Data Structures

Binary Search Tree

CS 225

Brad Solomon

July 16, 2022

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science

Join Sail 2023 Staff!

All Majors Encouraged to Apply!

Want to gain experience while also helping to inspire future CS students at Illinois? Apply to join the Sail 2023 Staff! **Applications are due 9/ 19!**

Apply here!!

Follow us on Instagram to stay updated!

ILLINOIS_CS_SAIL

Committees

- logistics
- web
- marketing
- design

Learning Objectives

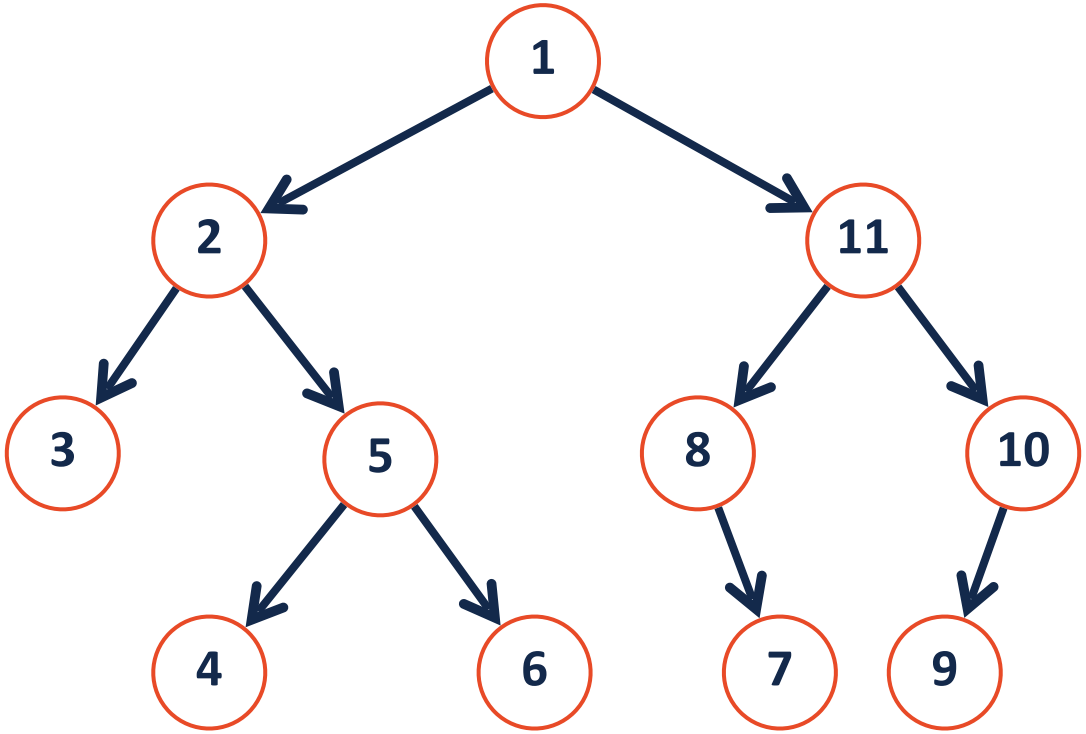
Review binary trees

Introduce the binary search tree

Conceptualize and pseudo-code BST ADT

Discuss the key weakness of a BST and foreshadow improvements

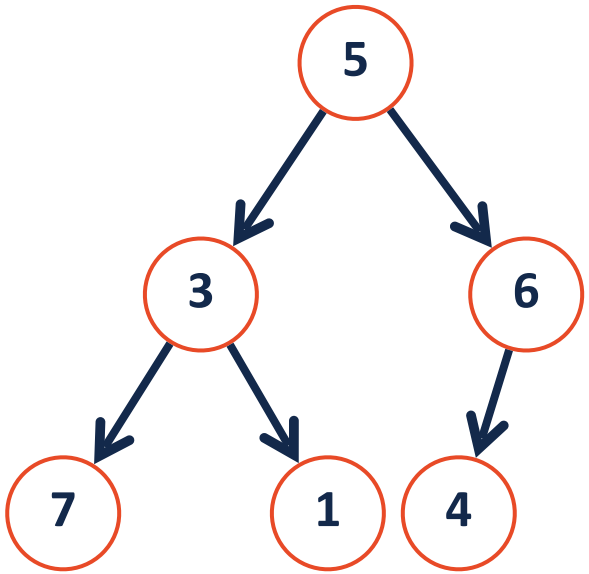
Binary Trees

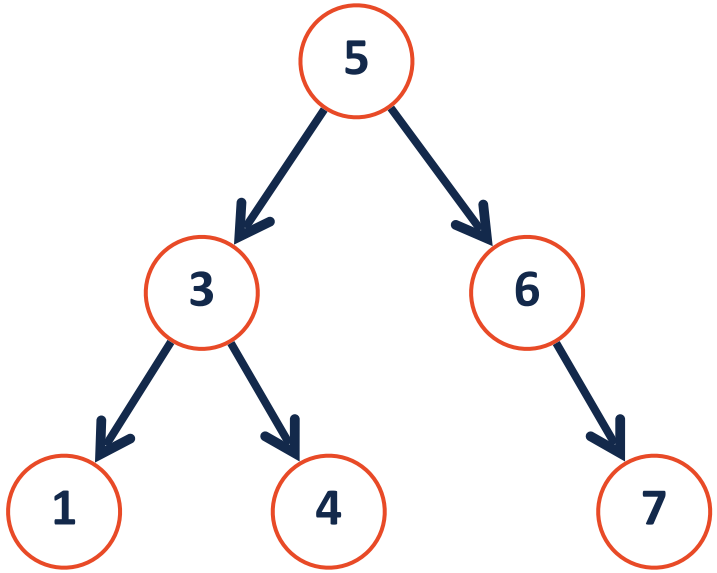


Improved search on a binary tree

7	3	1	5	4	6

1	3	4	5	6	7





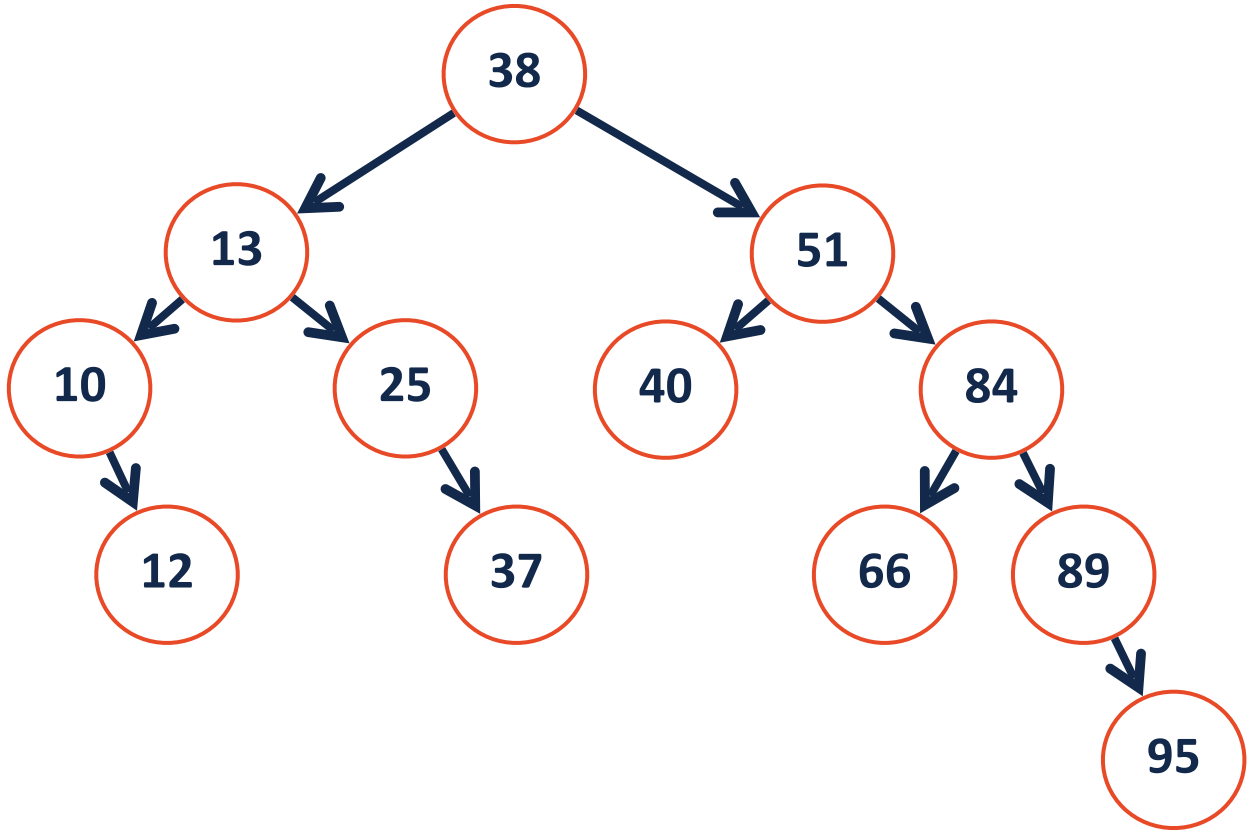
Binary Search Tree (BST)

A **binary search tree** T is either:

