Today – Lecture 14 – CS163

1) Topic #10 2-3-4, AVL and Red Black

2) Heaps & Graphs } <u>Next</u> Week

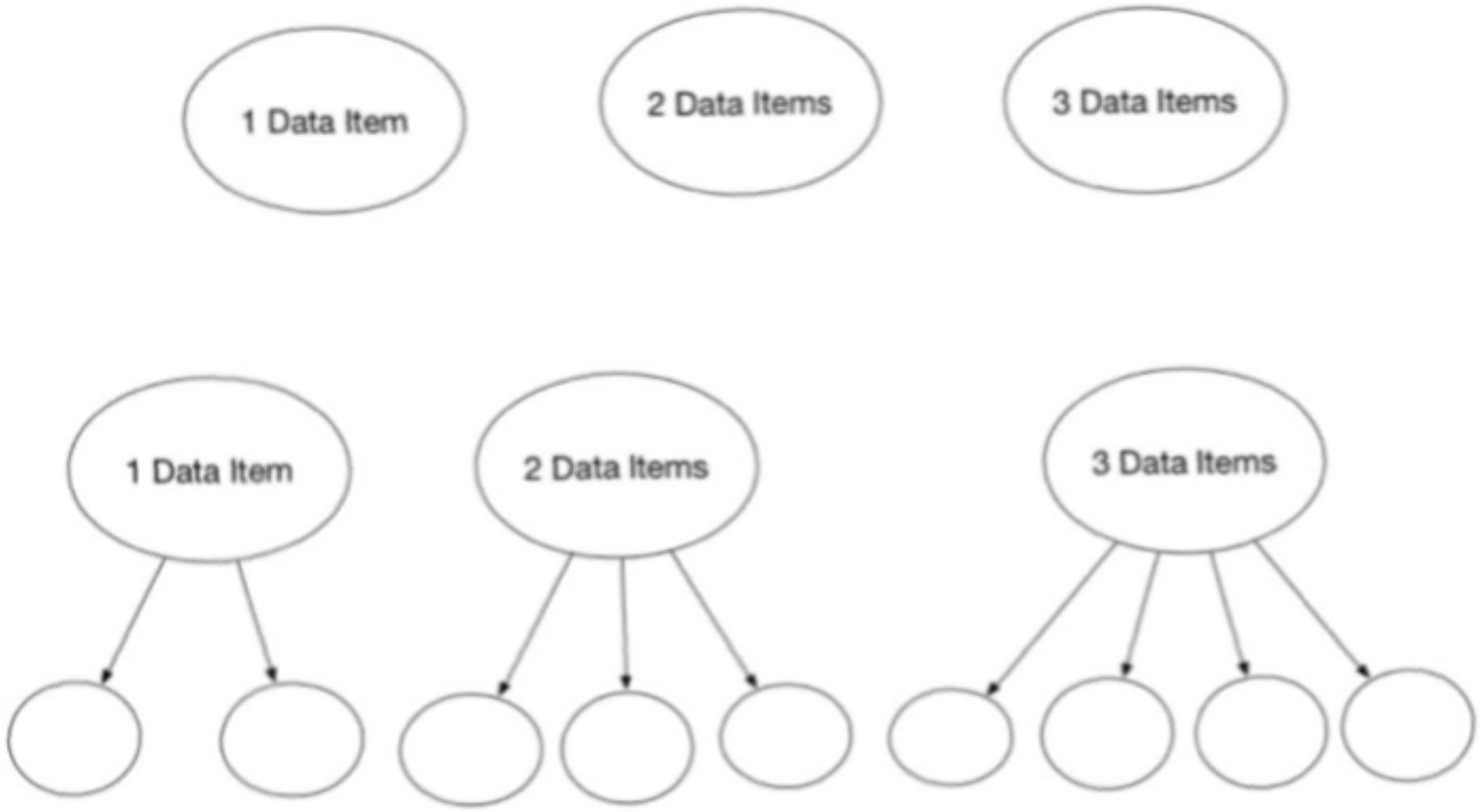
3) Efficiency & Sorting }

Announcements

* Final Exam Times have been posted

* Make sure to plan ahead to demo!

