2-3 Insertion

This lesson will explain how insertion is done in 2-3 Trees based on multiple scenarios which are explained in the insertion algorithm.

We'll cover the following

Introduction

Insertion in 2-3 Trees is a lot different from Binary Search Trees. In 2-3 Trees, values are only inserted at leaf nodes based on certain conditions. As discussed before, the insertion algorithm takes $O(Logn)$ time where n is the number of nodes in the tree. Searching an element is done in $Log(n)$ and then insertion takes a constant amount of time. So overall the time complexity of insertion algorithm is $O(Logn)$. Let's see how it works.

Insertion Algorithm:

The insertion algorithm is based on these scenarios:

- Initially if the tree is empty, create a new leaf node and insert your value
- If the tree is not empty, traverse through the tree to find the right leaf node where the value should be inserted
- If the leaf node has only one value, insert your value into the node
- If the leaf node has more than two values, split the node by moving the middle element to the top node
- Keep forming new nodes wherever you get more than two elements

Example -1

Let's take a look at the following example where we will build a 2-3 Tree from scratch by inserting elements one by one.

Insert 50!

1 of 22

50

2 of 22

Insert 30!

50

3 of 22

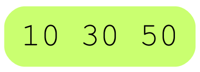
30 50

4 of 22

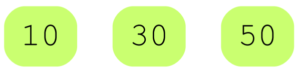
Insert 10!

30 50

5 of 22

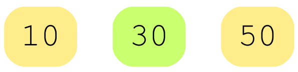


6 of 22



Split the nodes

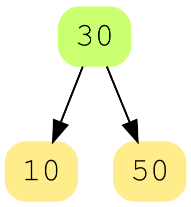
7 of 22



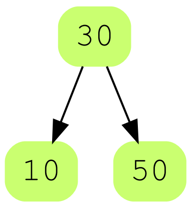
Move the mid key up

8 of 22

Make it parent of rest of the two!

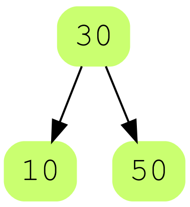


9 of 22



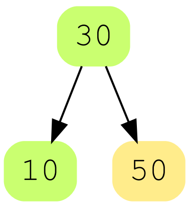
10 of 22

Insert 70!

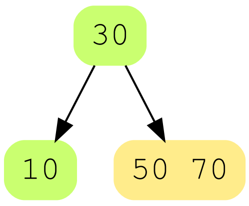


11 of 22

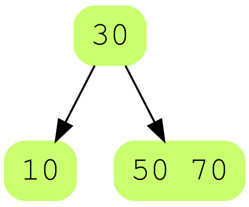
Traverse to the right node!



12 of 22

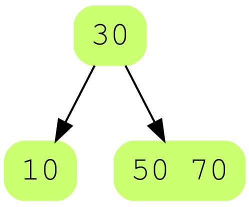


13 of 22

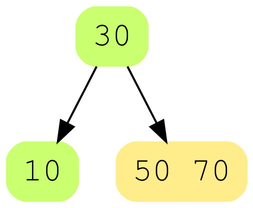


14 of 22

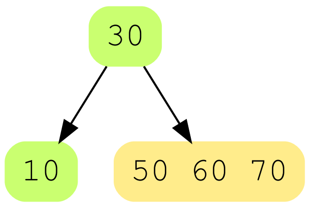
Insert 60!



15 of 22

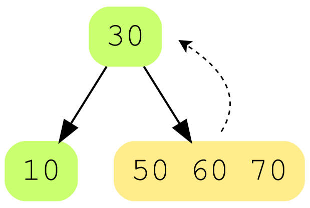


16 of 22

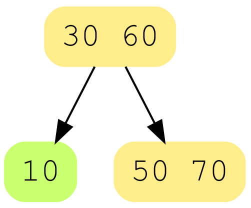


17 of 22

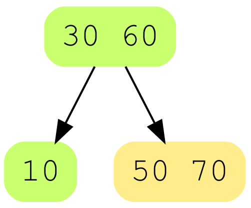
Again, move the mid key up!