-

OR

-

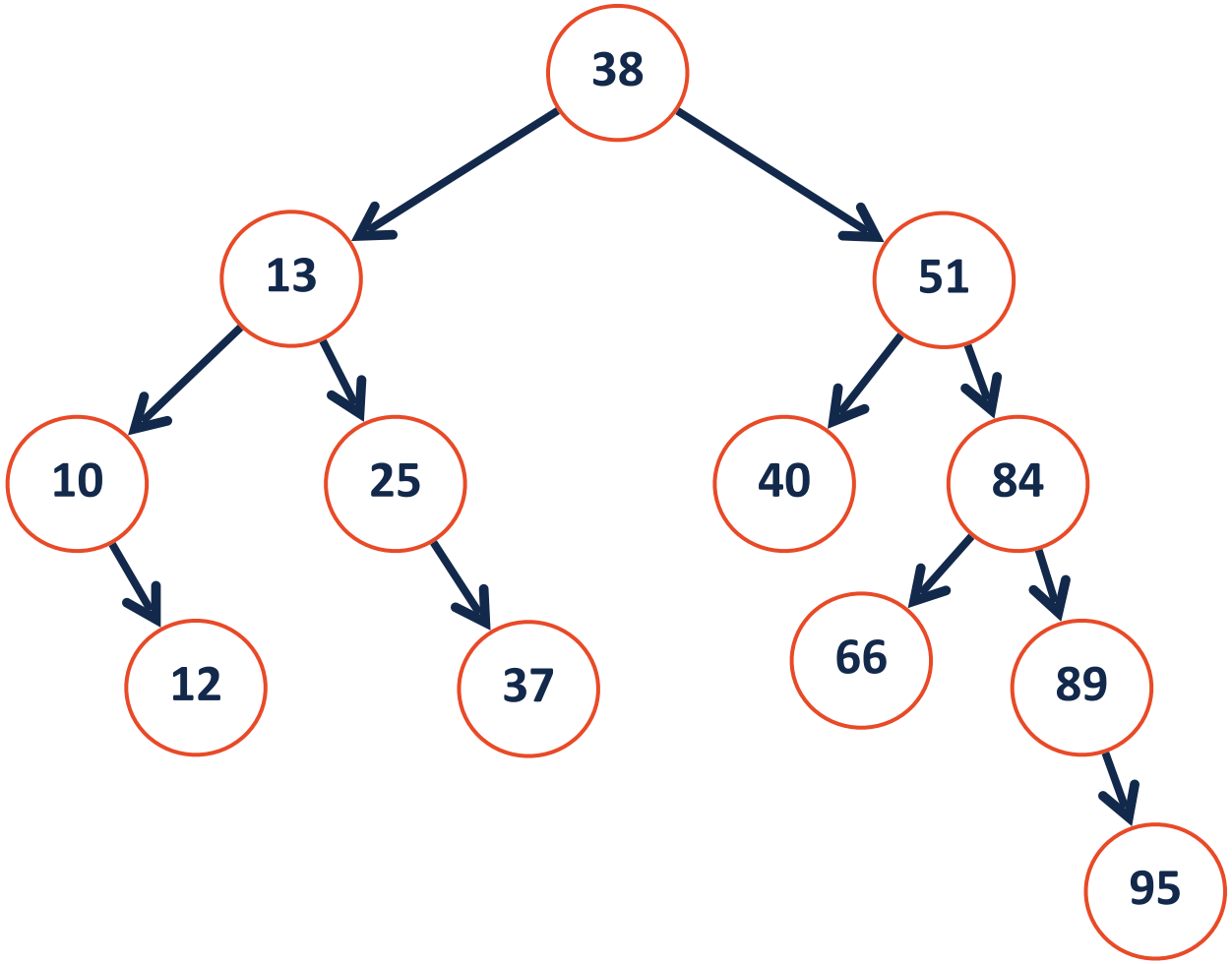


BST.h

```
 1 #pragma once
 2
 3 template <typename K, typename V>
 4 class BST {
 5   public:
 6     BST();
 7     void insert(const & K key, const V & value);
 8     V remove(const K & key);
 9     V find(const K & key) const;
10
11   private:
12     struct TreeNode {
13        TreeNode *left, *right;
14        K  key;
15        V  value;
16        TreeNode();
17     };
18
19     TreeNode *head_;
20 };
21
22
23
```