2-3-4 Tree



```
struct node
{
        data *items[3];  } 7 pointers per node
        node *child[4];
};
```

2-3-4 Insertion Algorithm

1) Travel to the appropriate leaf to add

2) As we traverse down the tree, ANYTIME a node with 3 pieces of data is encountered push UP the middle data item and "split" the node. Then, continue traversing.

3) There WILL ALWAYS be room in the leaf for the new item being added

4) Provides consistent run time performance at the cost of Memory overhead

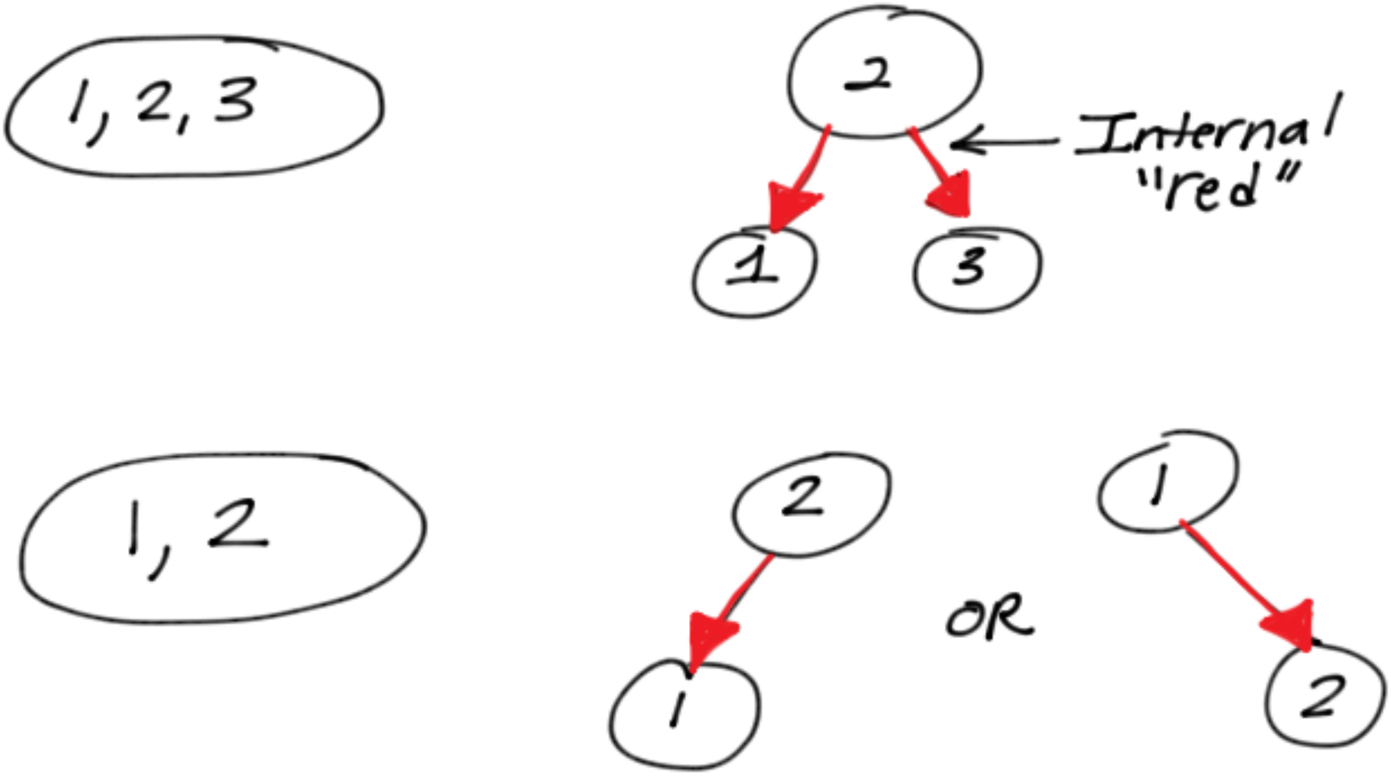
Build a 2-3-4 Tree

50 20 15 55 32 10 45 25 70 5

Red - Black Tree

1) BST with color flags

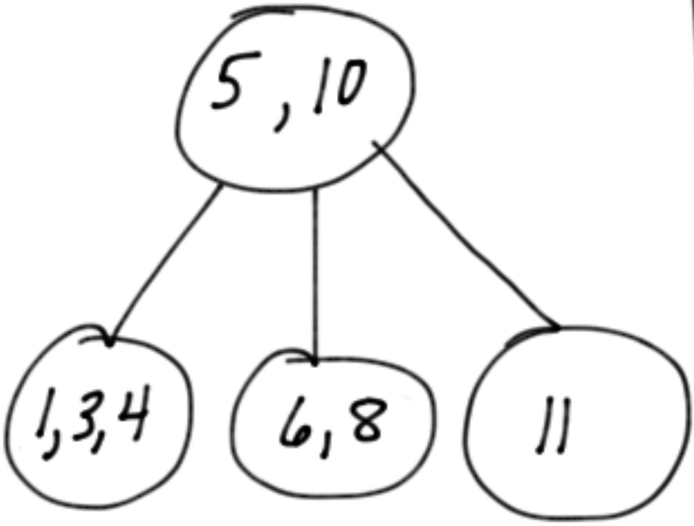
2) Use to simulate 2-3-4 tree



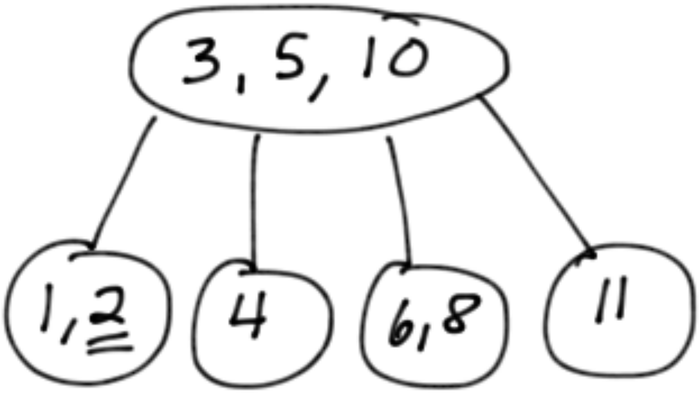
3) Nodes now 2 color flags which represent the type of relationship with each child

Red-Black Trees – "split the node"