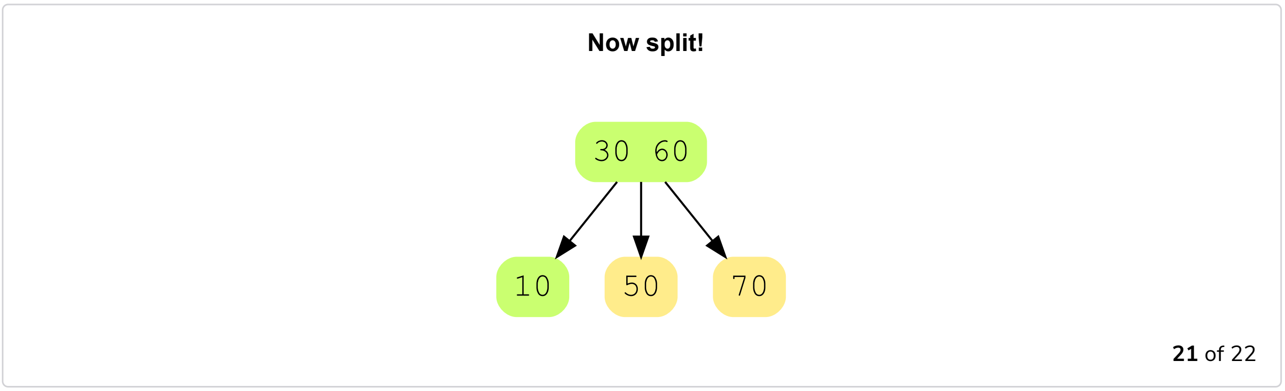
18 of 22

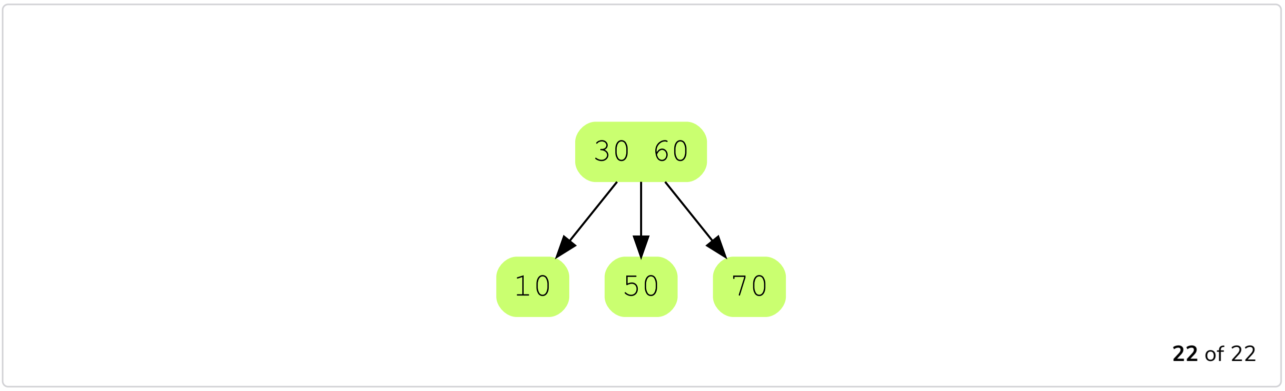


19 of 22



20 of 22





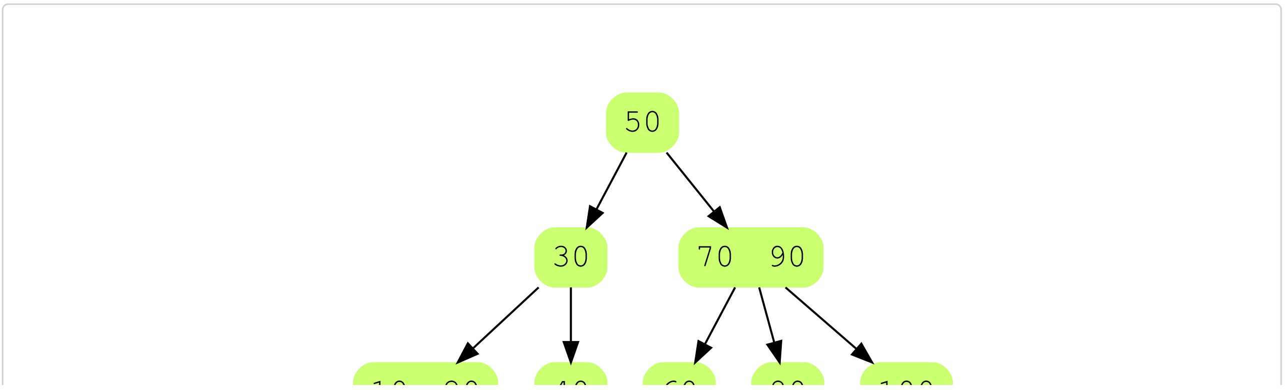
Explanation:

As you see, the tree is initially empty, so we will create a new node and insert **50** in it. Then we will insert **30** in the same node as it has one space left. Then insert **10**, but since the node can only contain two values at max, we shift the median value to the top and split the node into two children, this makes **30** the root node with **10** and **50** as its left and right child respectively.