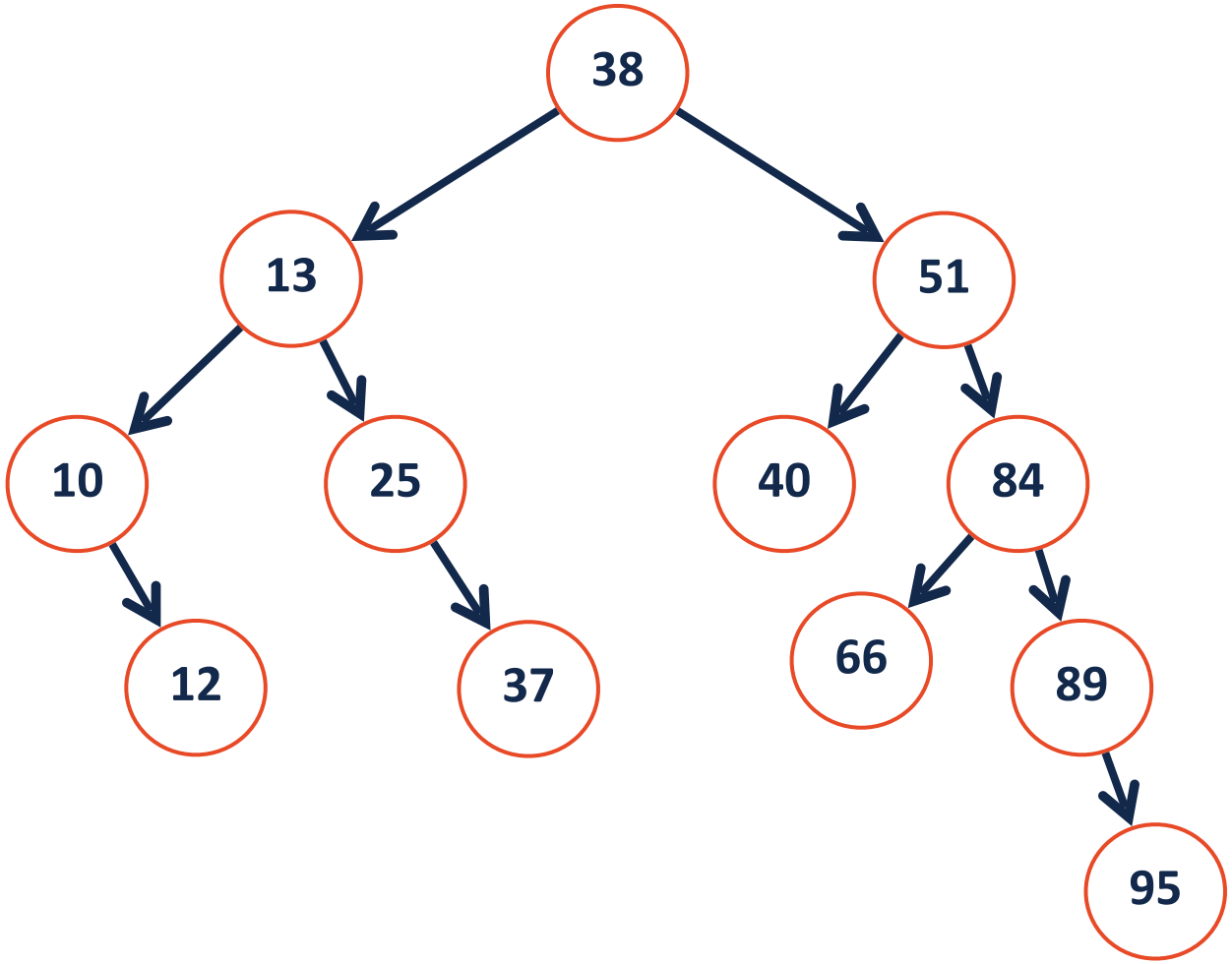
BST Find

`find(66)`



BST Find

`find(9)`

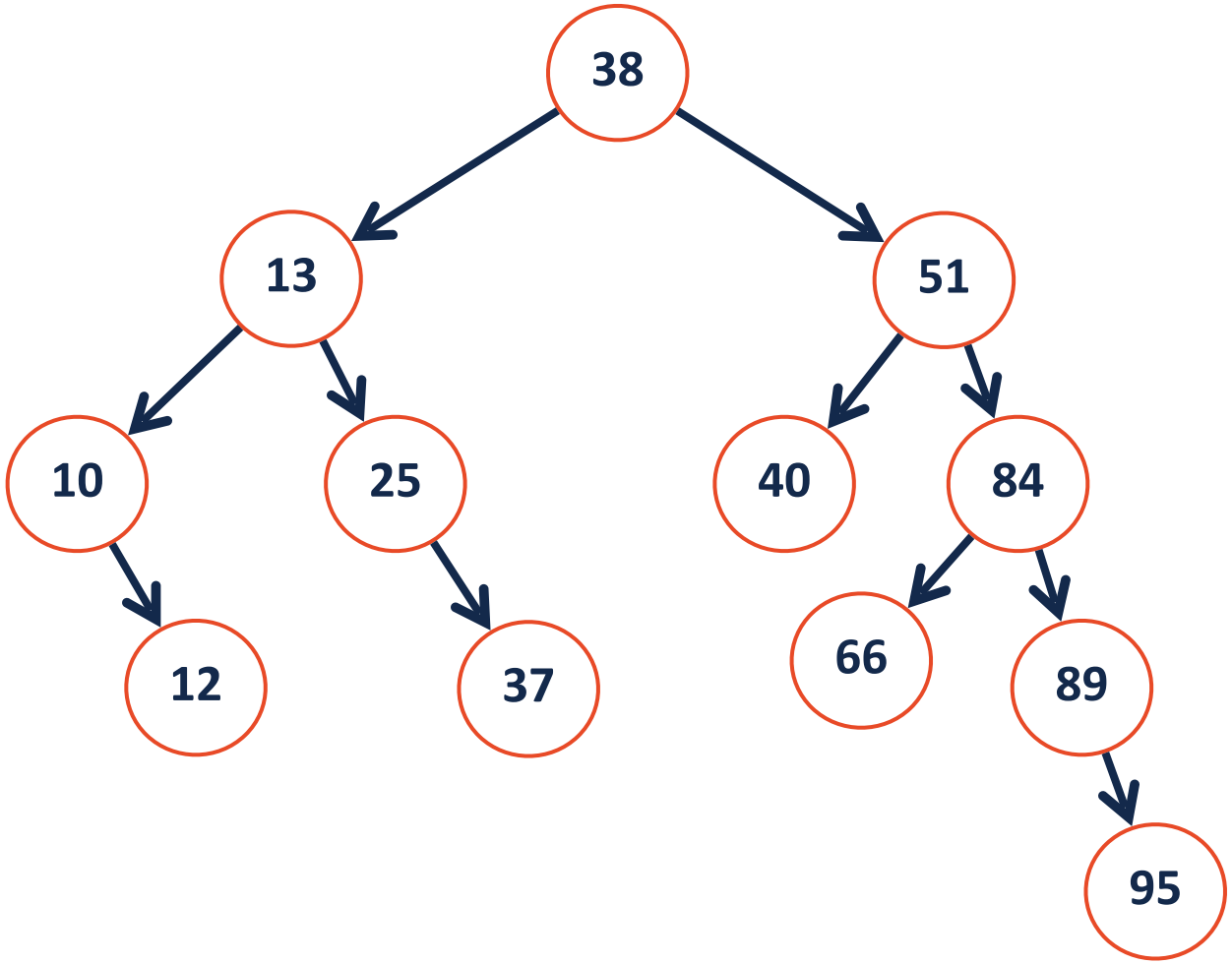


```
template<typename K, typename V>

_____________________________ _find(TreeNode *& root, const K & key) {









}
```

```
template<typename K, typename V>

