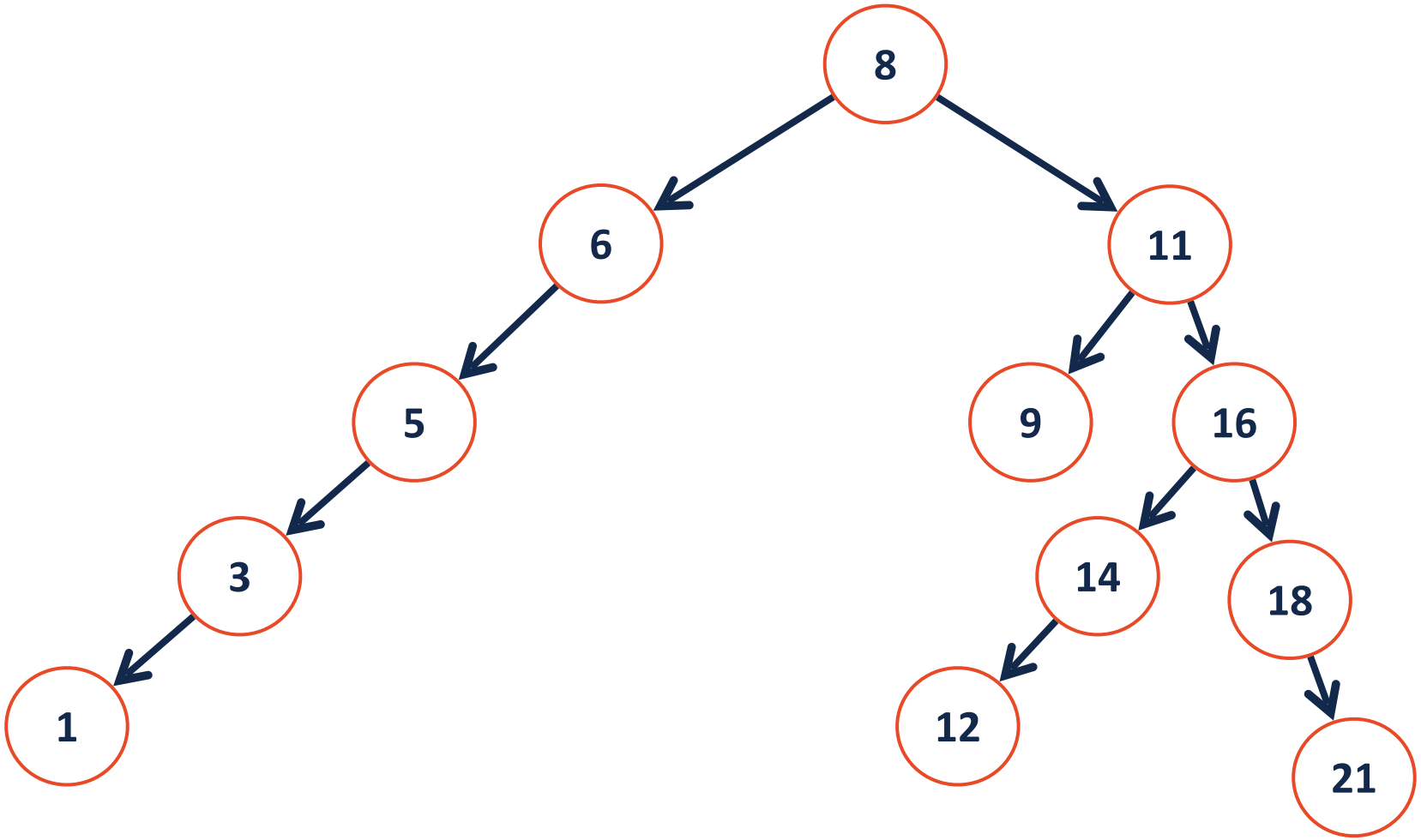
BST Remove

```
def remove(self, key):
    self.root = self.remove_helper(self.root, key)
```

```
def remove_helper(self, node, key):
```

BST Remove

What will the tree structure look like if we remove node 16 using IOS?



BST Analysis – Running Time

Operation	BST Worst Case
find	
insert	
delete	
traverse	