Now we insert **70**, as **70** is greater than the root key so it will be inserted in the right child. Similarly, **60** will be inserted in the same node, but as you can see, the values got more than two again so we will perform the same series of step again, shifting **60** to the root node, and so on.

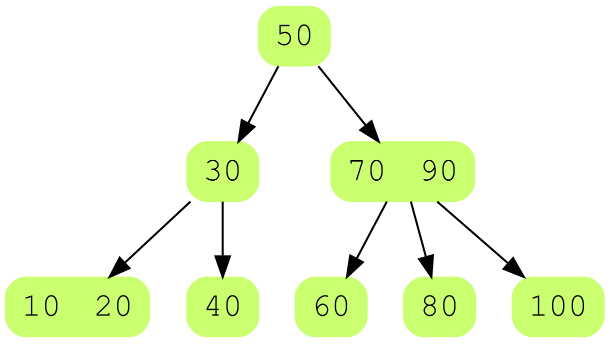
Example - 2

This example is a little harder than the previous one. See if you can solve it on your own!



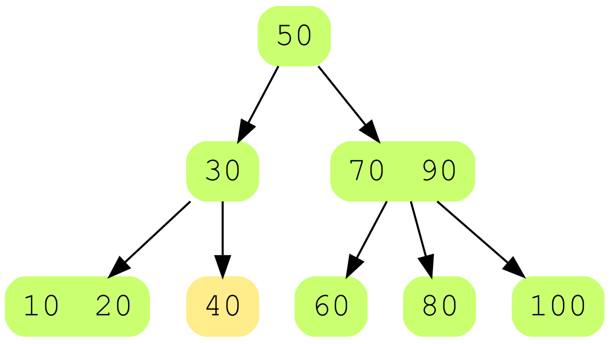
1 of 29

Insert 39!