V find(const K & key) const {









}
```

BST Insert

`insert(33)`



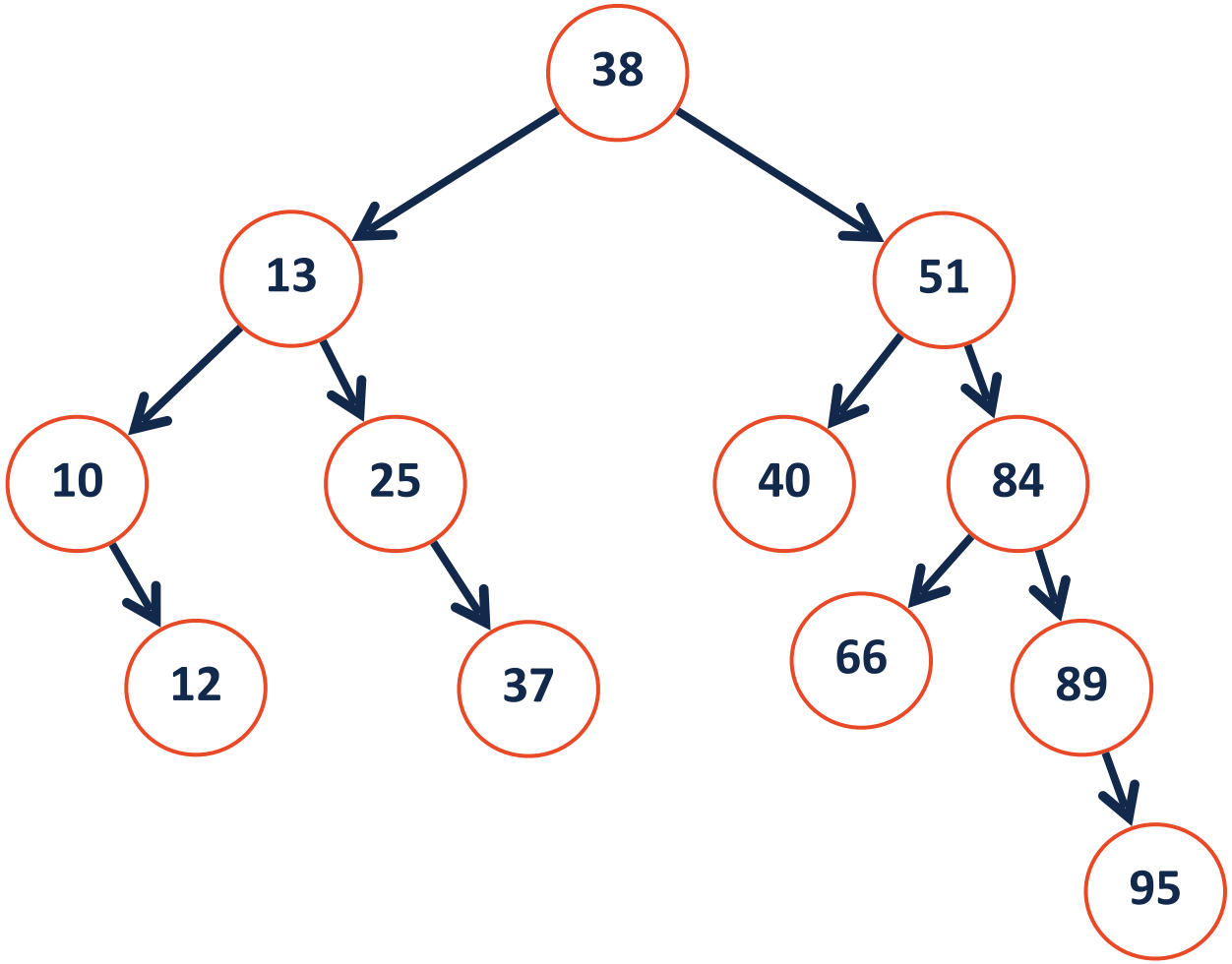
```
1  template<typename K, typename V>
2
3  void insert(const K & key, const V & val) {
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23 }
```

