Data Structures & Algorithms

FG24GP

Today

- Binary Search Tree – Node Deletion
- What is a balanced tree? … and why do we want it?
- The AVL Tree

  (A self balancing tree)

A Tree

Has nodes where each node has a list of children

A Binary Tree

Has nodes where each node has 2 children (left & right)

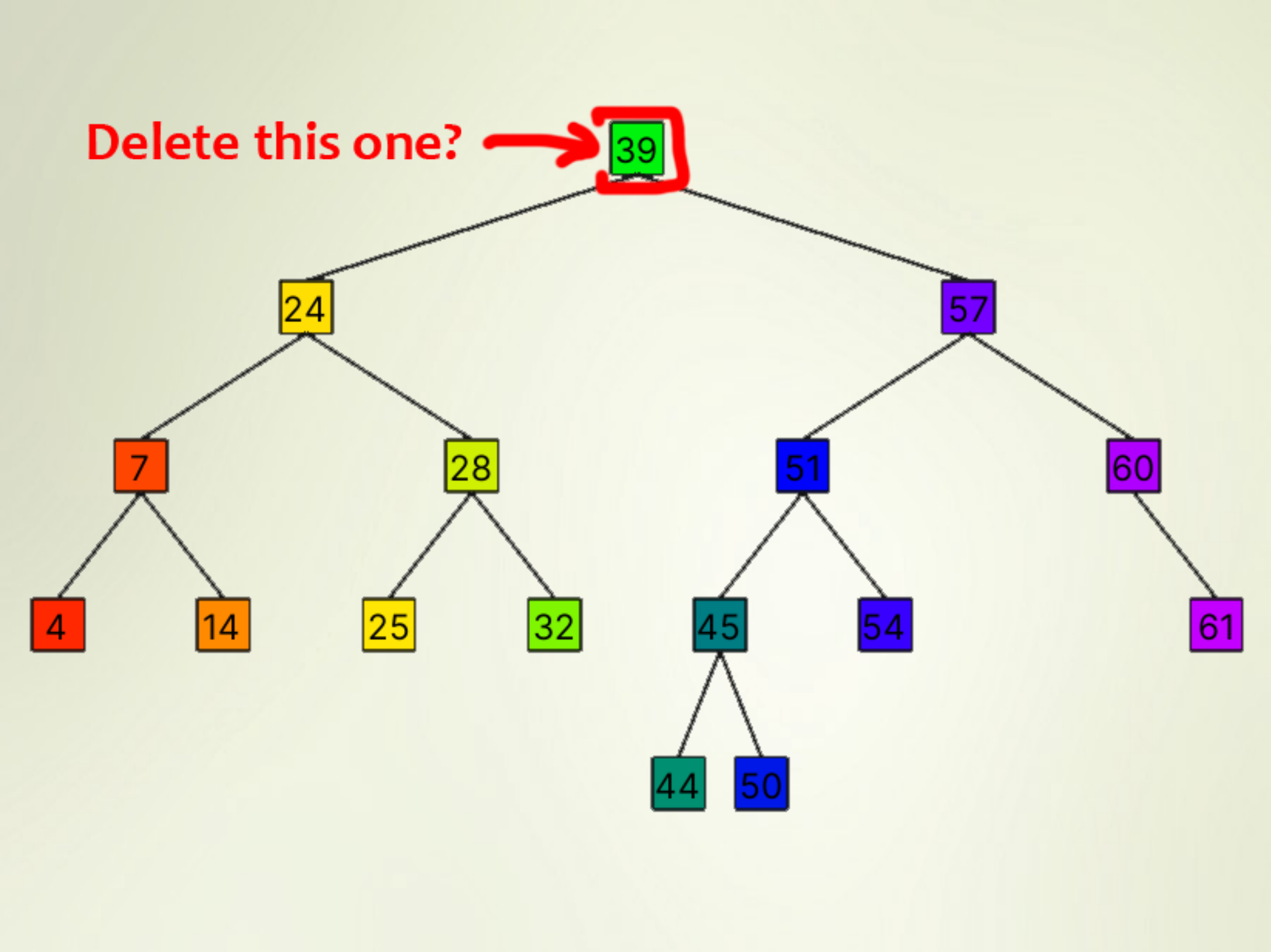
A Binary Search Tree

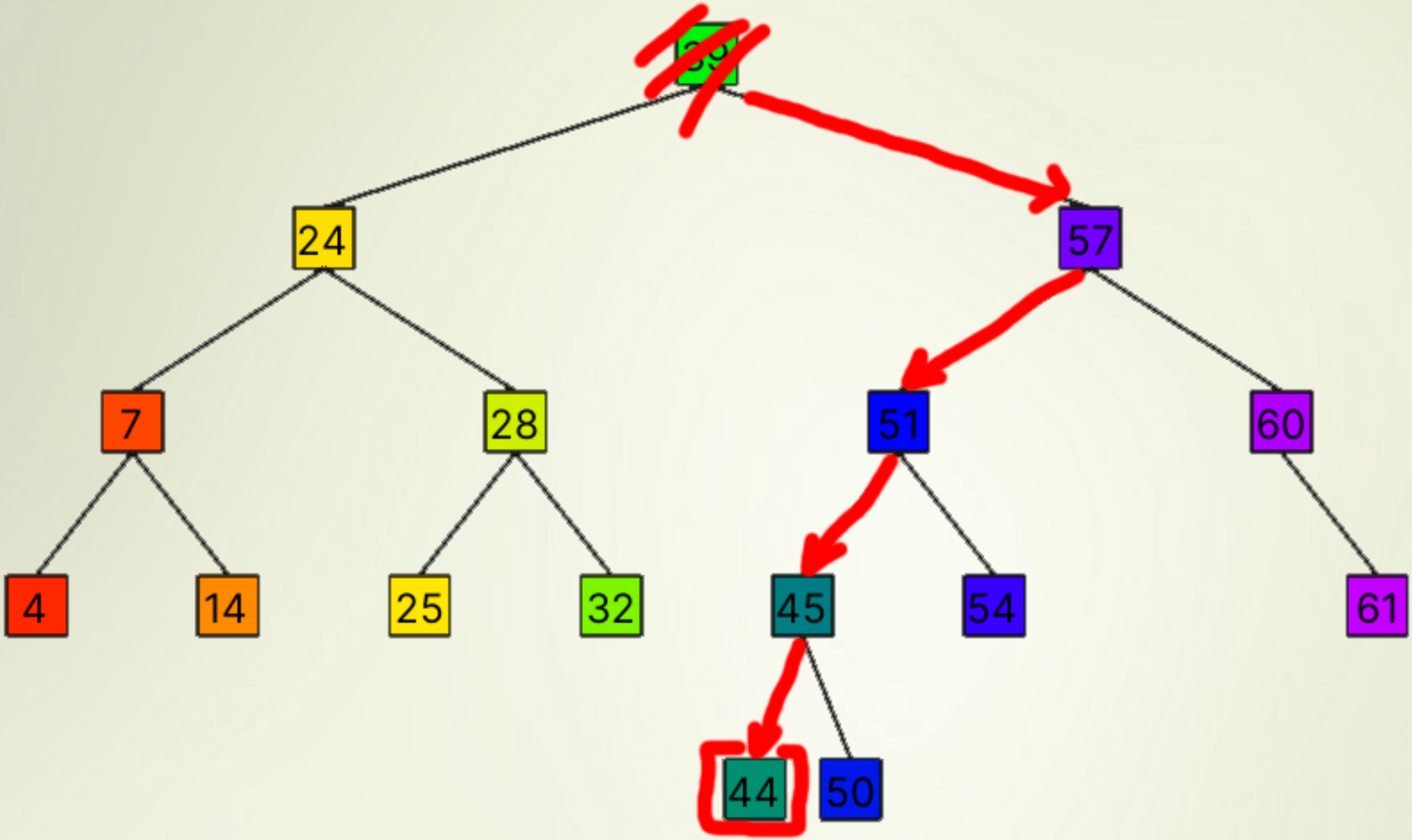
Is a Binary Tree where the data is sorted