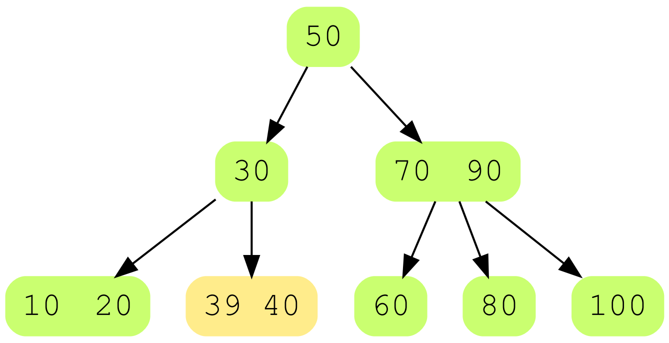


2 of 29

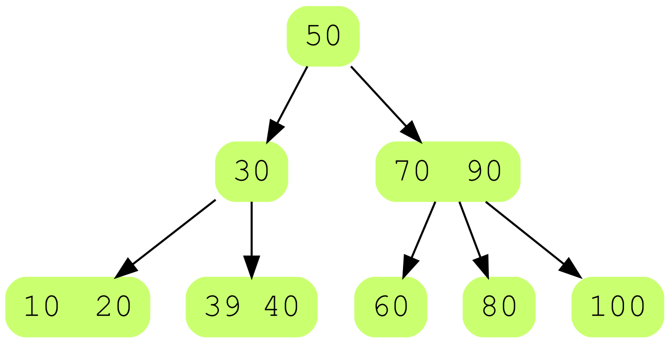
Traverse to the right node



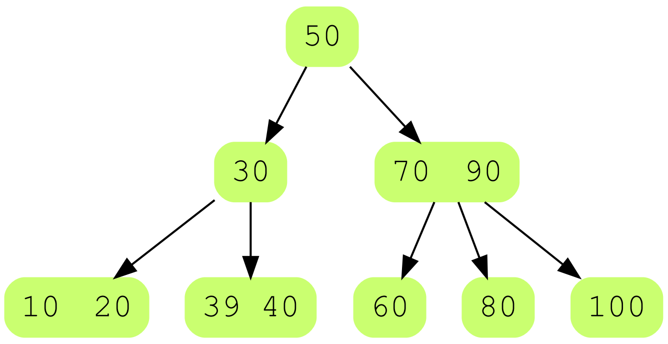
3 of 29

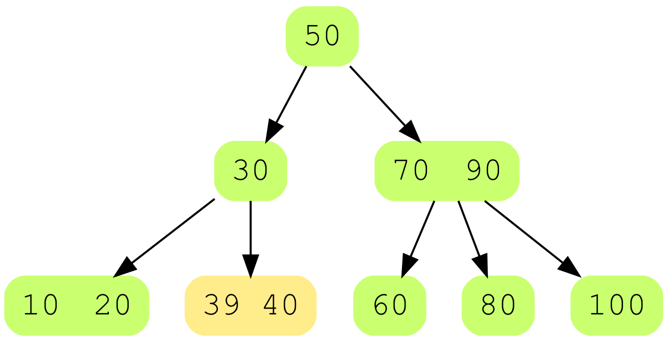


4 of 29

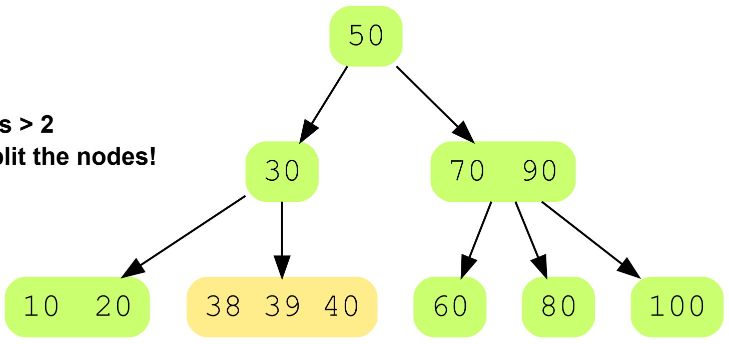


Insert 38!