BST Insert

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

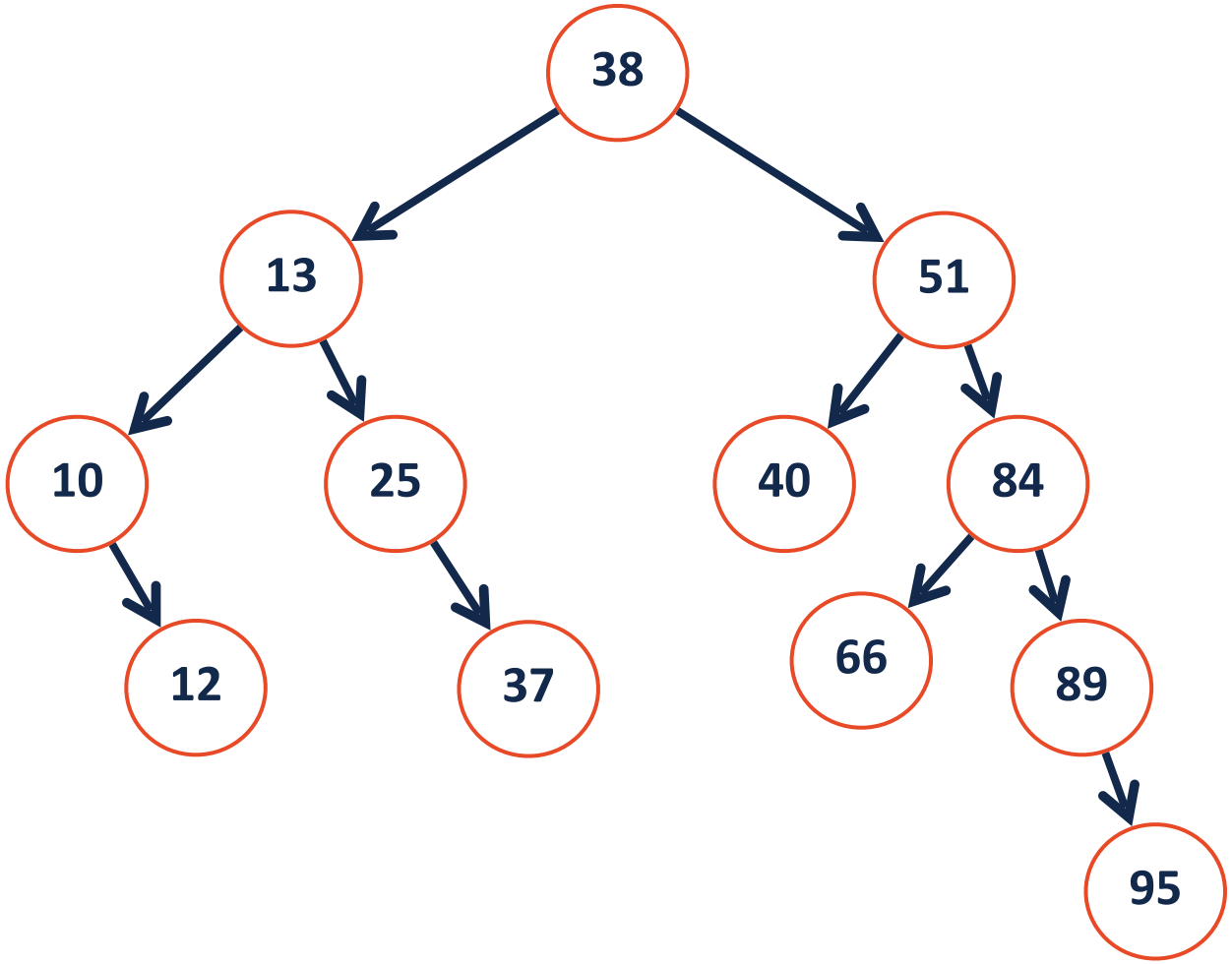
BST Remove

`remove(40)`



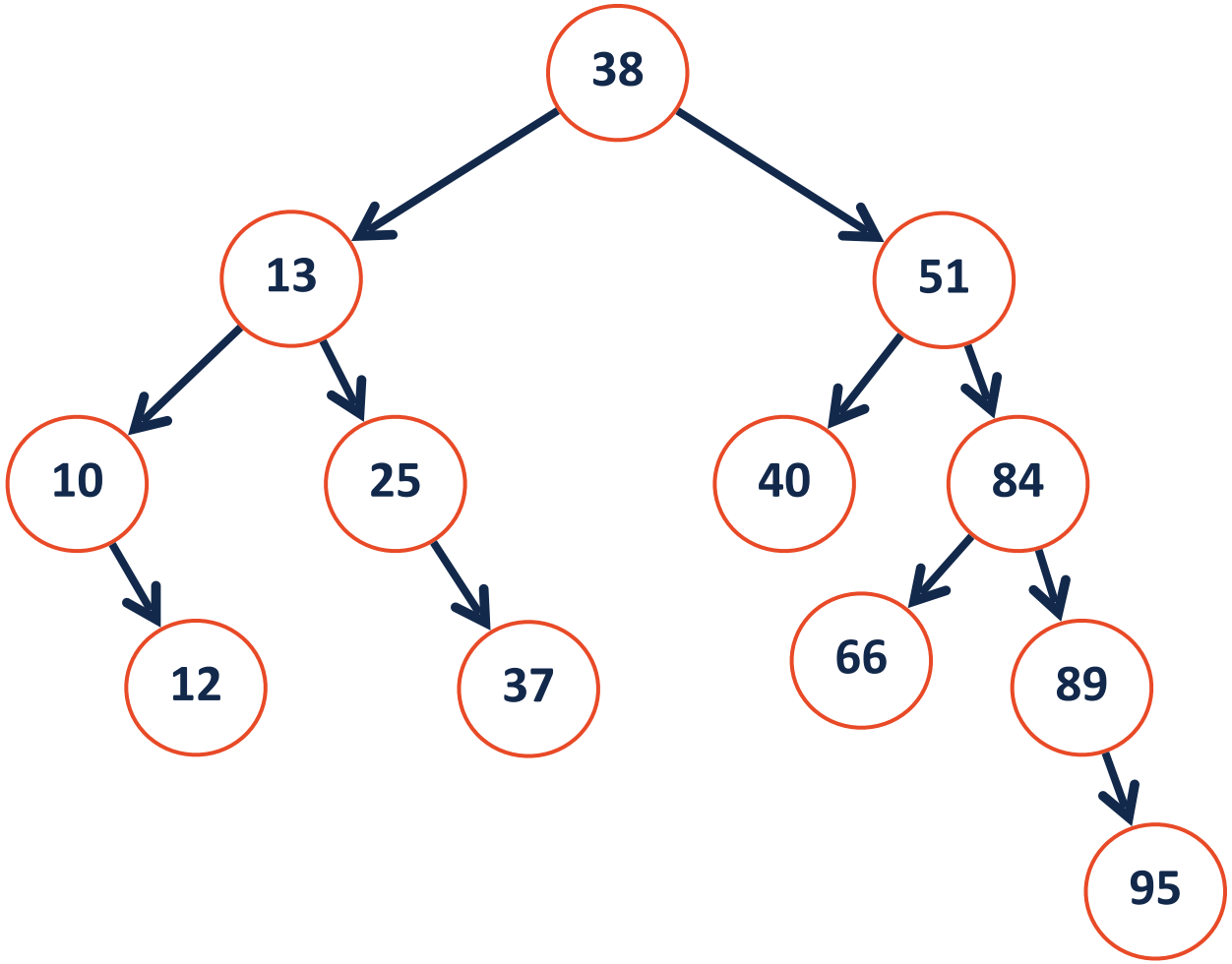
BST Remove

`remove(25)`



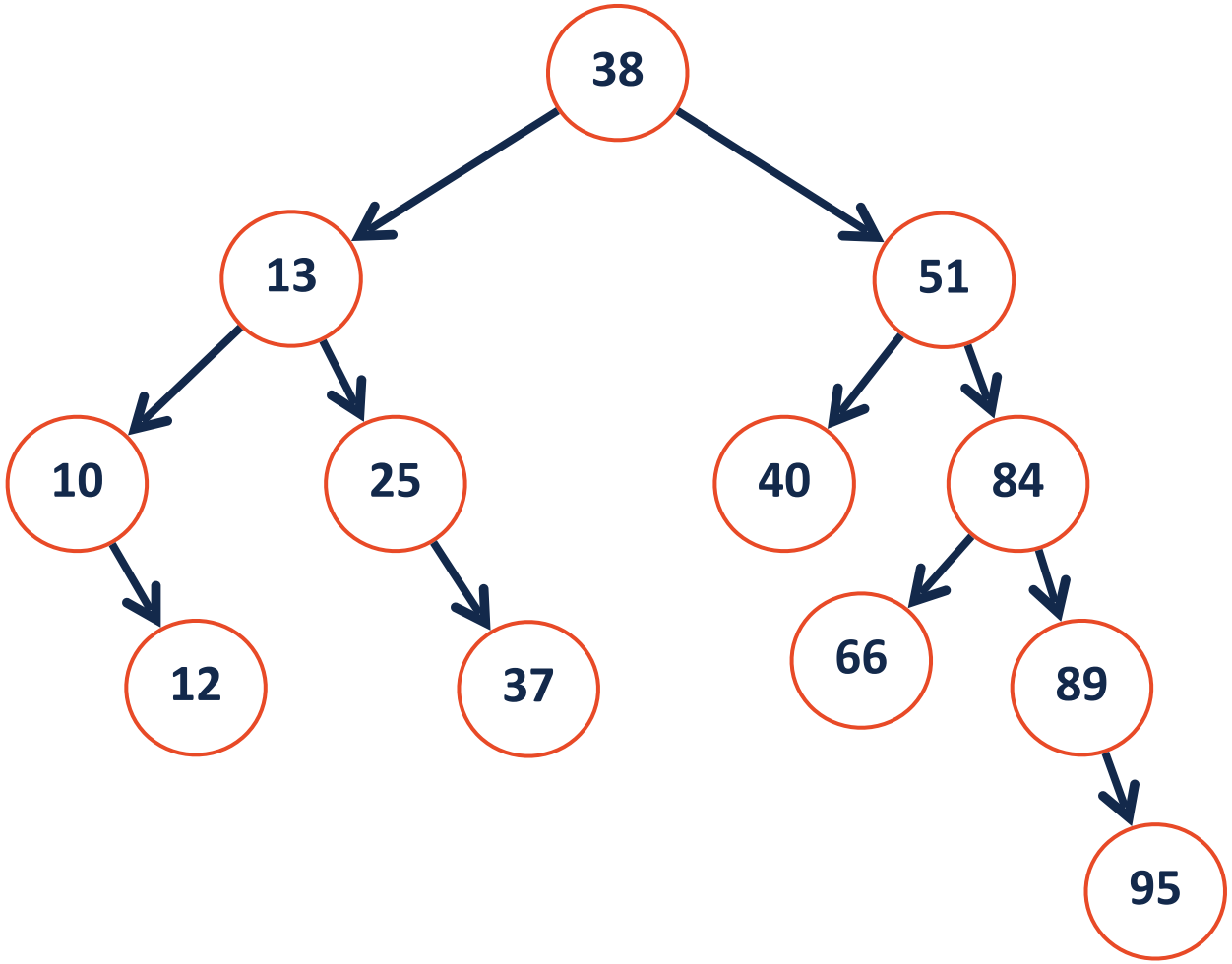
BST Remove

`remove(13)`



BST Remove

`remove(51)`



BST Analysis – Running Time

Operation	BST Worst Case
find	
insert	
delete	
traverse	

BST Analysis

Every BST operation that we have studied depends on the height **O(h)**

… how can we relate this in terms of **n**, the amount of data?

BST Analysis

What is the **max** number of nodes in a tree of height h ?

BST Analysis

What is the **min** number of nodes in a tree of height h ?