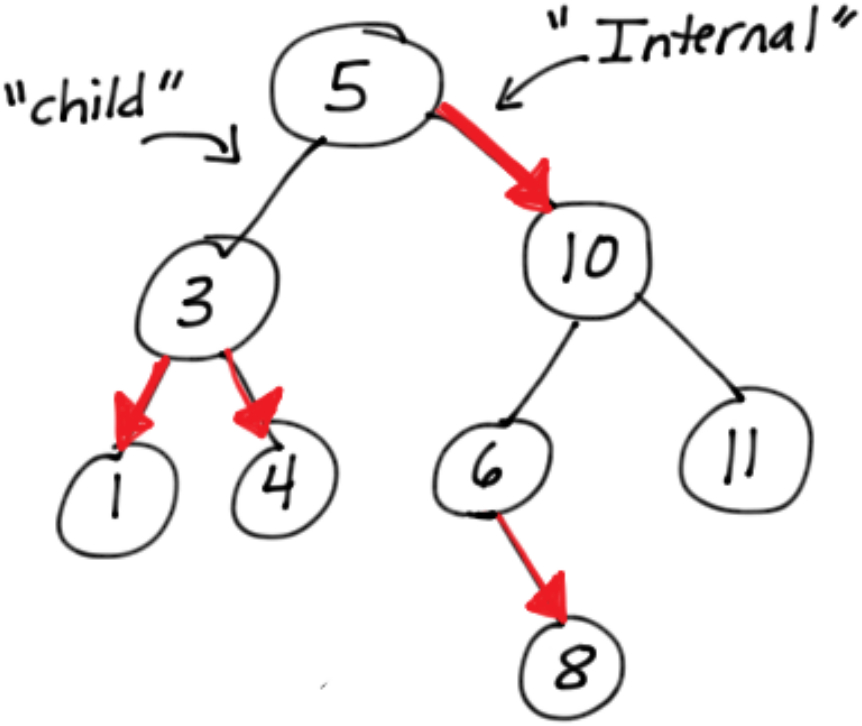
2-3-4 Tree

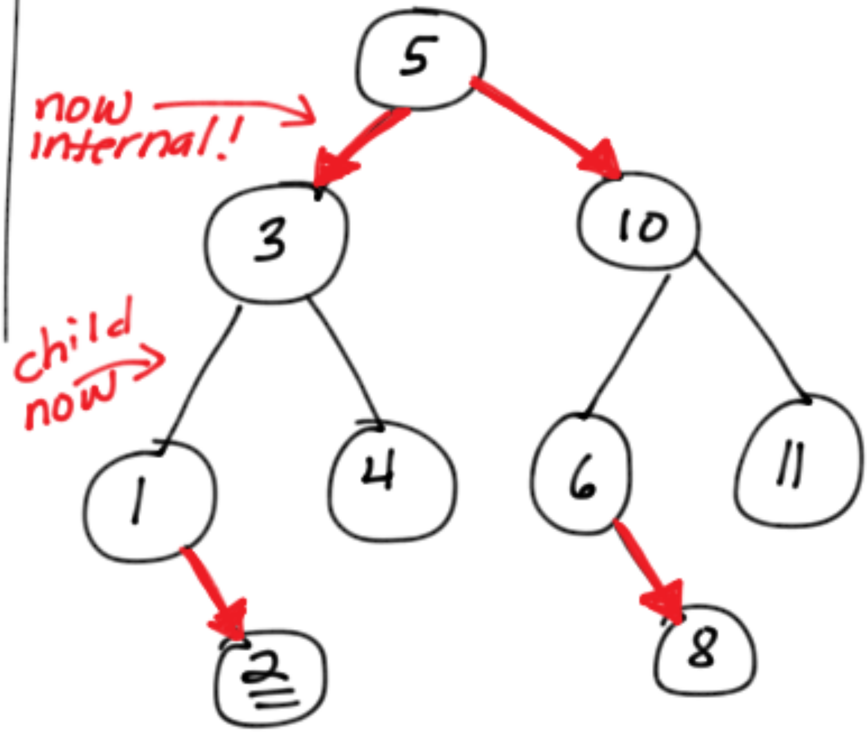


Add ② :



Red-Black Tree

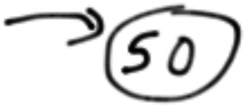




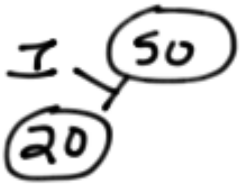
Creating a Red Black Tree

50 20 15 55 32 10 30 70

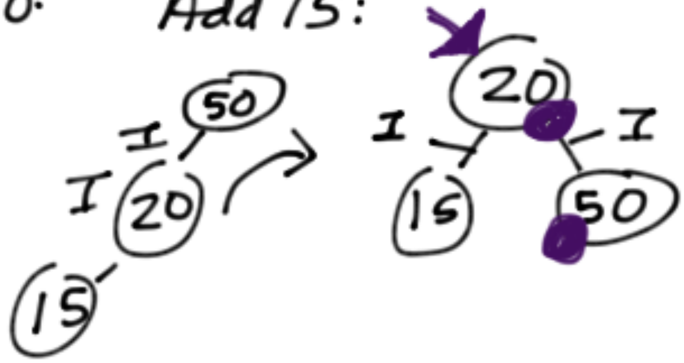
Add 50:



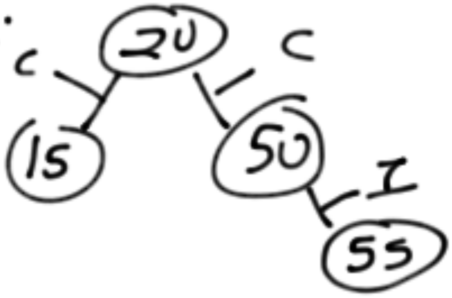
Add 20:



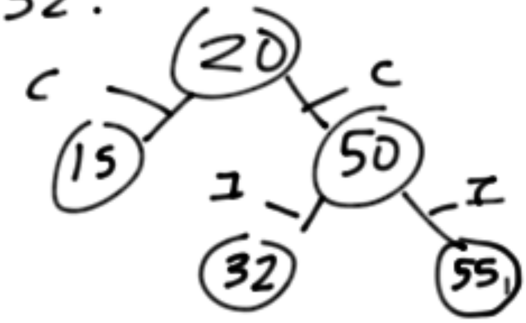
Add 15:



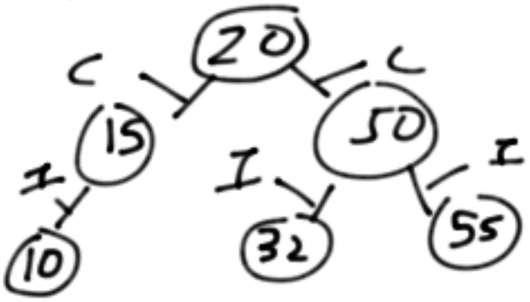
Add 55:



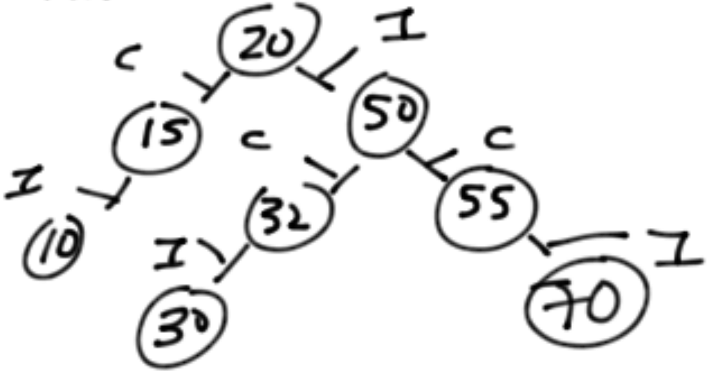
Add 32:



Add 10:



Add 30:



Add 70:

What happens if the data is sorted?

10 20 30 40 50 60

AVL - "Self" balancing BST

1) Named after two Soviet inventors: Adelson-Yelski and Landis

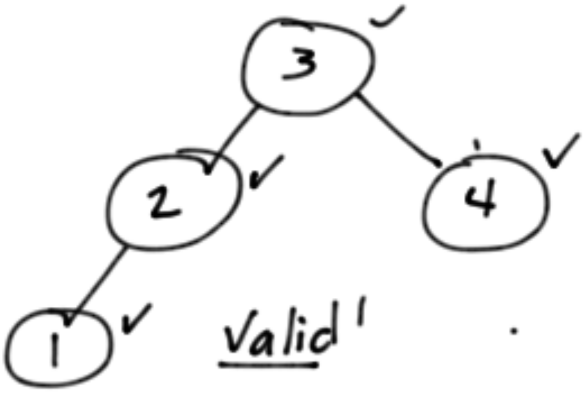
2) Uses a balance factor at each node

3) Provides fast retrieval

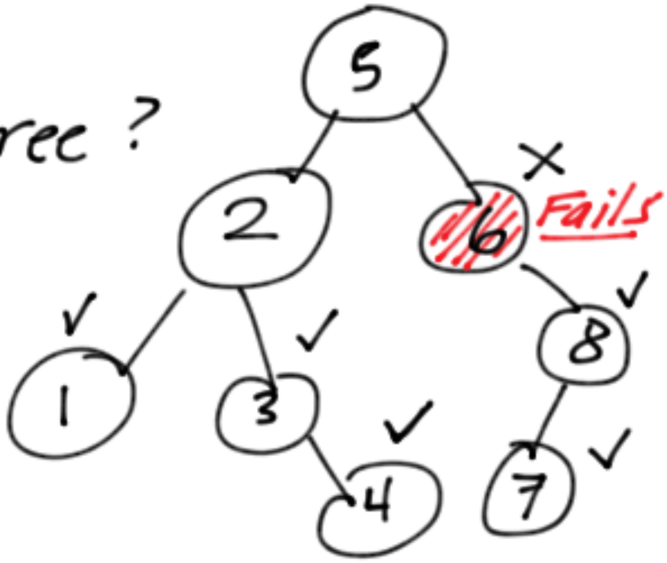
4) Red Black provides faster insertion & removal

5) when inserting & removing if the balance factor at any node is not 1, ϕ, -1 — then either a <u>single</u> rotation or a <u>double</u> rotation is performed.

6) Is this a valid AVL tree?



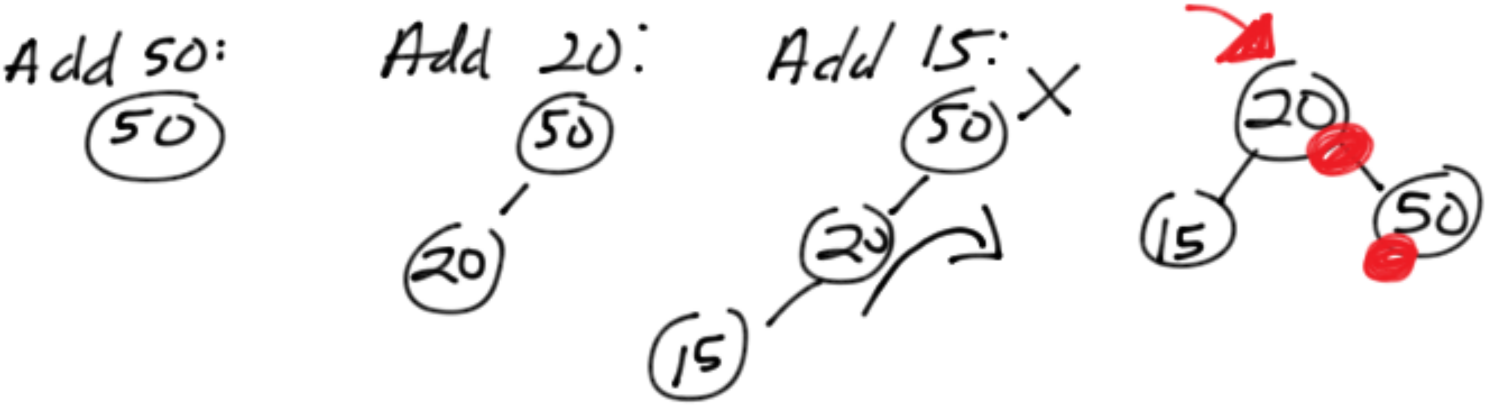
<u>Valid!</u>



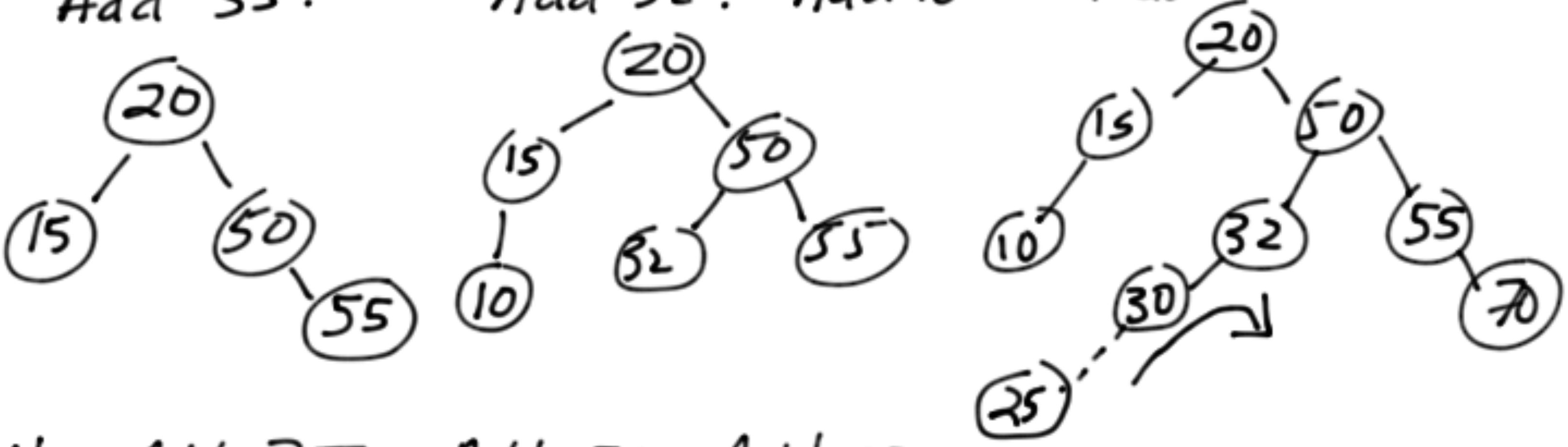
Build an AVL Tree

50 20 15 55 32 10 30 70

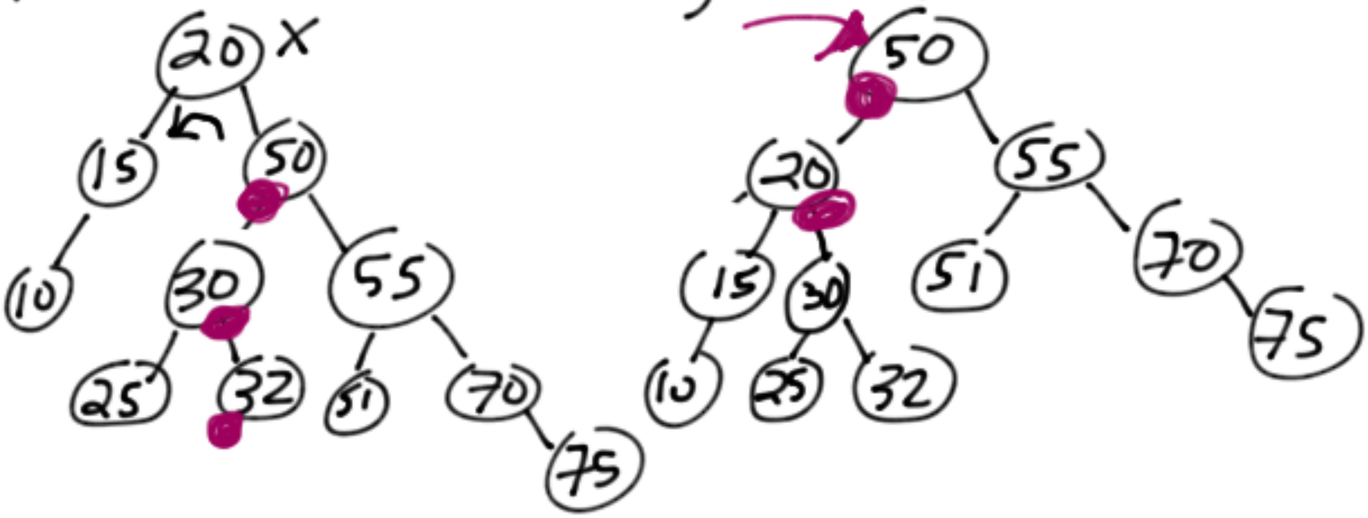
Add 50: Add 20: Add 15:



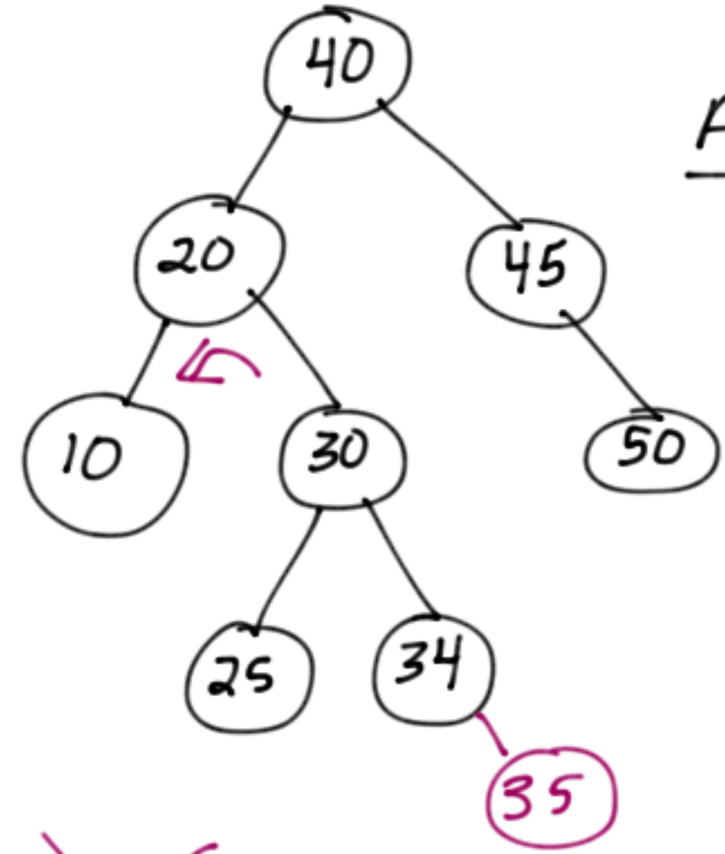
Add 55: Add 32: Add 10: Add 30: Add 70:



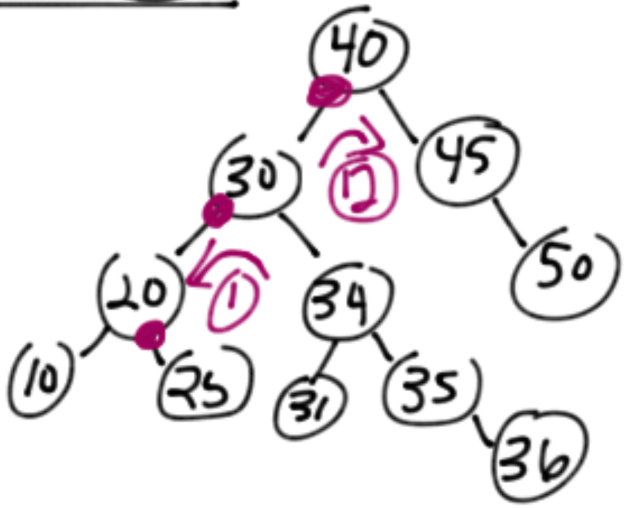
Now Add 25: Add 51; Add 75:

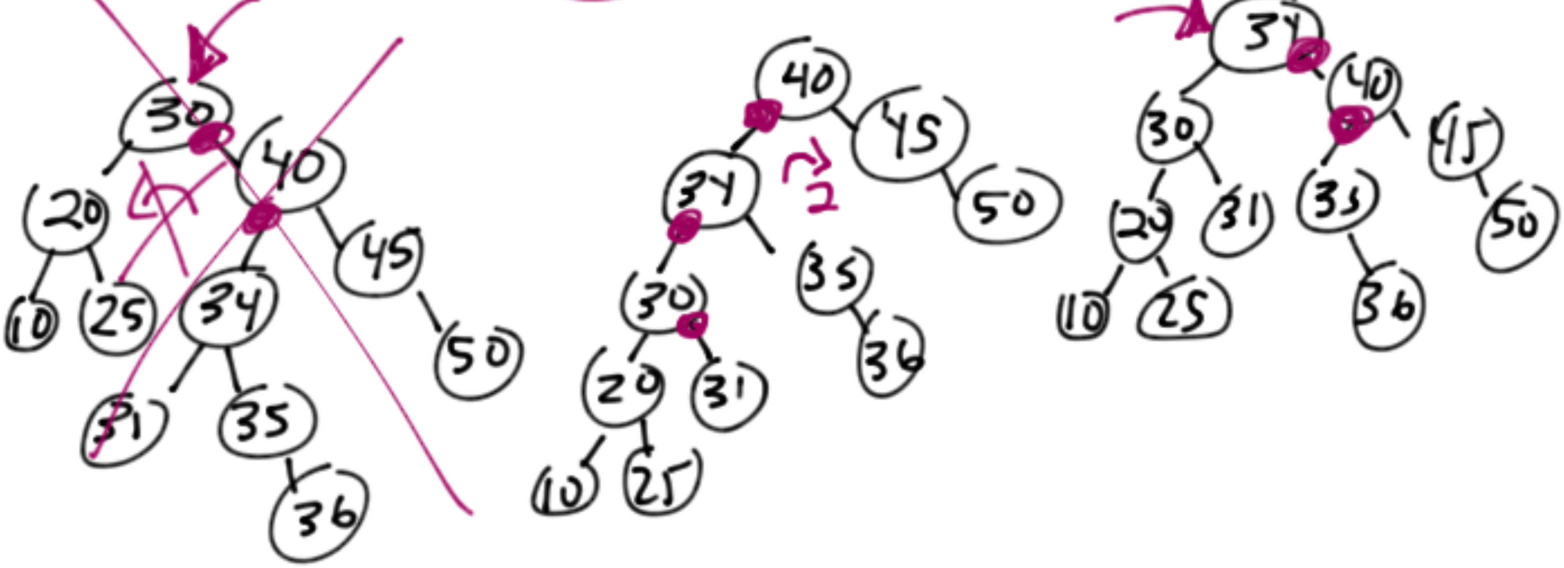


Experience a "double" rotation



Add 35 31 36


40
30
45
20
34
50
10
25
31
35
36
2
1


30
20
40
10
25
34
45
31
35
50
36
40
34
45
30
35
50
20
31
36
10
25
2
34
30
40
20
31
35
45
10
25
36
50