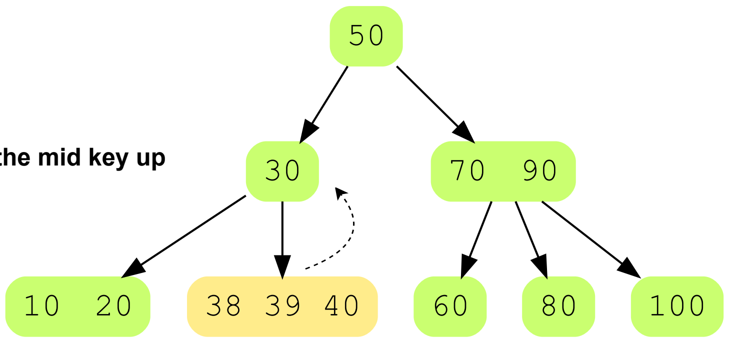


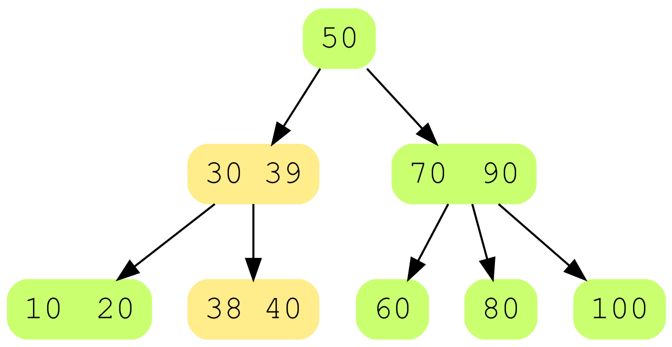


# Keys > 2
so, split the nodes!

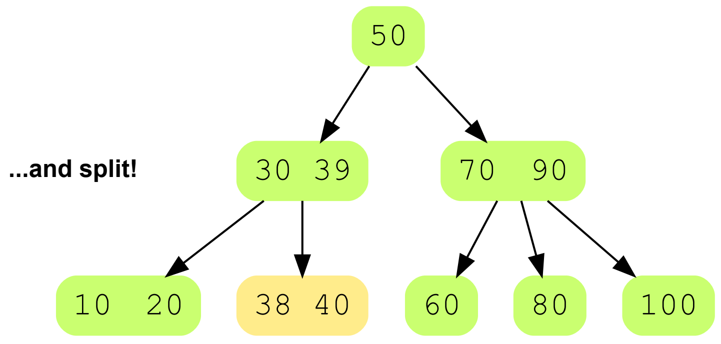


Shift the mid key up

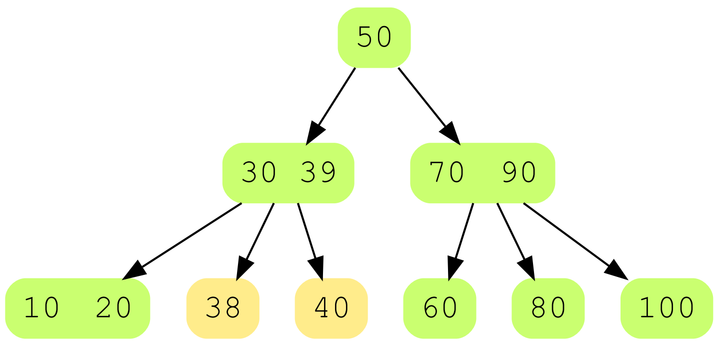




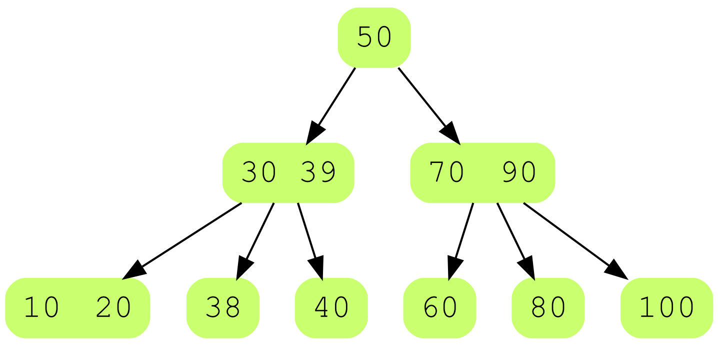
10 of 29



11 of 29

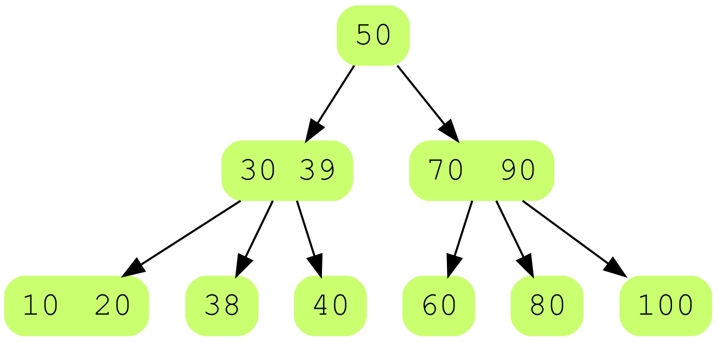


12 of 29

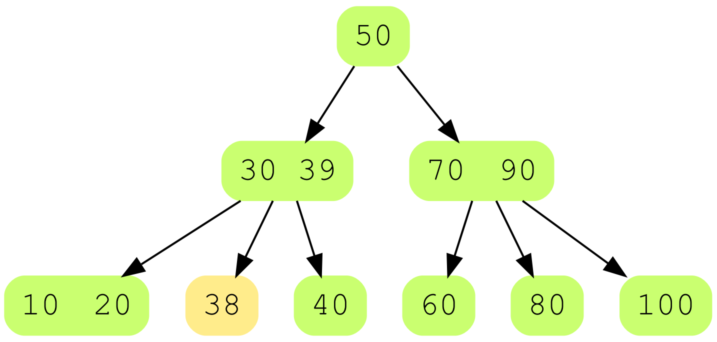


13 of 29

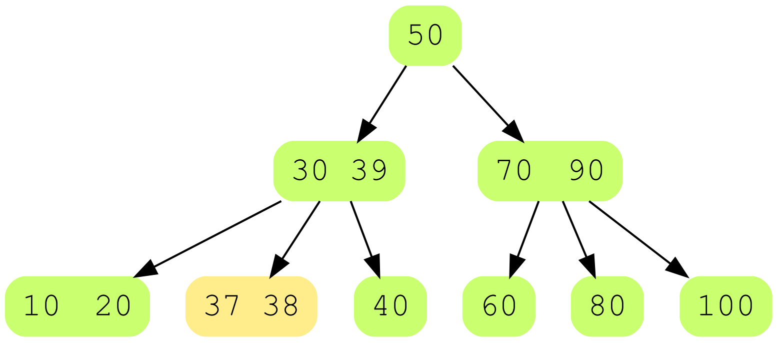
Insert 37!