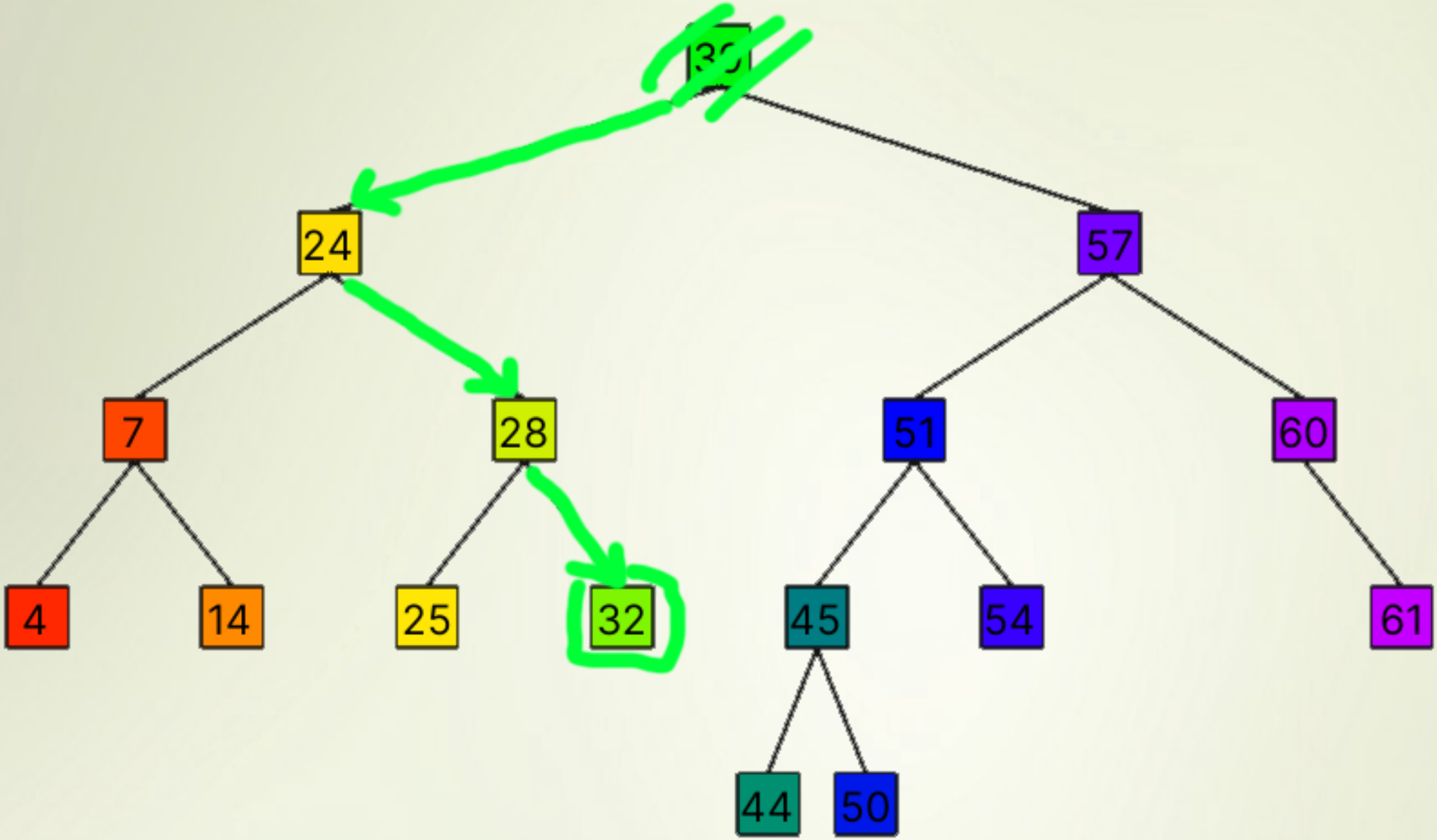
(usually smaller values into the left subtree, and larger values into the right subtree)

Binary Search Tree
Node Deletion

- A leaf Node? Easy!
- Single Child Node? No Problem!
- Two Kids? … A bit more Complicated.