BST Analysis

The height of a BST depends on the order in which the data was inserted

Insert Order: [1, 3, 2, 4, 5, 6, 7]

Insert Order: [4, 2, 3, 6, 7, 1, 5]

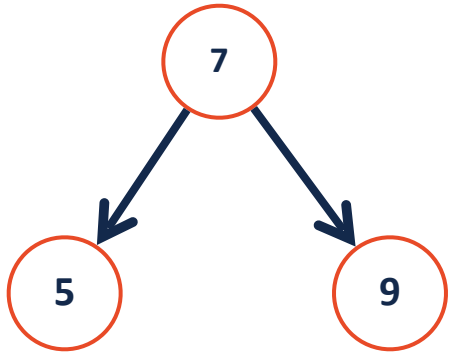
BST Analysis

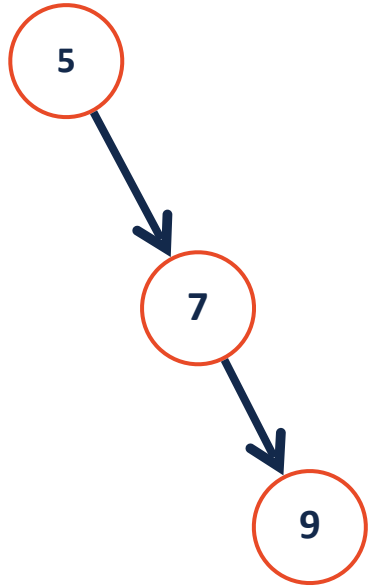
How many different ways are there to insert n keys into a BST?

Claim: The average height of all arrangements is $O(log\ n)$

Height-Balanced Tree

What tree is better?



