15CSE201 : Data Structures and Algorithms

Lecture 12 :Height Balanced Trees

Ritwik M

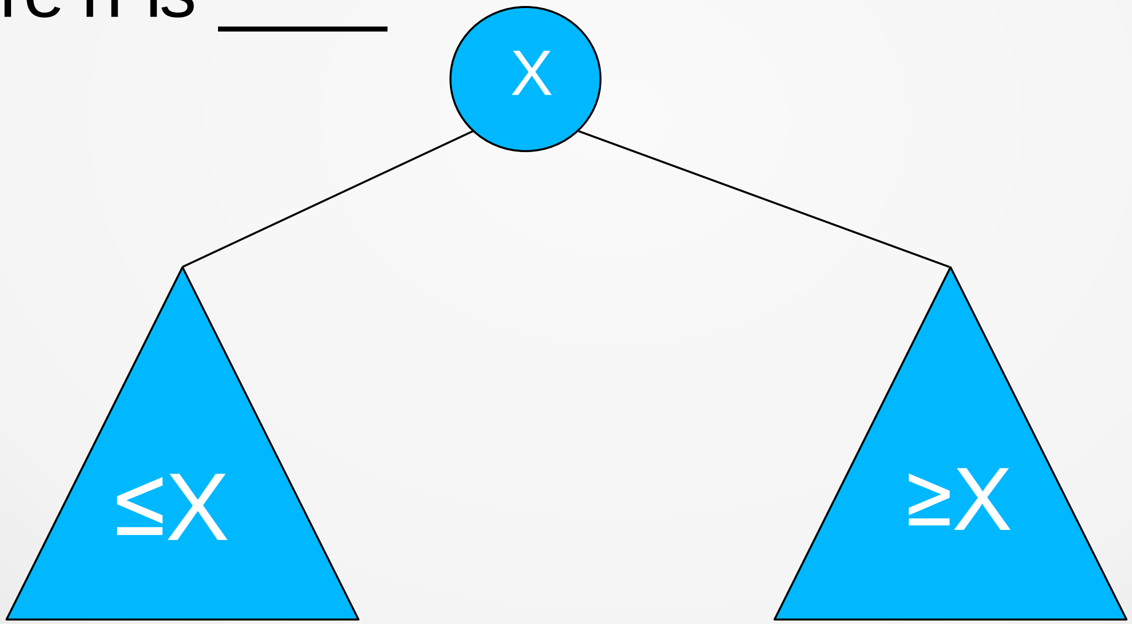
Based on the reference materials by Prof. Goodrich and Dr. Vidhya Balasubramanian

Contents

- Recap
- The importance of being balanced
- AVL trees
  - Definition and balance
  - Rotations
  - Insertion

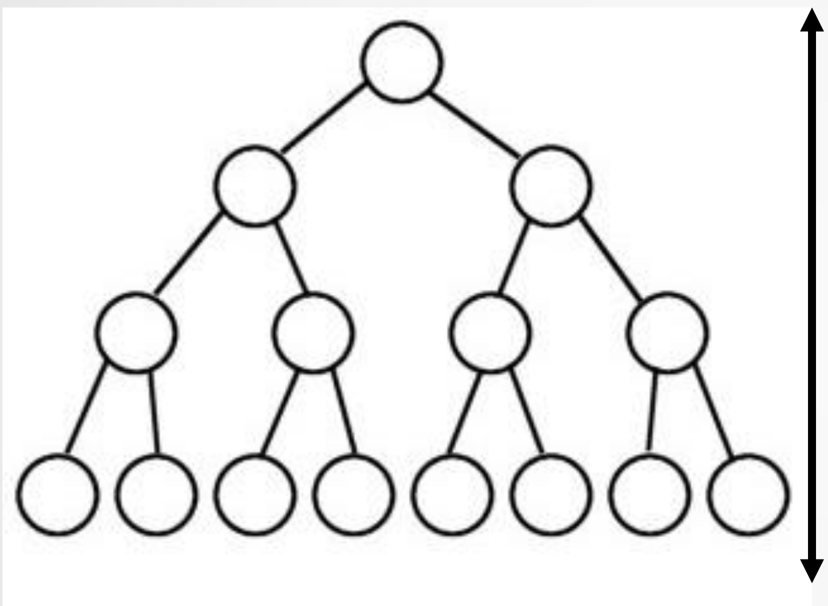
Recap: BST

- Rooted Binary Tree with additional properties
  - Support insert, delete, min-max, in O(h) time.
  - Where h is ____



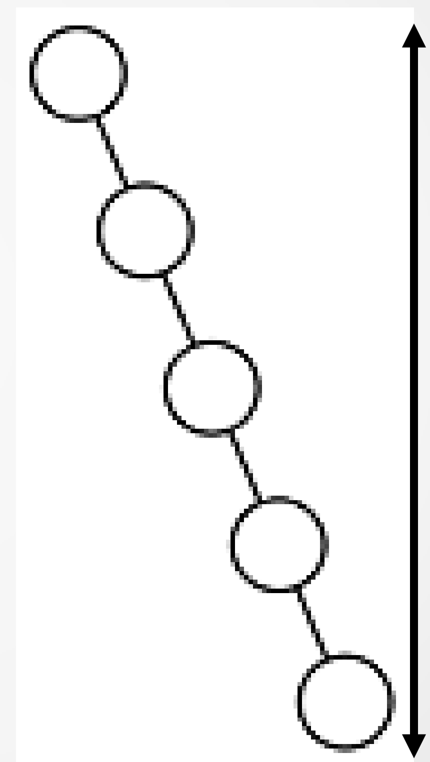
Balancing…

In a perfect world:



O(logn)

An Unbalanced Tree



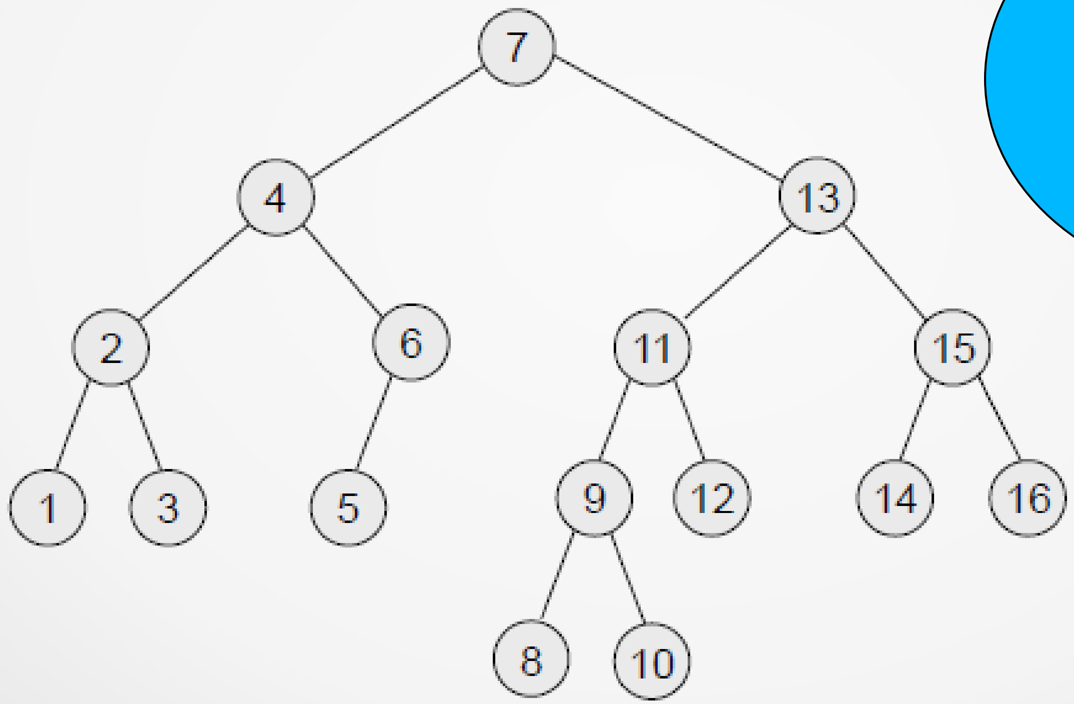
O(n)

So, what is the definition of Height?

Height Revisited

- What is the height of this tree?
- Height of nodes→7,11,4,16,9,6,2?

Can you create a formula for the height of a node?



Height = max(height of left child, height of right child)+1

Height Balanced Trees

- The height of a binary search tree depends on many factors
  - Order of Insertion of values
  - Impact of deletions

Why Balance?

- It is possible that a series of operations results in a tree with linear height
  - Worst case performance of search is linear
- Balance the height of the binary search trees so that the search cost is always O(logn)
  - <u>Height Balance Property</u>
    - For **<u>every internal</u>** node v of T, the height of the children of v differ by <u>at most 1</u>

AVL Trees

- Is a binary search tree that satisfies the height-balance property
  - Self balancing search tree
  - The subtree of an AVL tree is itself an AVL tree
- Named after its inventors
  - Adel'son- Vel'skii, and Landis
- The height-balance property has also the important consequence of keeping the height small

AVL Trees Cont.

- The height of an AVL tree storing n items is O(log n)
- Searching:
  - As in an ordinary binary search tree
  - Cost : O(log n)

Which of the following is not an AVL Tree?

Fig 1:

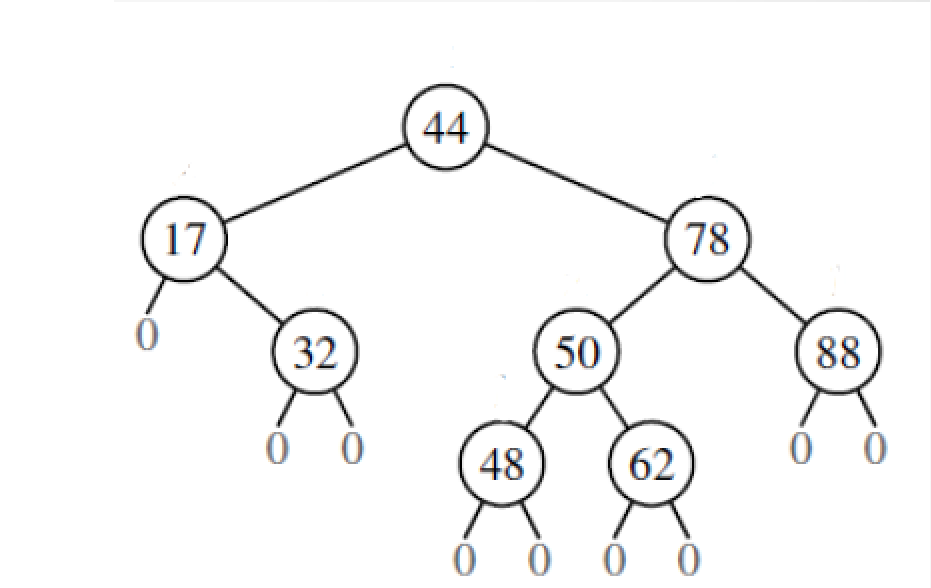
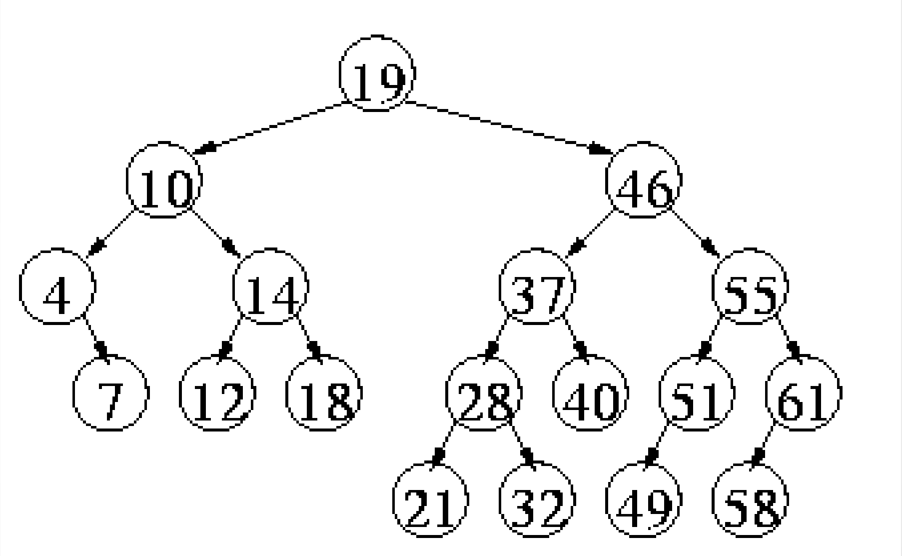
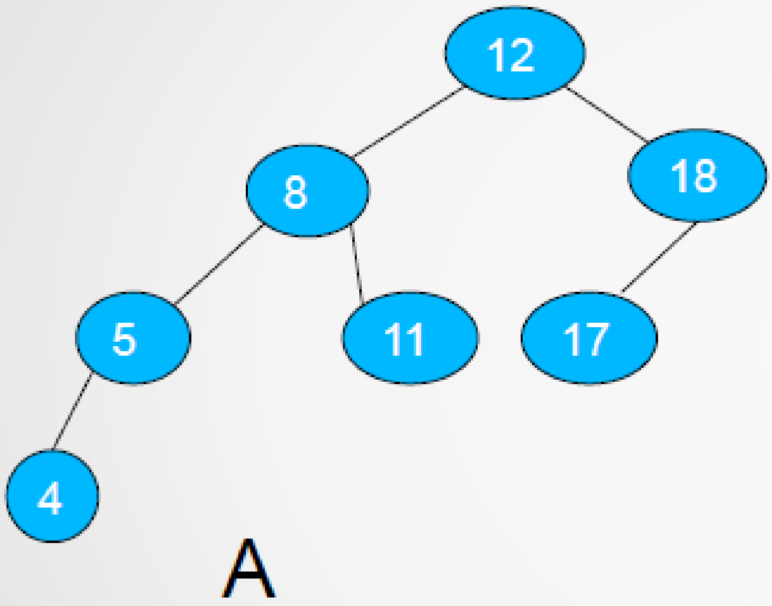
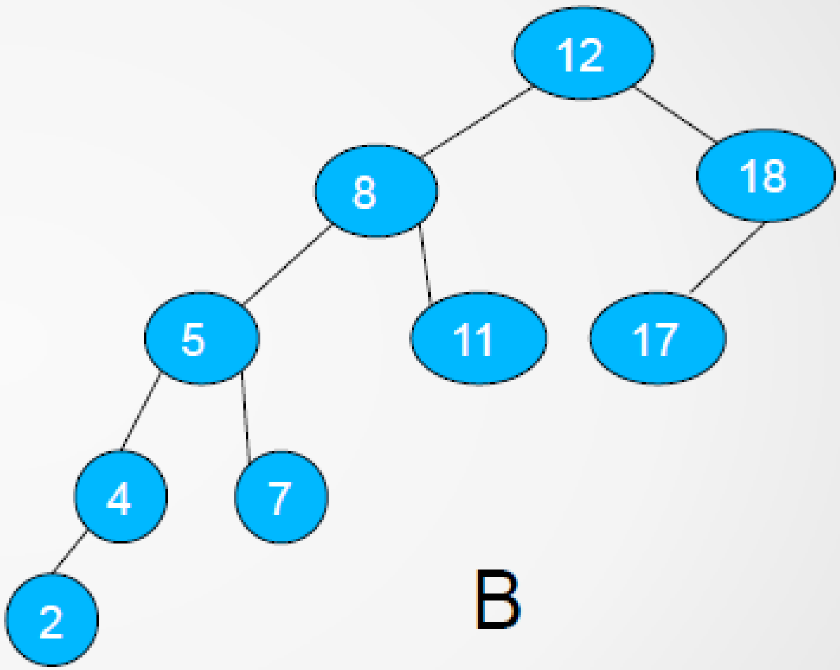


Fig 2:



More Examples





Which is the AVL tree? A or B?

Properties

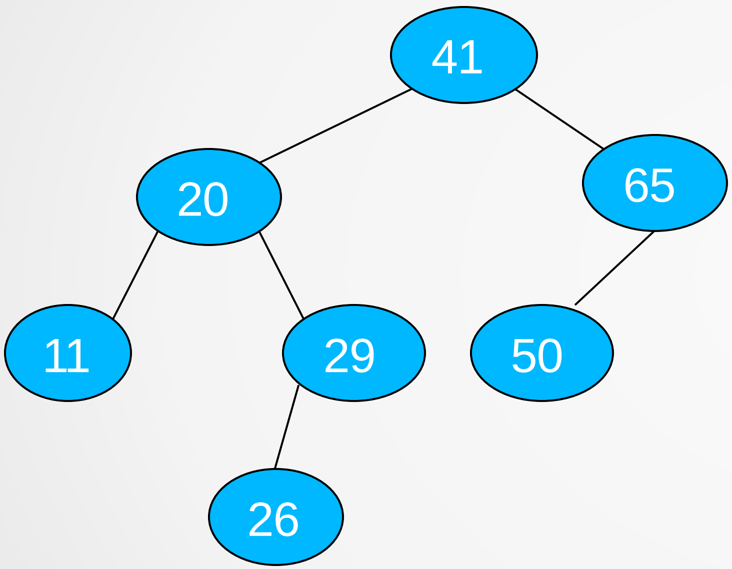
- The depth of a typical node in an AVL tree is very close to the optimal log N.
- Consequently, all searching operations in an AVL tree have logarithmic worst-case bounds.
- An update (insert or remove) in an AVL tree could destroy the balance. It must then be rebalanced before the operation can be considered complete.
- After an insertion, only nodes that are on the path from the insertion point to the root can have their balances altered.

AVL Tree Insertion

1. Simple BST insert
2. Fix AVL property(height balance property)
   - Rotations

Example

Insert 23

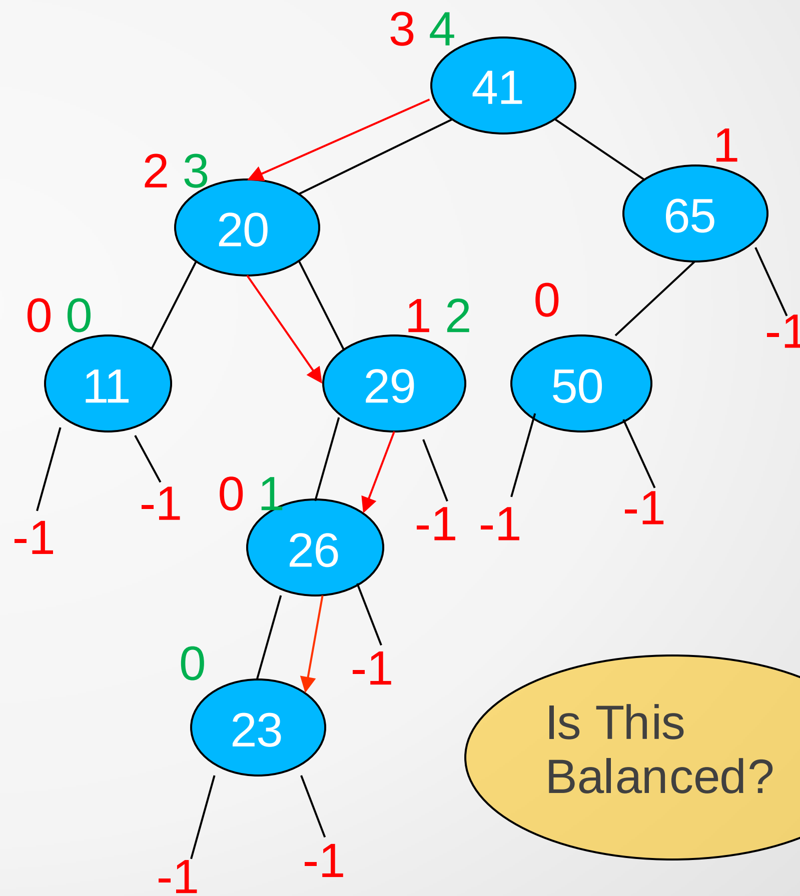


Path of new insert

Original Height

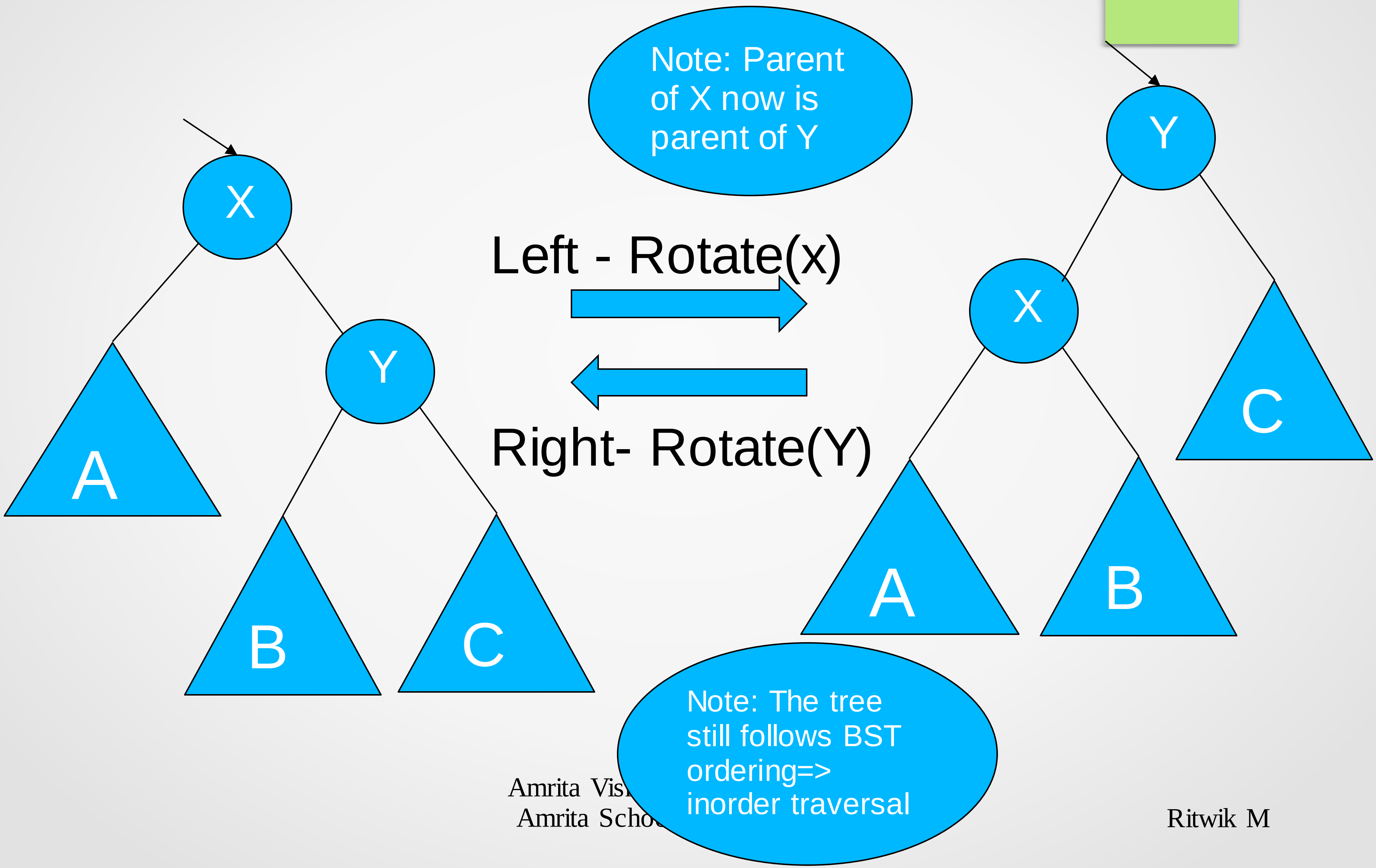
New Height

Step 1: As in BST



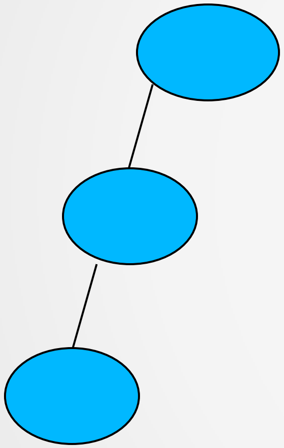
Is This Balanced?

Rotations

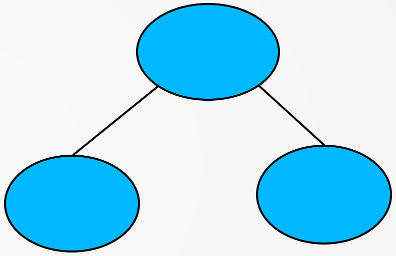

Note: Parent of X now is parent of Y
X
Y
A
B
C
Left - Rotate(x)
Right- Rotate(Y)
Y
X
C
A
B
Note: The tree still follows BST ordering=> inorder traversal

Example Cont.

In the example the nodes looked like:



But we would like it as:

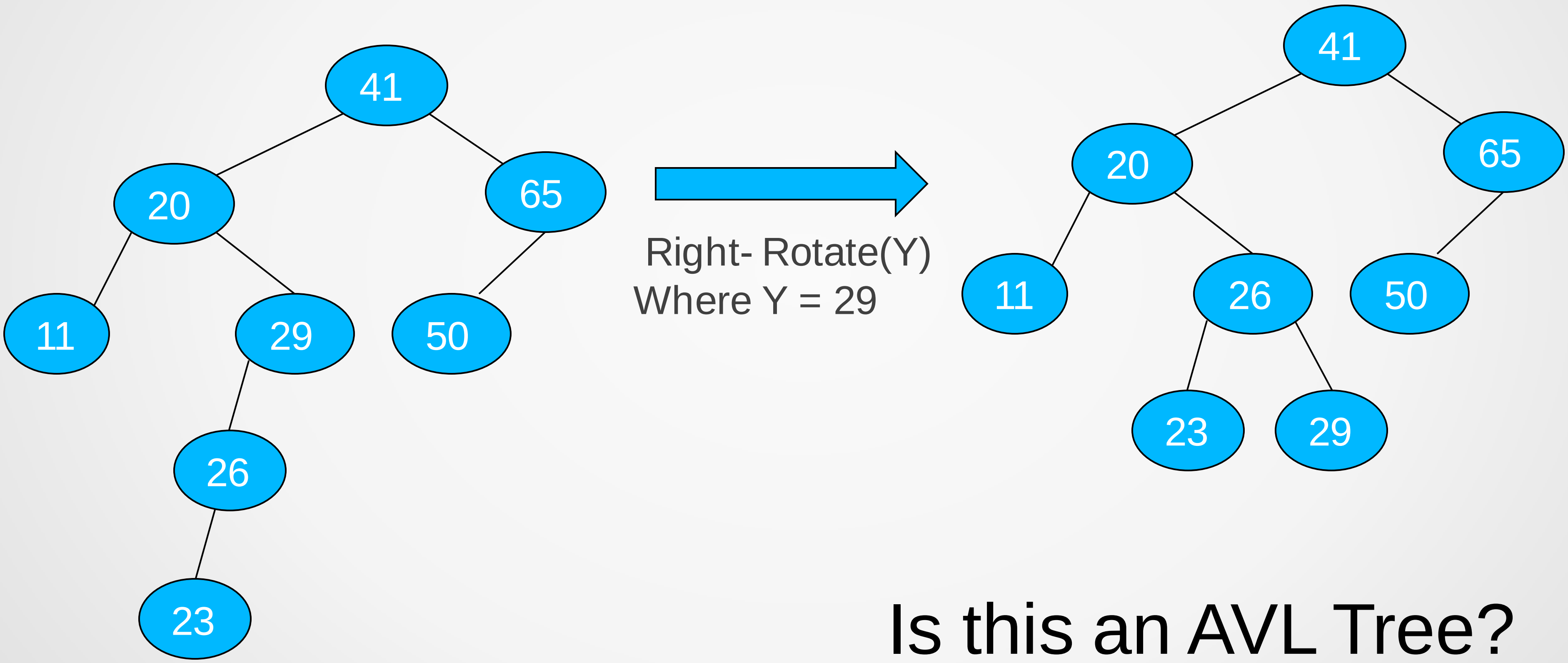


This is a Right - Rotate(29)

Example Cont.

Original

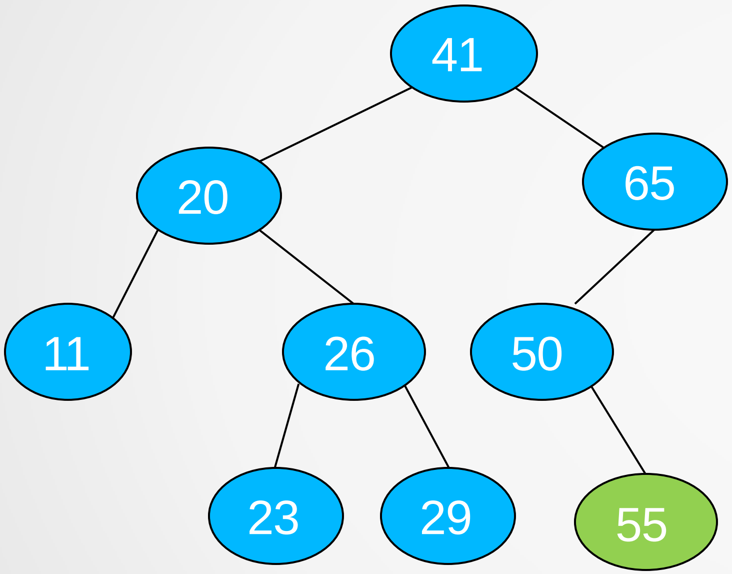
Post Rotation



Is this an AVL Tree?

Example Cont.

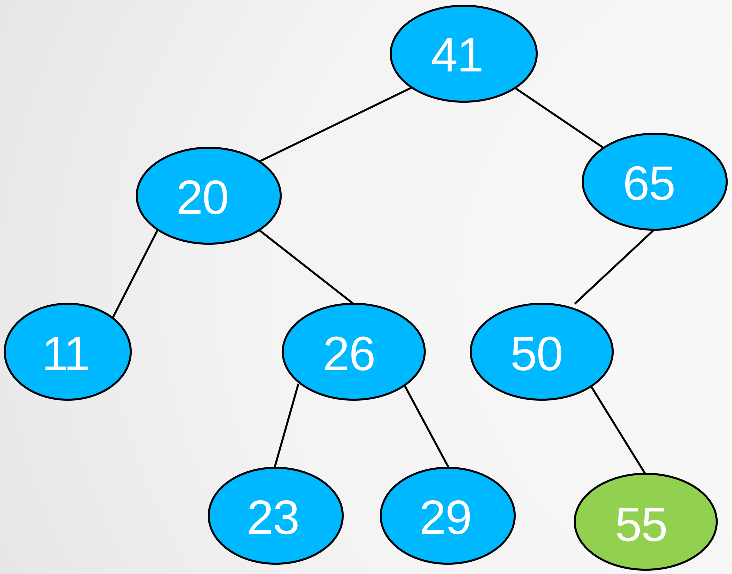
Now insert 55



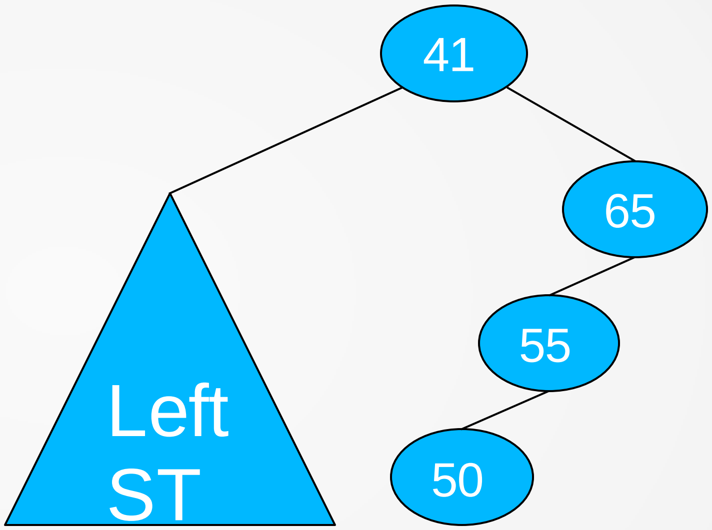
Does this satisfy the height order property?

Example Cont.

Original



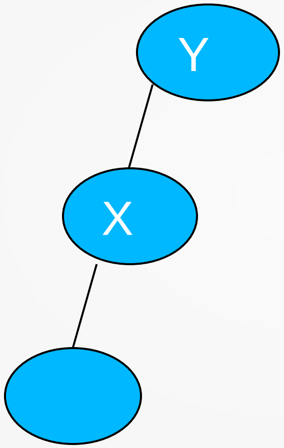
Left - Rotate(50)



Does this solve the issue? Is it height balanced now?

Example Cont.

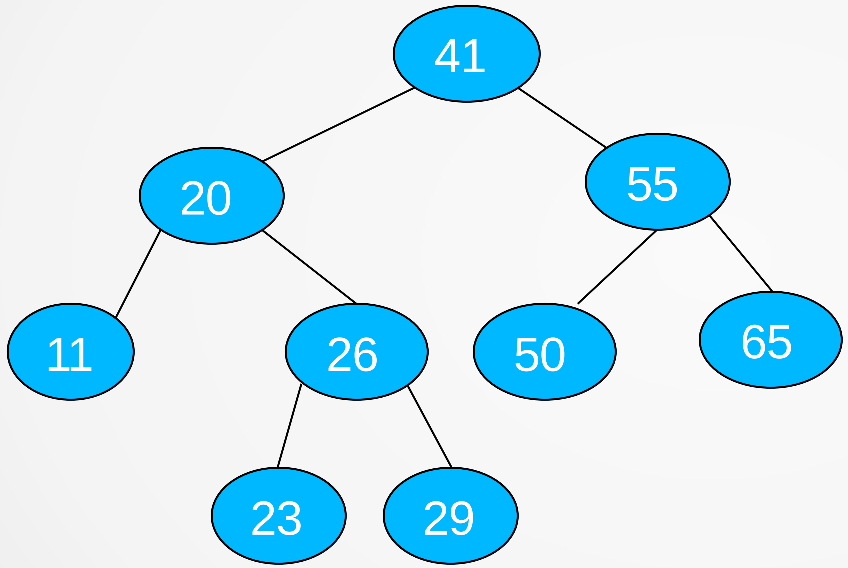
No. But it looks similar to a case already solved.



Our solution was Right – Rotation (Y)

Example Cont.

Right – Rotate(65)



Is this finally an AVL Tree?

Observations

- To ensure that a BST becomes an AVL tree – Rotations are all that we require
- Any BST can be an AVL tree with either one or 2 rotations.
- Using 2 rotations is also called as double rotation
- The example solved only the base cases – sometimes rotations can upset the higher nodes in the tree

Deletion

- Delete element as in the case of the BST
- Restore height balance property

ADT operations

- Insert(x)
- Delete(x)
- findMin()
- nextLarger(x) & nextSmaller(x) or successor(x) and predecessor(x)

Analysis

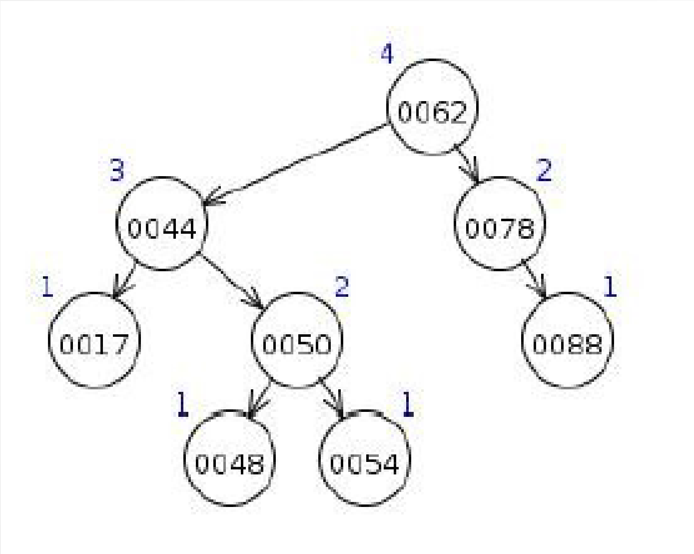
- Single restructure : O(1)
  - using a linked-structure binary tree
- Finding an element: O(log n)
  - height of tree is O(log n)
- Insertion: O(log n)
  - initial find: O(log n)
  - Restructuring up the tree, maintaining heights is O(log n)
- Removal: O(log n)
  - initial find: O(log n)
  - restructuring up the tree, maintaining heights is O(log n)

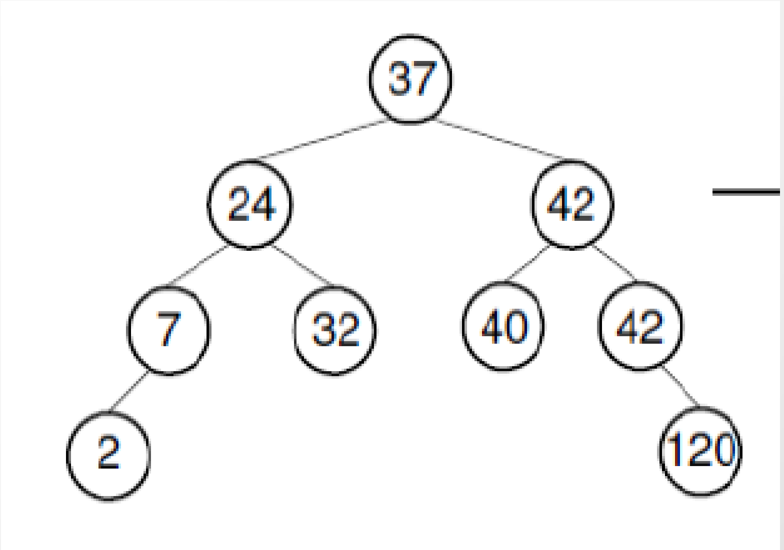
Applications Of AVL Trees

- Sorting
  - Insert n items – O(n log n)
  - In-order traversal – O(n)
- Balanced BST
- Priority Queue
  - Heaps are best as they are in-place but AVL's can also be used for this as a sub-optimal case

Practice

- Draw the AVL tree resulting from the insertion of an item with key 52, 95, 65 in the tree in the left given below
- Show the result (including appropriate rotations) of inserting the following values into the tree on the right
  - 39, 300,50,1





Amrita Vishwa Vidhyapeetham
Amrita School of Engineering

Ritwik M

Exercise

- Consider the following sequence of keys
  - 5,16,22,45,2,10,18,30,50,12, 1
  - Create an AVL tree by inserting one element at a time in order
  - What happens when you delete 16, 30 from the tree