Delete this one?
39
24
57
7
28
51
60
4
14
25
32
45
54
61
44
50


24
57
7
28
51
60
4
14
25
32
45
54
61
44
50

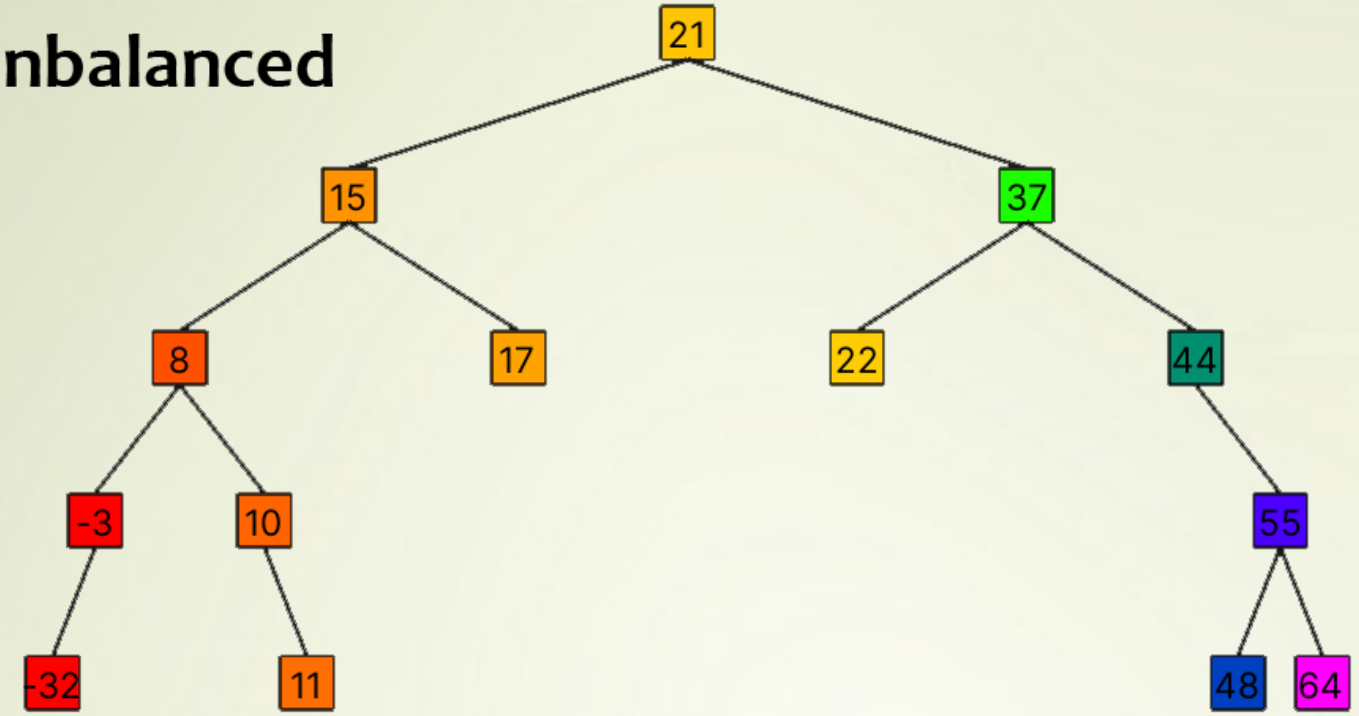

24
57
7
28
51
60
4
14
25
32
45
54
61
44
50

A Balanced Tree

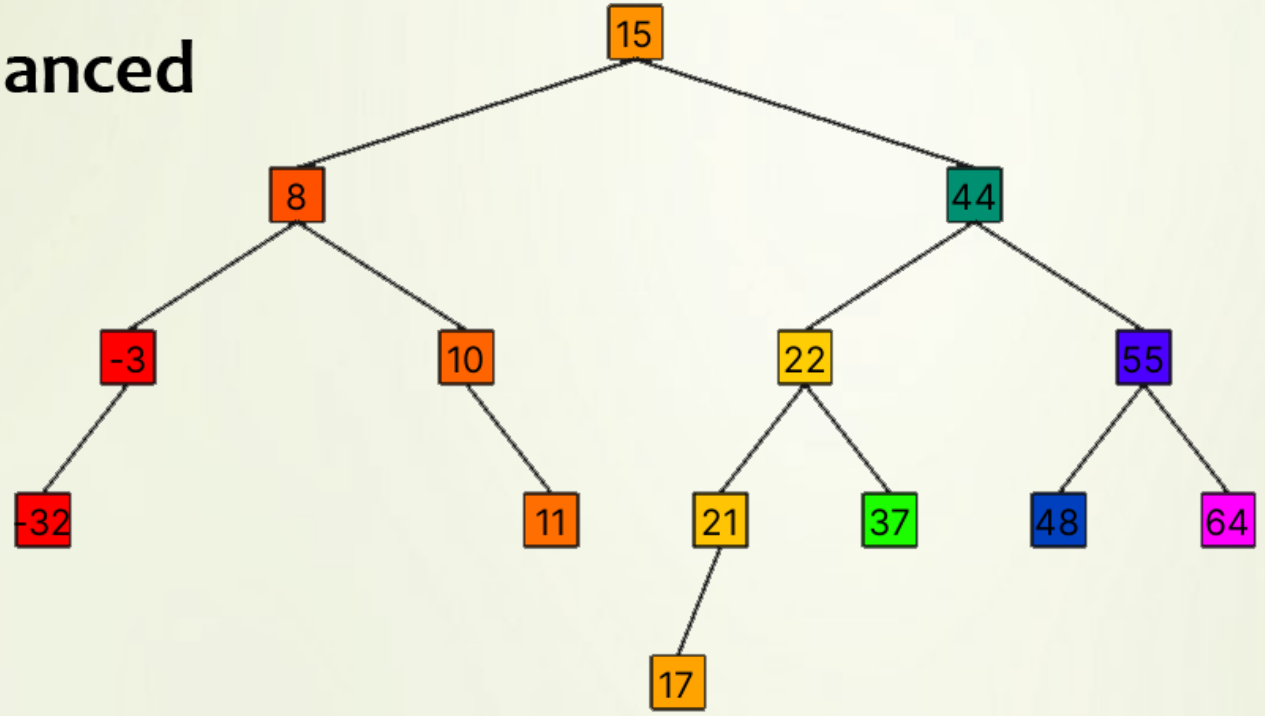
- A balanced tree is a tree where the depth difference of its right and left subtree is less or equal to 1

- Why do you think this is important?

Unbalanced



Balanced



AVL Tree

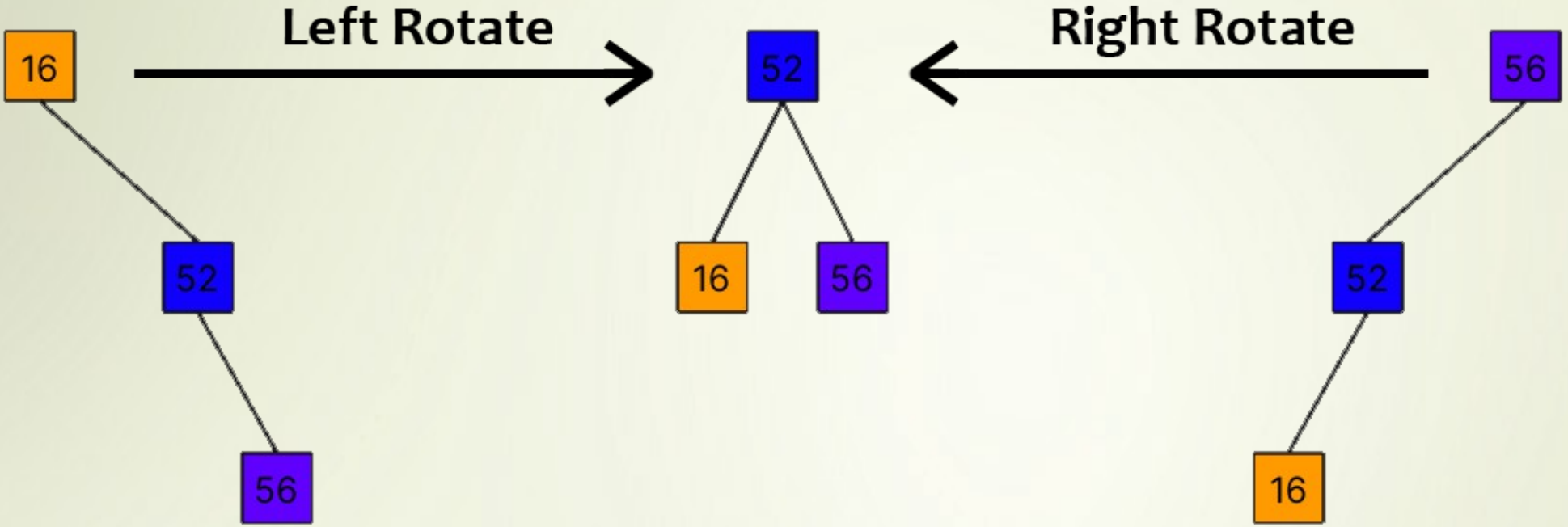
- Named after its creators:

  *Georgy **A**delson-**V**elsky and Evgenii **L**andis*

- Published in 1962

- A tree that maintains balance for insertions and deletions

Two Rotations



Exercises

Task #1 – Find Node

- Create a Find() function that given a value (key) looks up the **Node** of a Binary Search Tree and returns it

Task #2 – Find Range

- Write a function that searches a Binary Search Tree and returns all **values** that lies within a range
- Think about how to best return multiple values that doesn't cause unnecessary memory allocations

Task #3 – Tree Compare

- Create an array filled with 1000 random (but unique) numbers
- Create a **Binary Search Tree** (BST) using the array
- Create an **AVL Tree** using the array
- Using the Find() function from Task #1, perform 10000 lookups with a random number from the array on both the BST & AVL tree.
- Time the difference for the BST vs the AVL tree