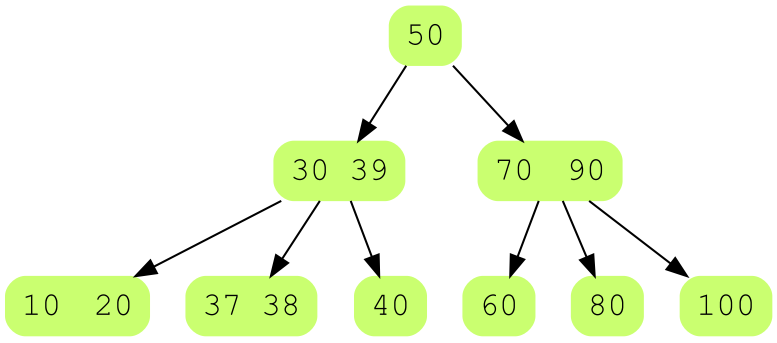
14 of 29



15 of 29

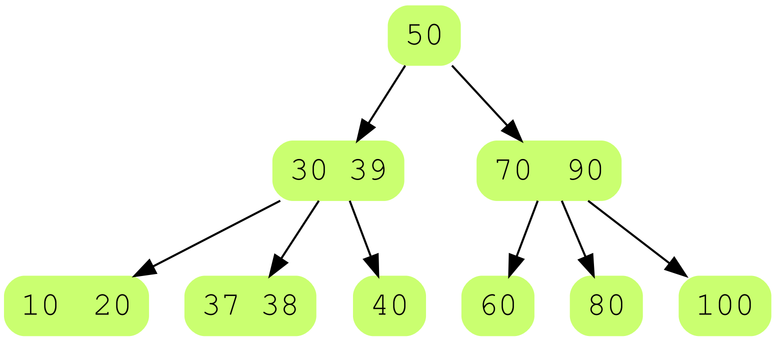


16 of 29

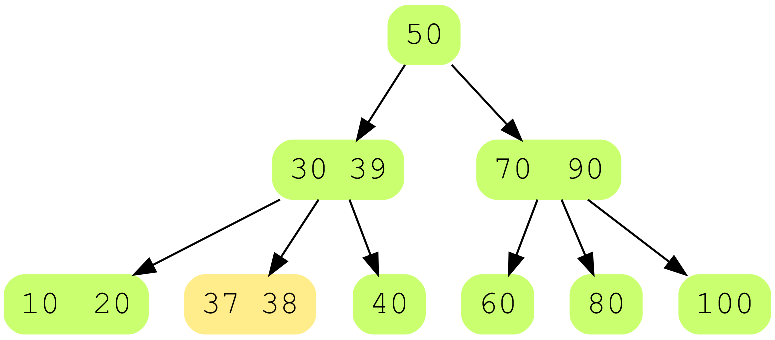


17 of 29

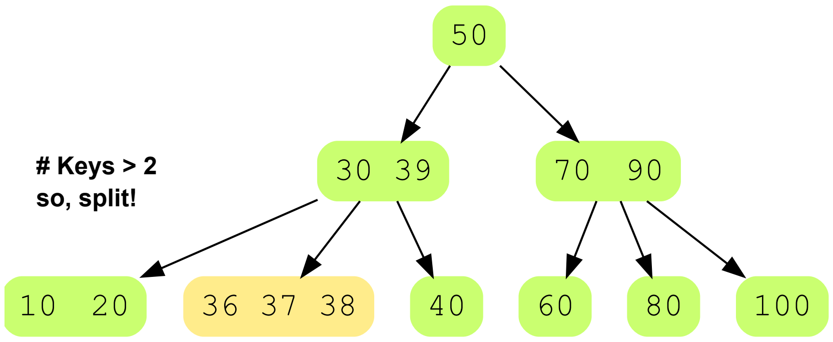
Insert 36!



18 of 29

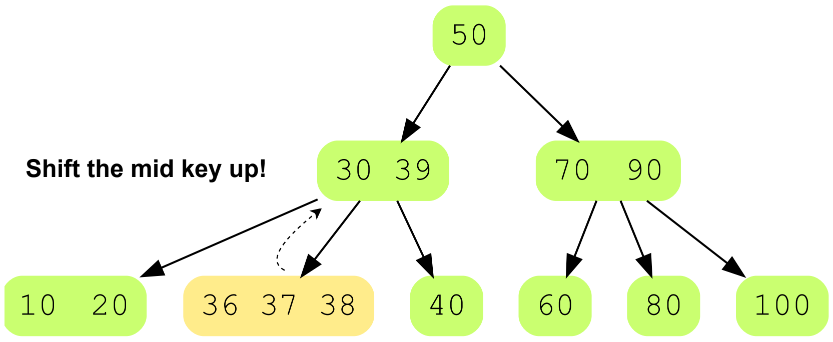


19 of 29



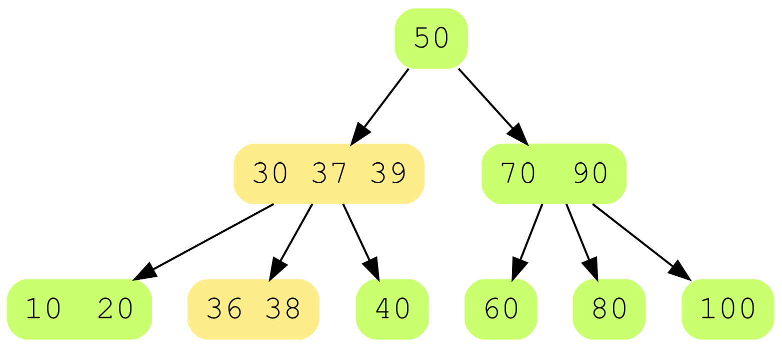
# Keys > 2
so, split!

20 of 29

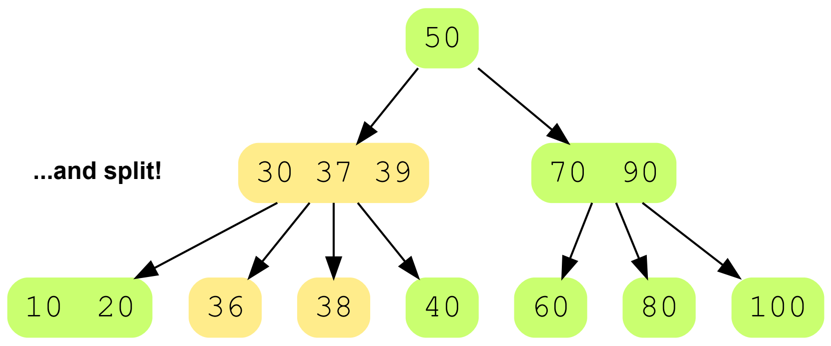


Shift the mid key up!

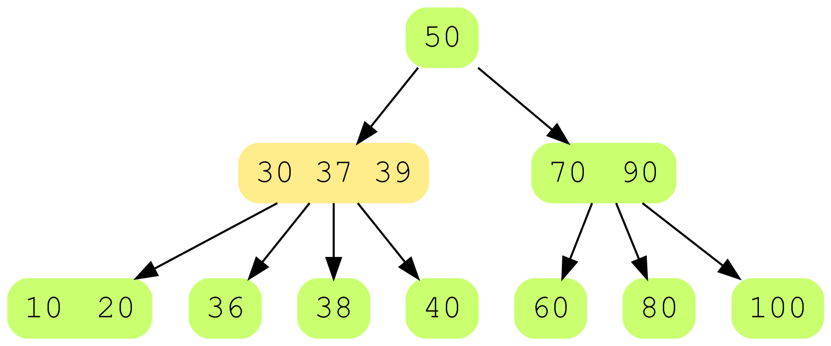
21 of 29



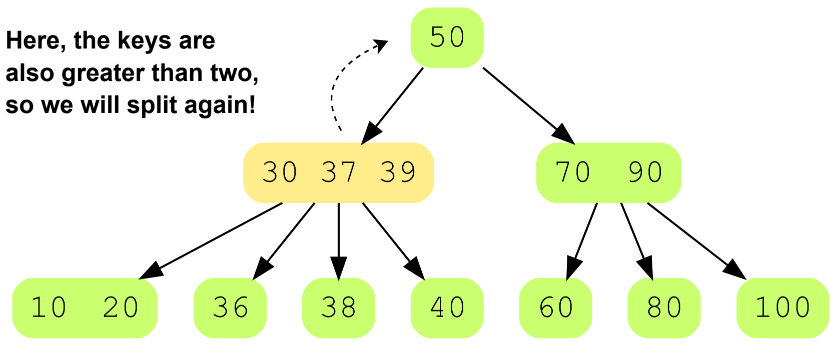
22 of 29



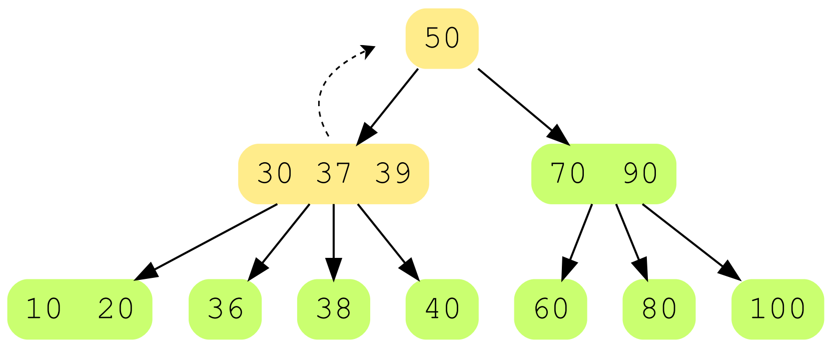
23 of 29



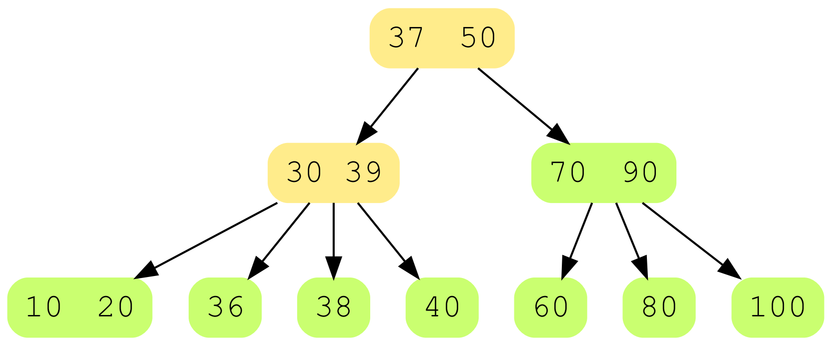
24 of 29



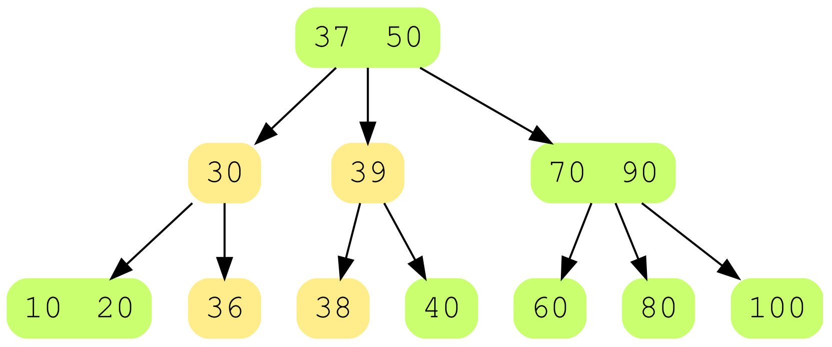
25 of 29



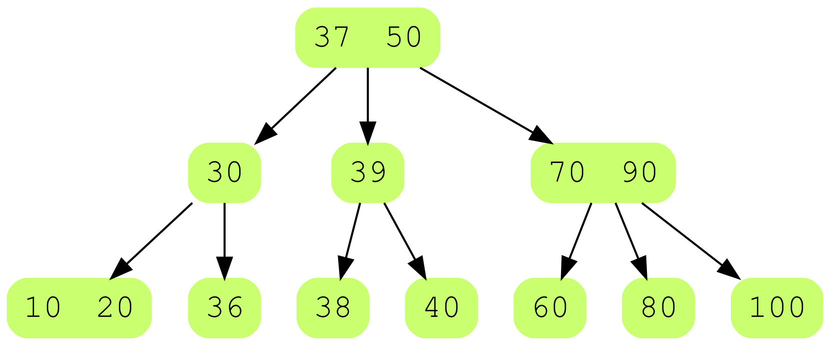
26 of 29



27 of 29



28 of 29



29 of 29

Explanation:

In this illustration, we have nodes on three levels. Initially, we insert 39 at the leaf node of 40. Then we insert 38 in the same node, but as the number of nodes exceeds from two, we will shift the middle element to the top, 39 will move to its parent node, i.e., 30. This is how we will keep inserting the elements till the end, you just need to make sure that all leaves come at the same height.

That's all! We are done with insertion. Now in the next chapter, we will see how elements are deleted from a 2-3 Tree.

Back
What is a 2-3 Tree?

Next
2-3 Deletion (Case #1)

✓ Mark as Completed

Report an Issue

Ask a Question
(https://discuss.educative.io/tag/2-3-insertion__introduction-to-trees__data-structures-for-coding-interviews-in-python)