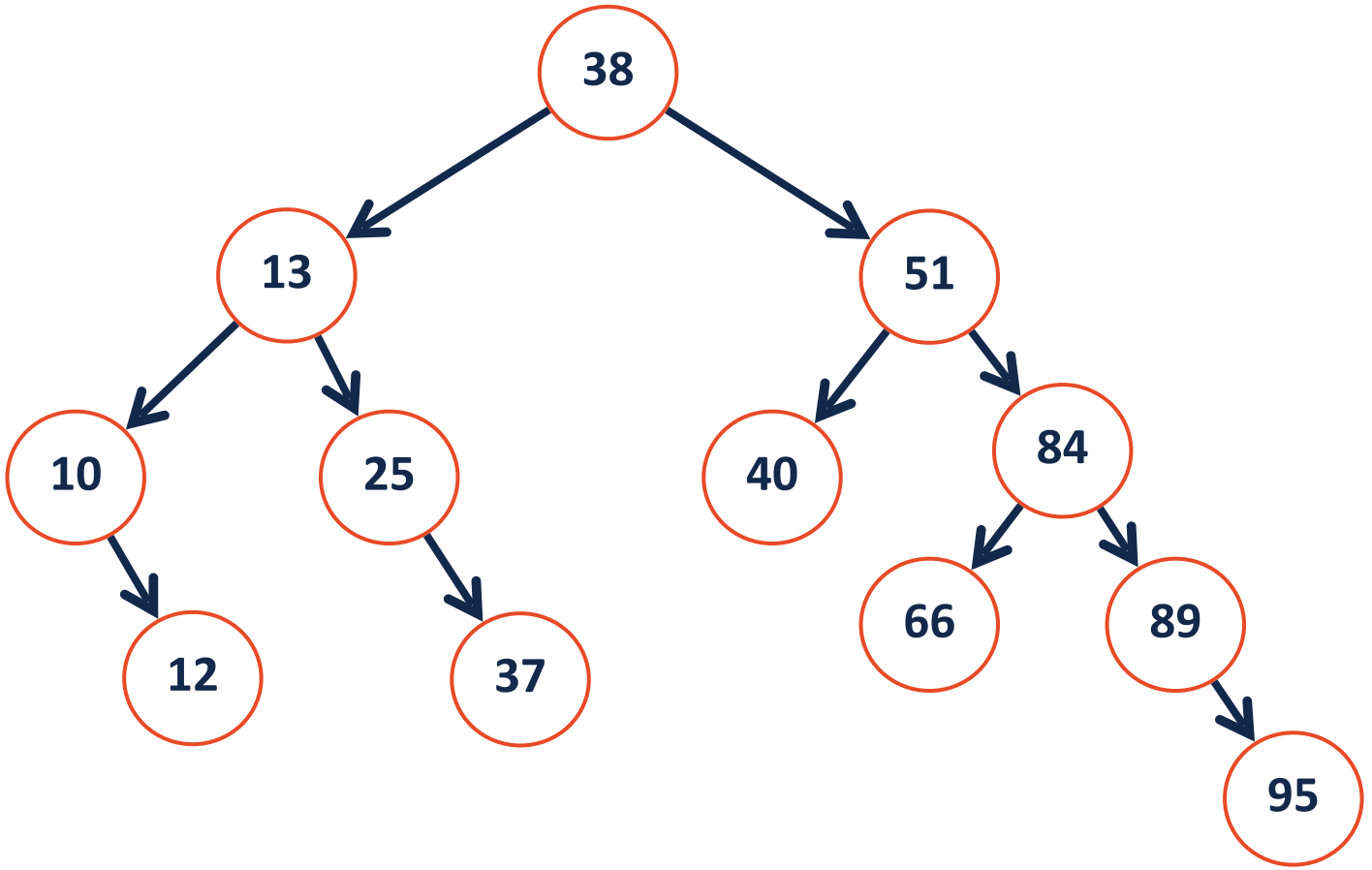


Height balance: $b = height(T_R) - height(T_L)$

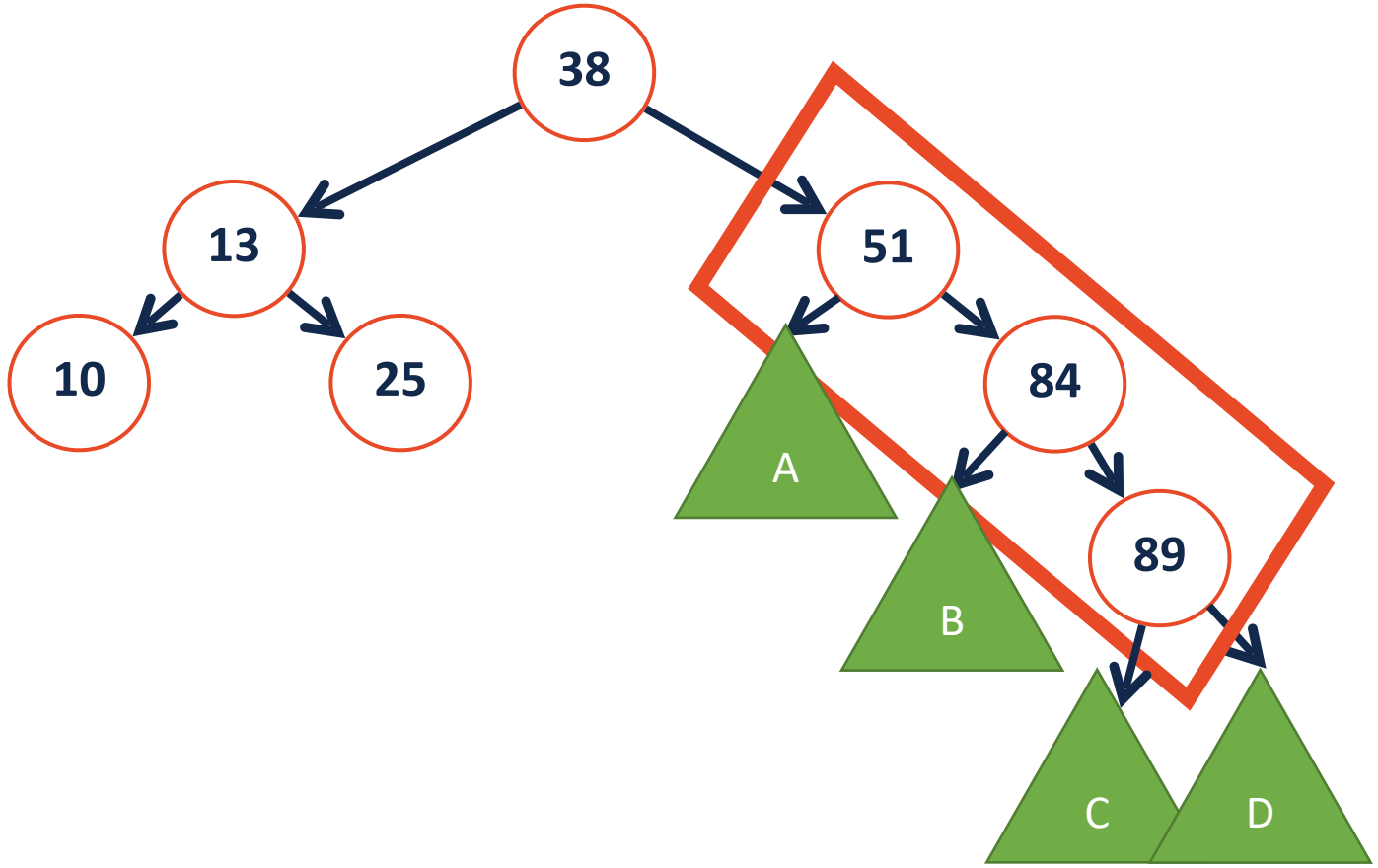
A tree is "balanced" if:

BST Rotations

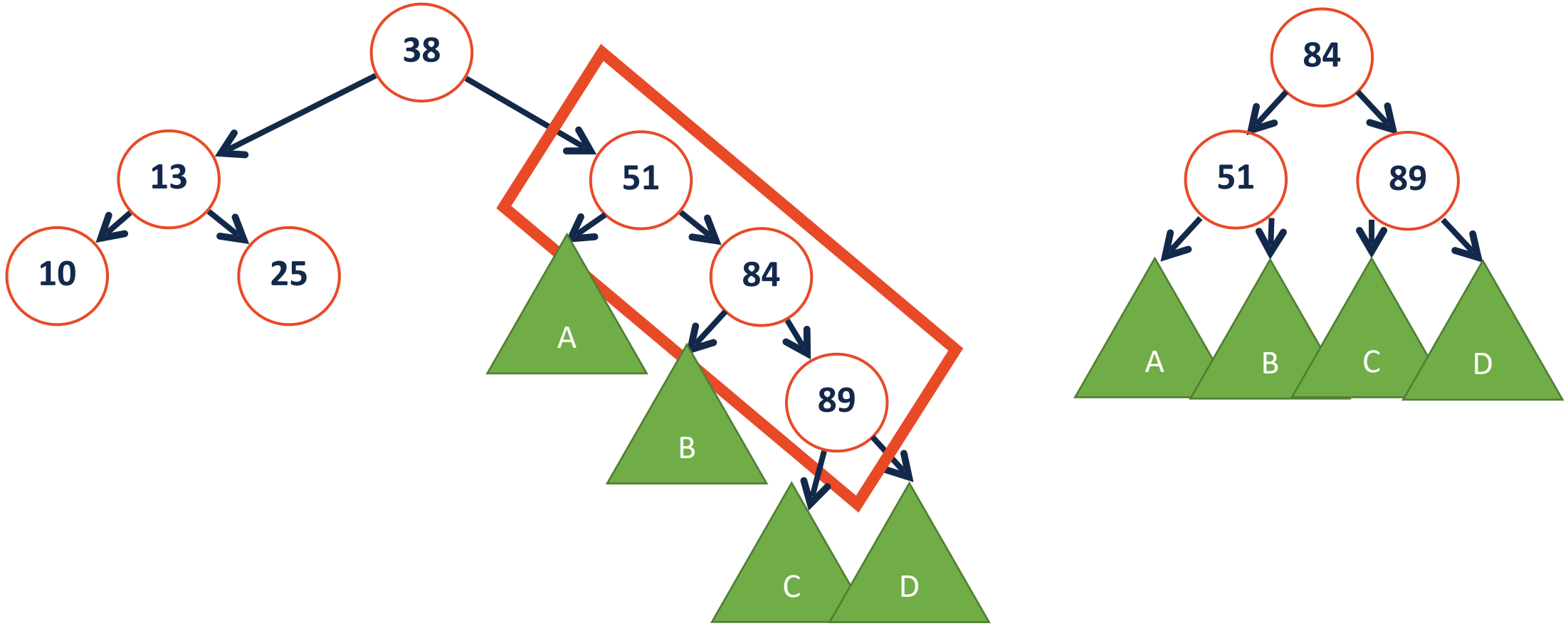
We can adjust the BST structure by performing **rotations**.



BST Rotations



BST Rotations



BST Rotations

Next week we will define four kinds of rotations (L, R, LR, RL)

We will see that:

1. All rotations are local (subtrees are not impacted)
2. The running time of rotations are constant
3. The rotations maintain BST property

Motivation for rotations:

We call these trees: