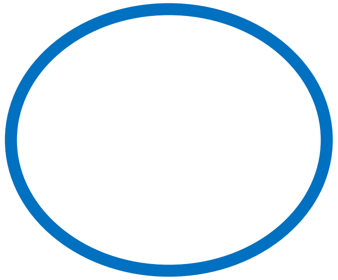
Data Structures & Algorithms

TREES

Trees



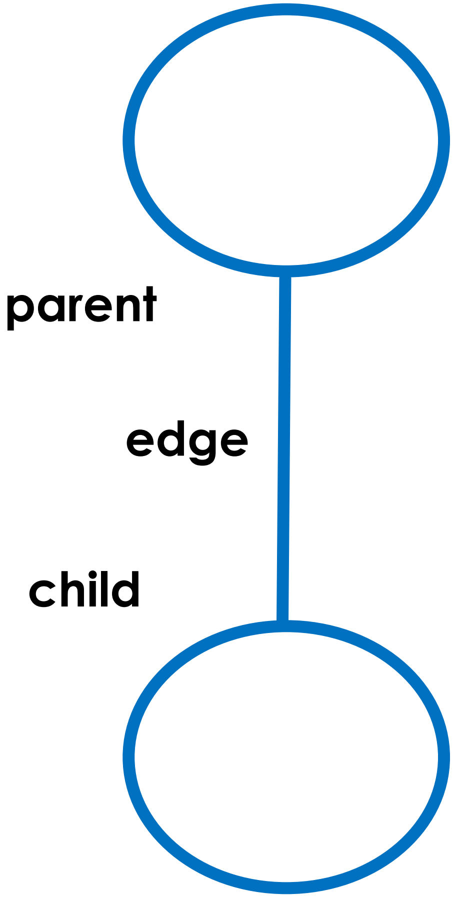
Root
Leaf

Node

Tree with one node

Height: 0

Trees



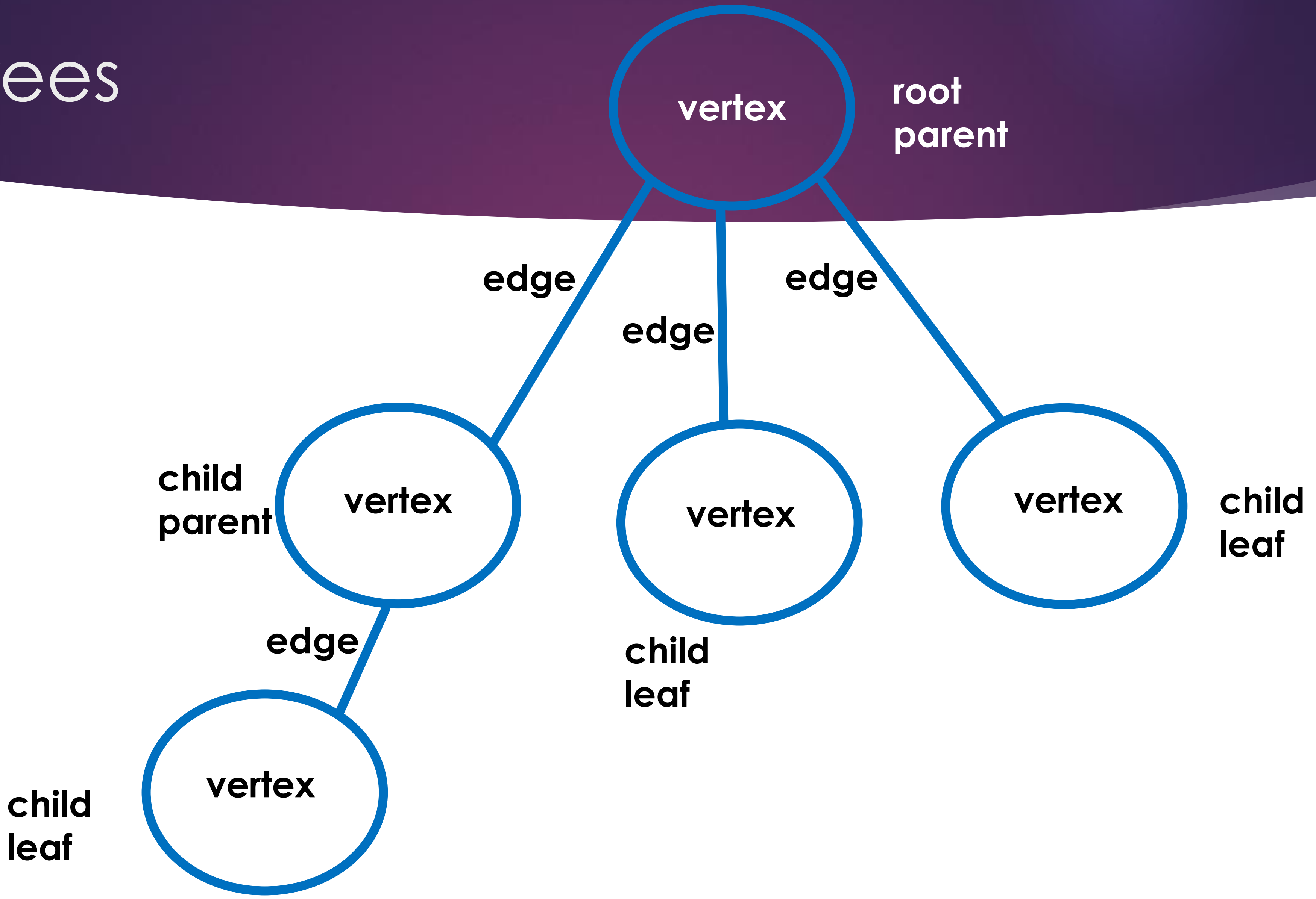
Root

Tree with two nodes

Height: 1

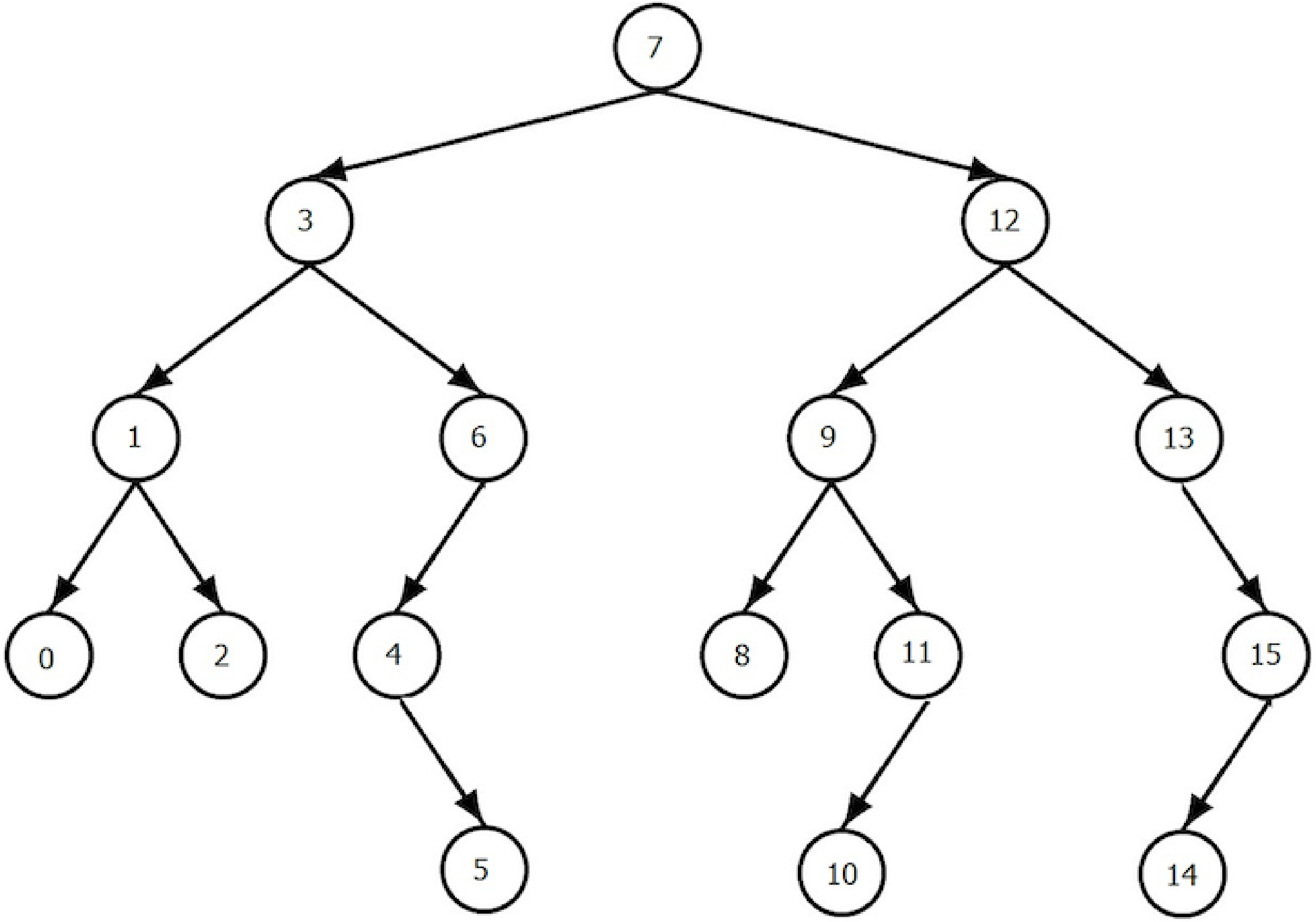
Leaf

Trees

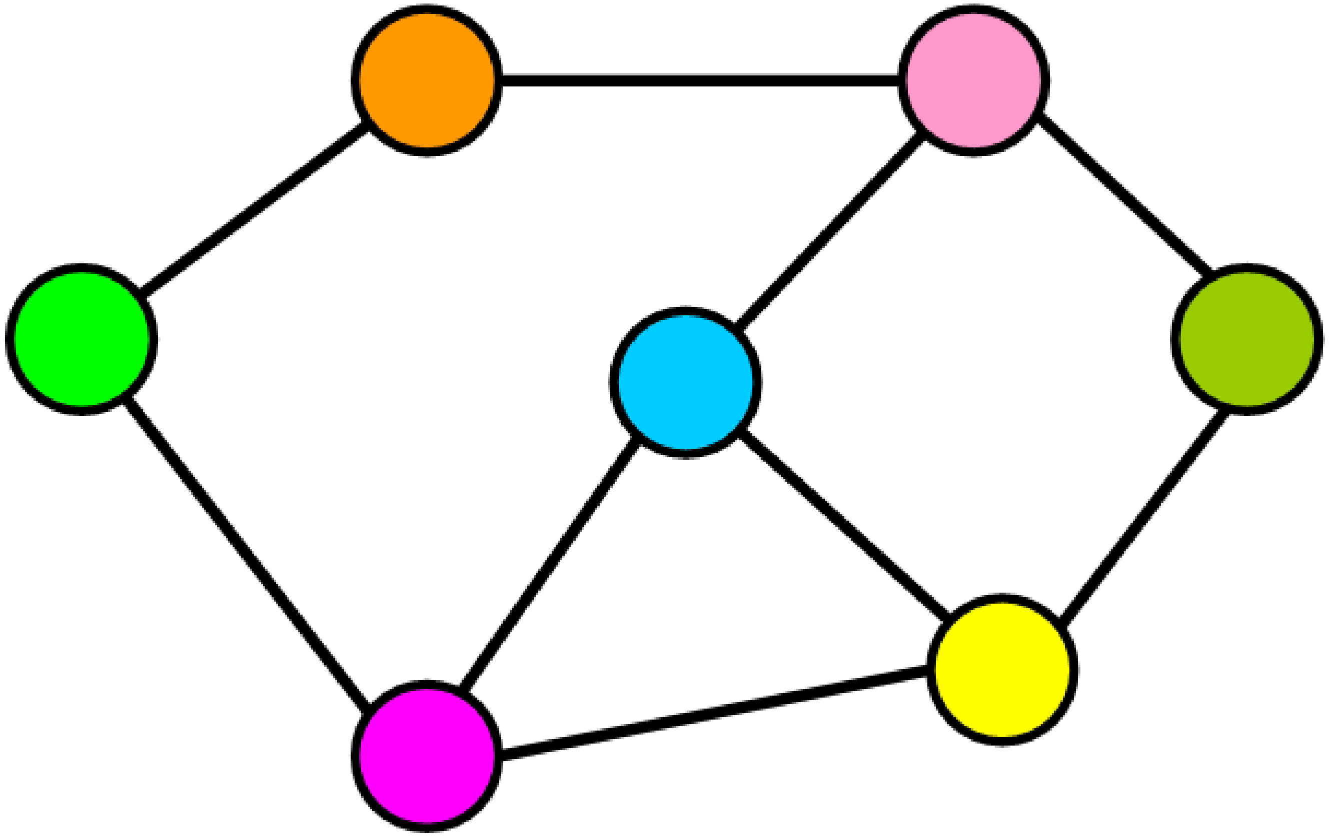


vertex
root
parent
edge
edge
edge
child
parent
vertex
vertex
vertex
child
leaf
edge
child
leaf
child
leaf
vertex

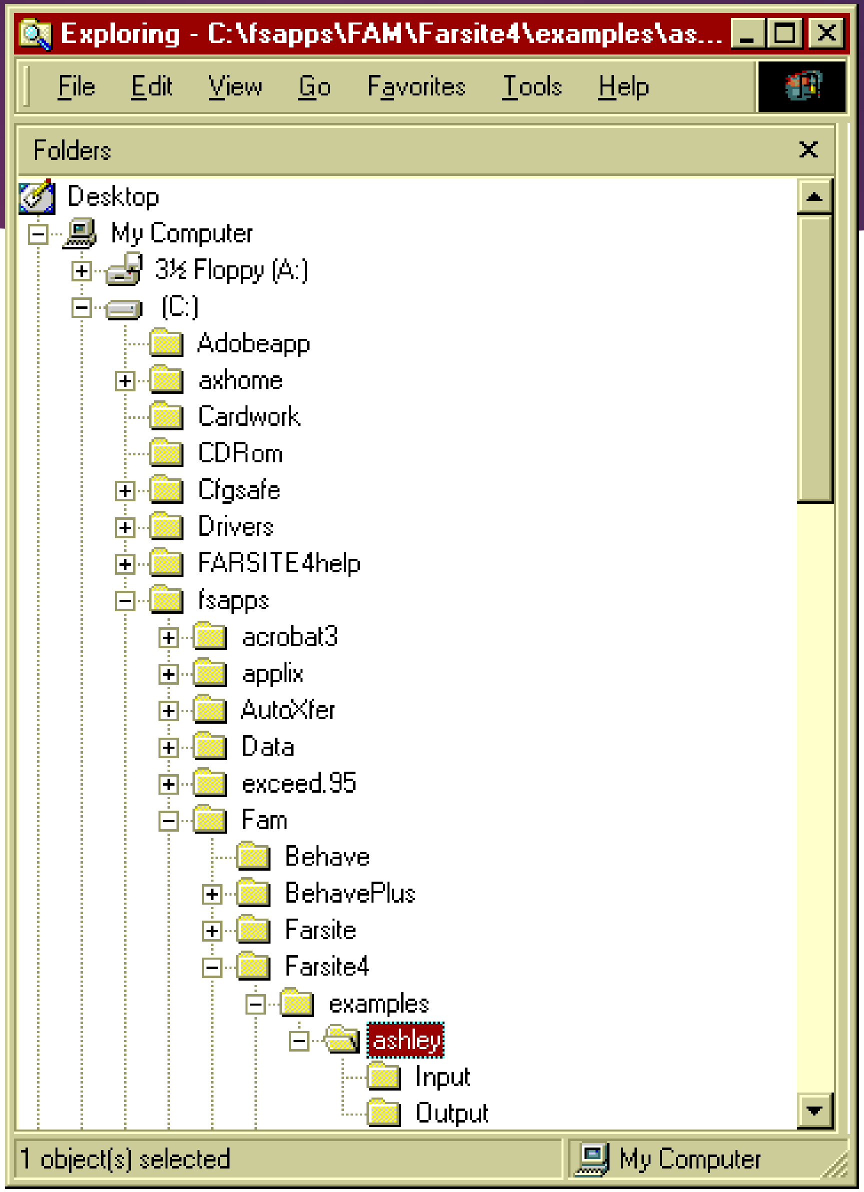
All Trees are Graphs



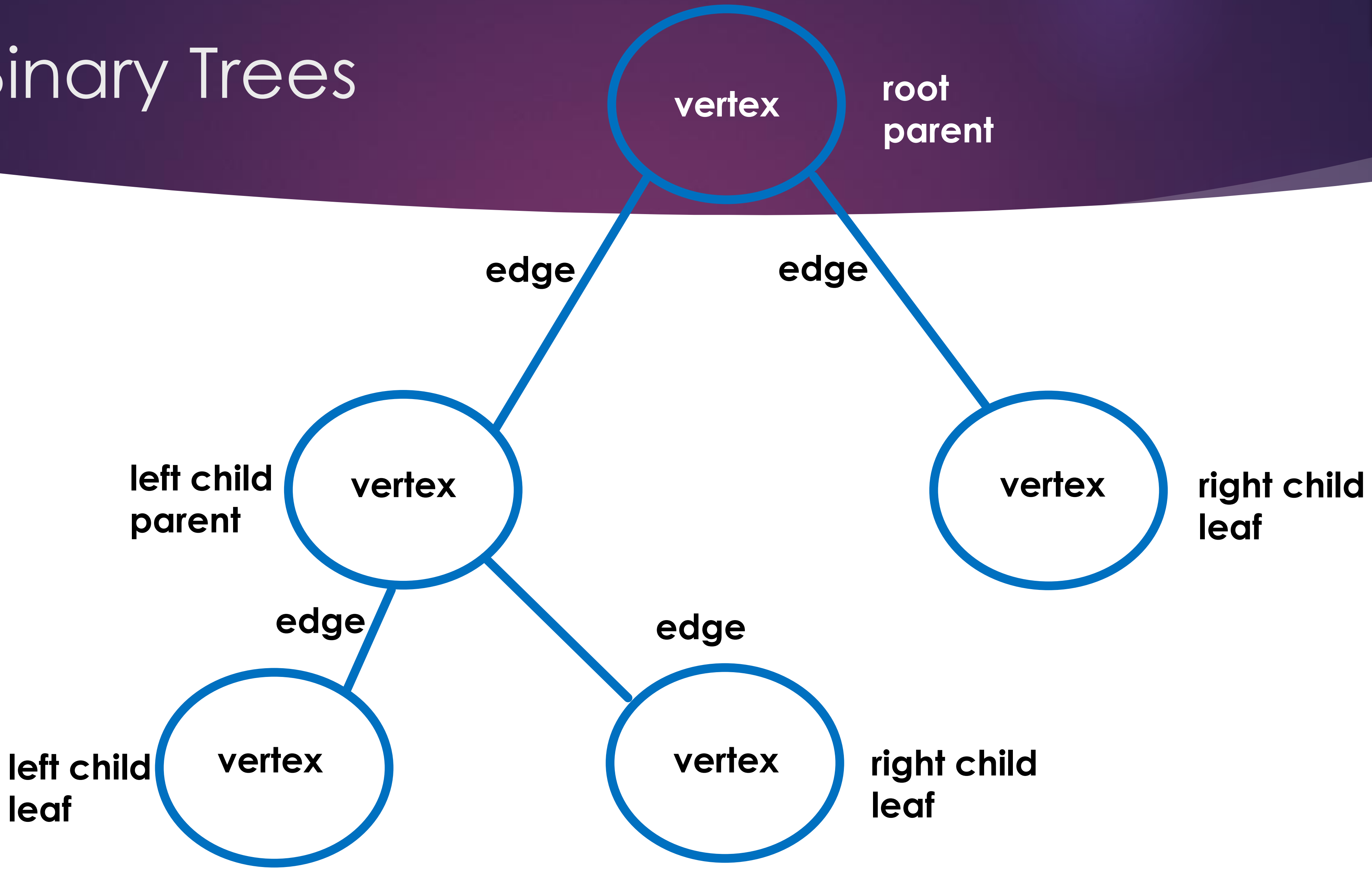
Not all Graphs are Trees



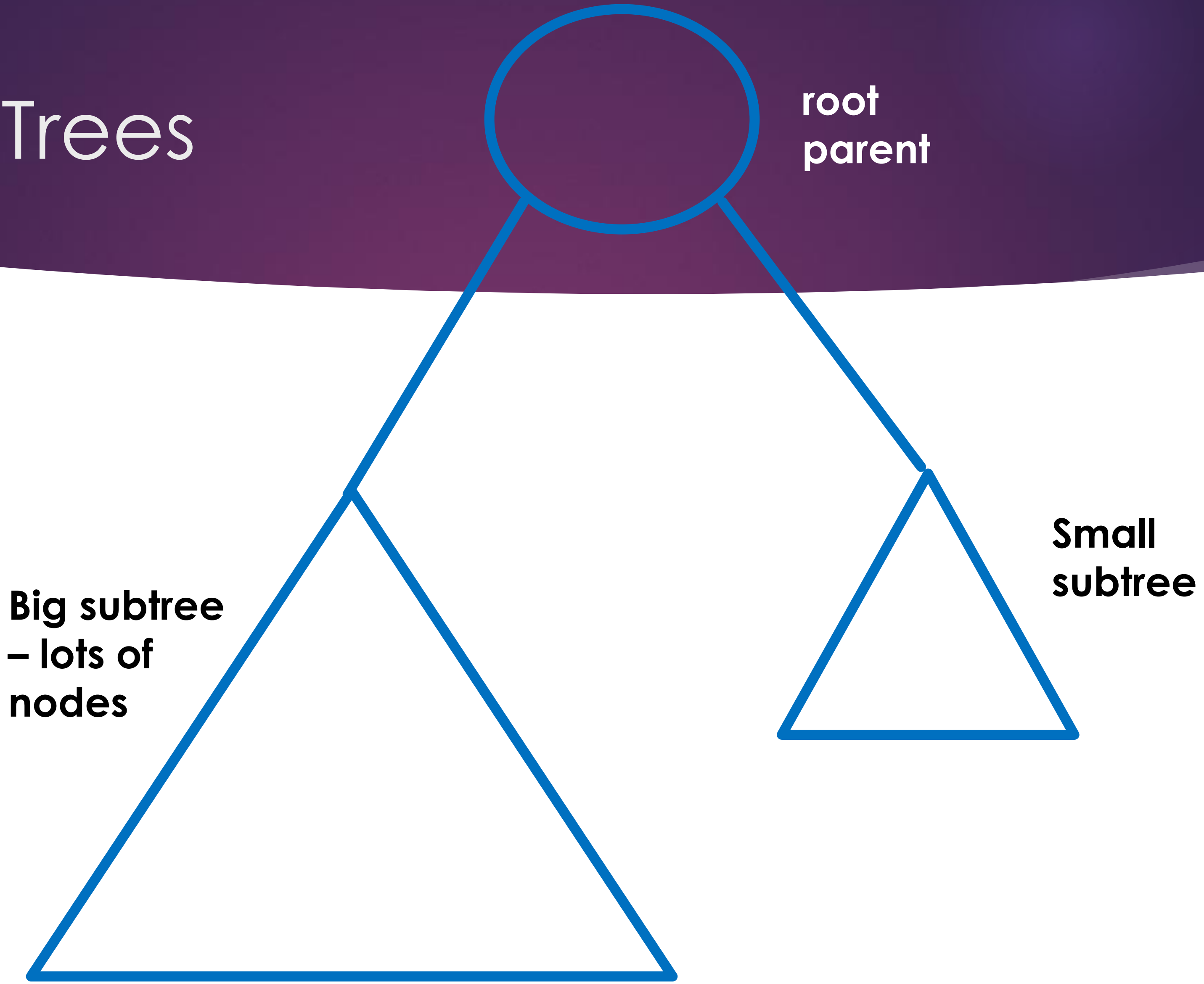
Trees


Exploring - C:\fsapps\FAM\Farsite4\examples\as...
File Edit View Go Favorites Tools Help
Folders
Desktop
My Computer
3½ Floppy (A:)
(C:)
Adobeapp
axhome
Cardwork
CDRom
Cfgsafe
Drivers
FARSITE4help
fsapps
acrobat3
applix
AutoXfer
Data
exceed.95
Fam
Behave
BehavePlus
Farsite
Farsite4
examples
ashley
Input
Output
1 object(s) selected
My Computer

Binary Trees

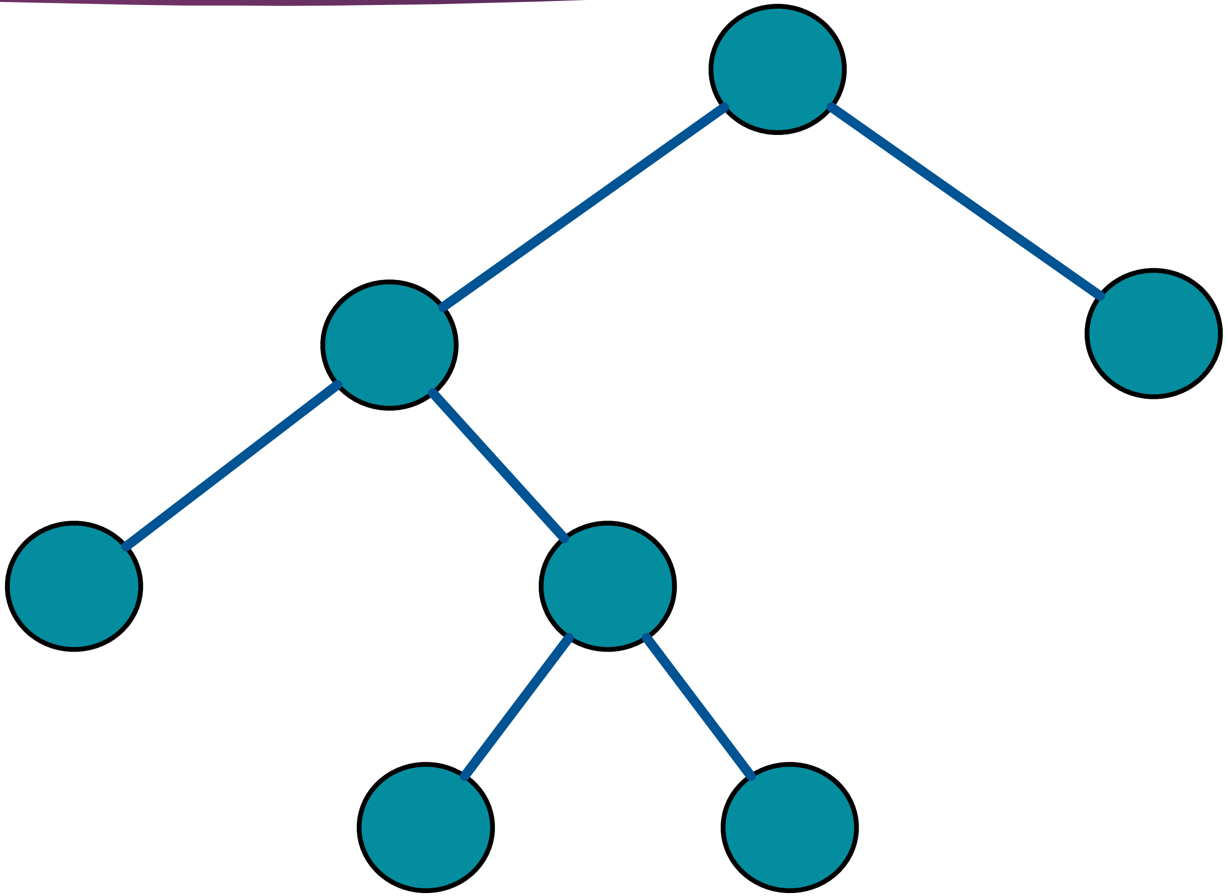


Binary Trees



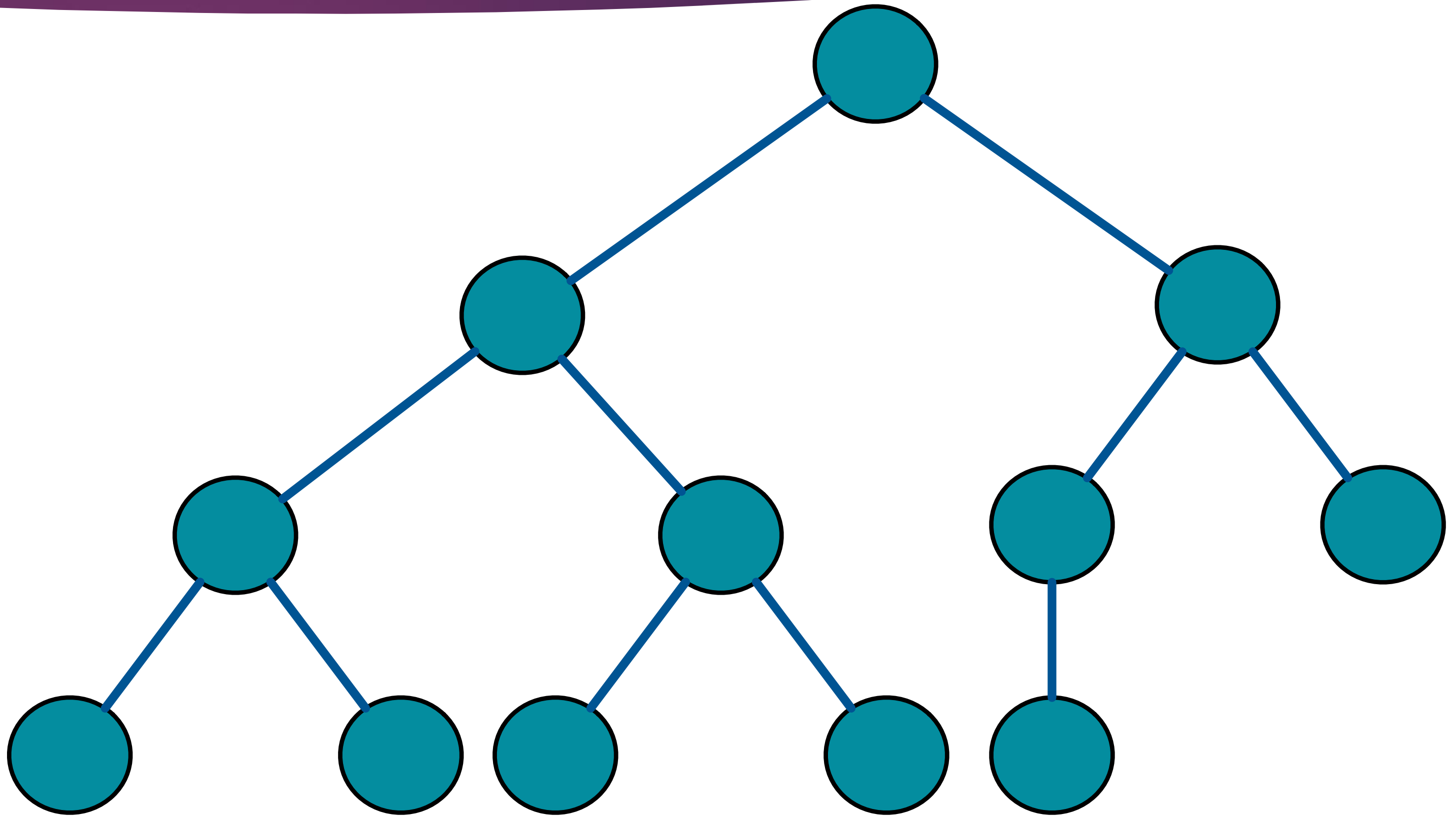
Full Binary Tree

- All Nodes have either Zero or Two Child Nodes



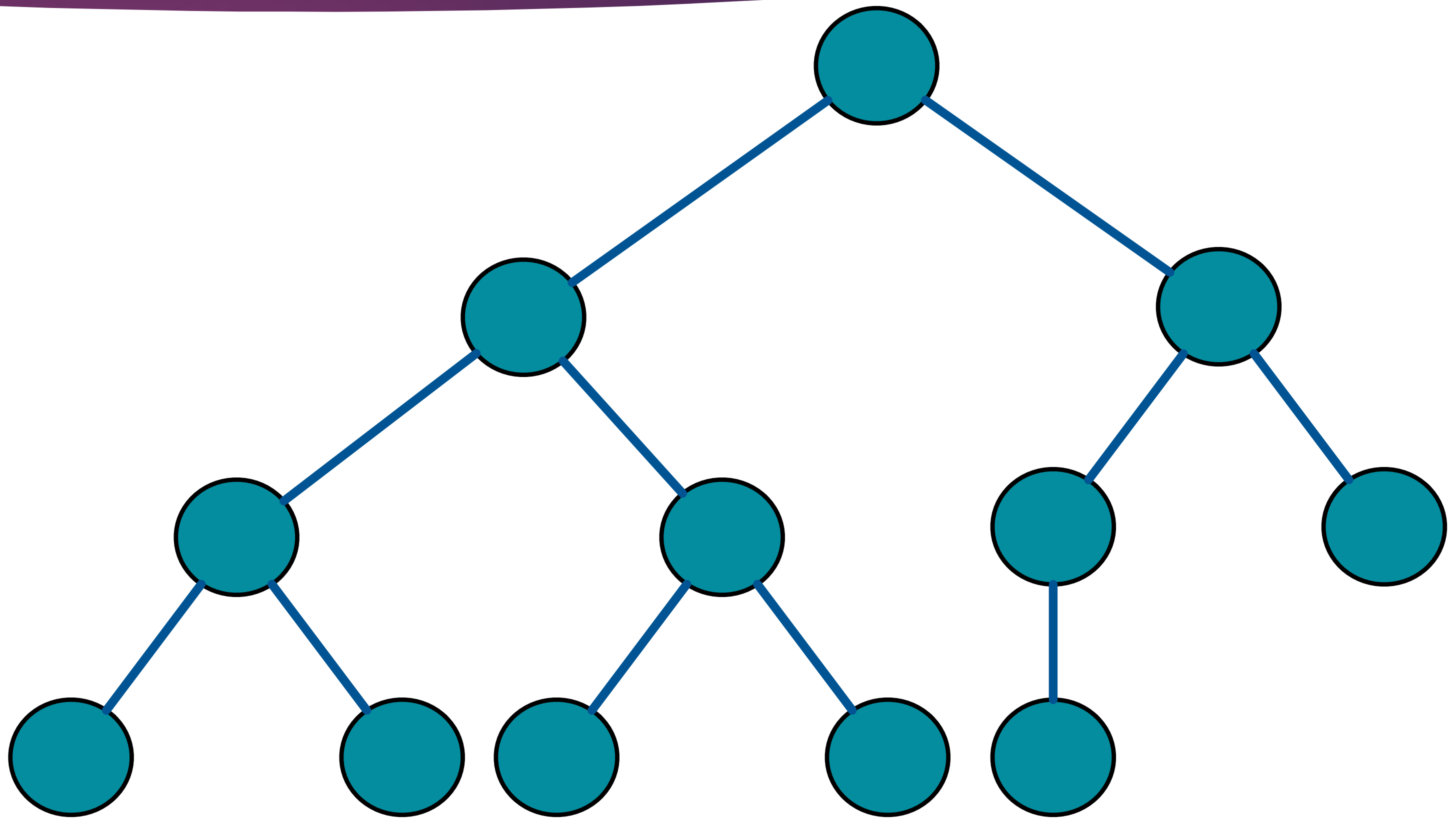
Complete Binary Tree

- All levels are full except possibly the last level, and all nodes on the last level are as far left as possible



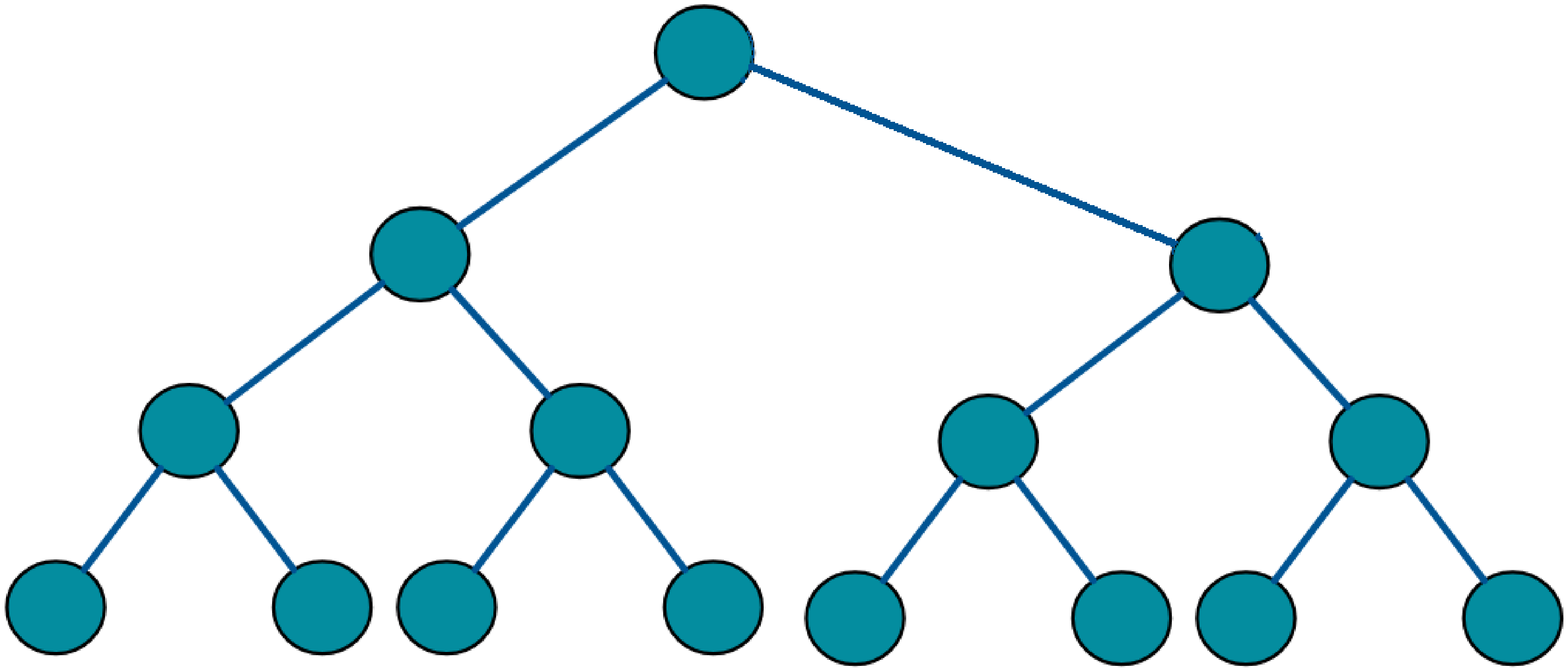
Balanced Binary Tree

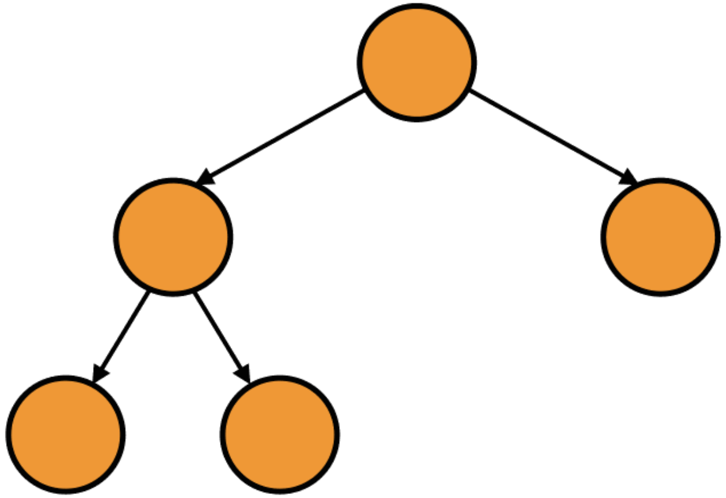
- The difference in height of any subtree is no greater than 1



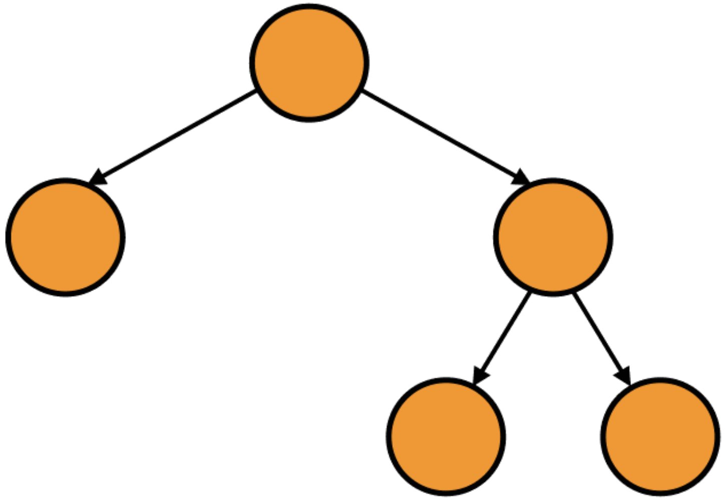
Perfect Binary Tree

- All interior nodes have two children
- All leaves have the same depth or same level

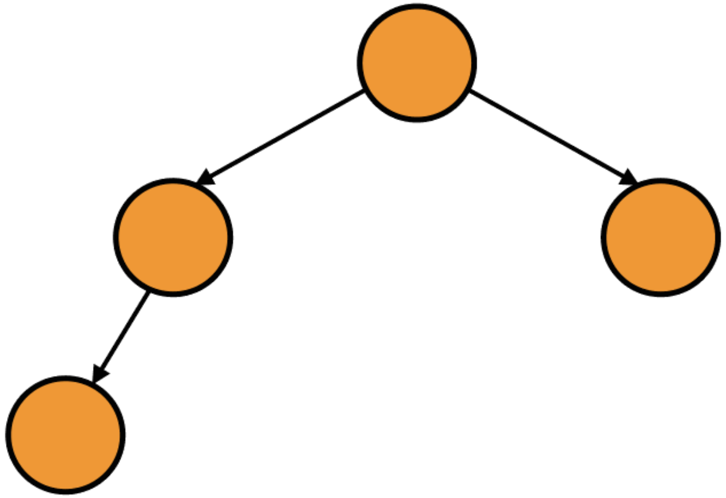




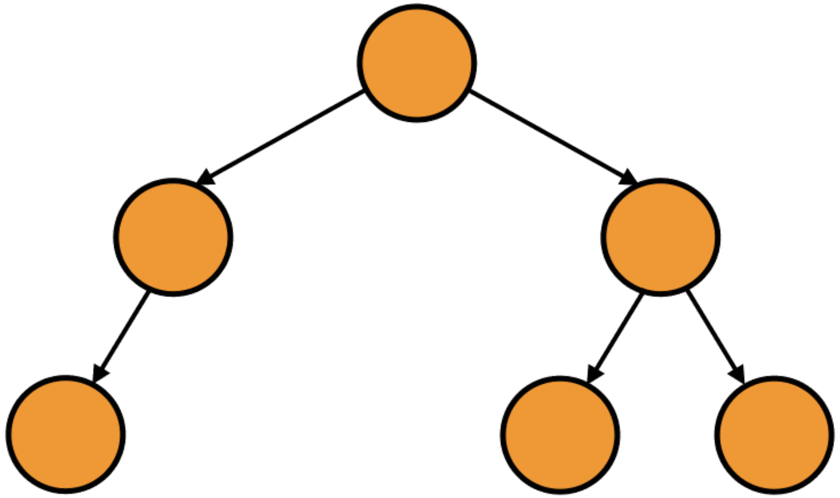
Complete and Full



Full but NOT Complete

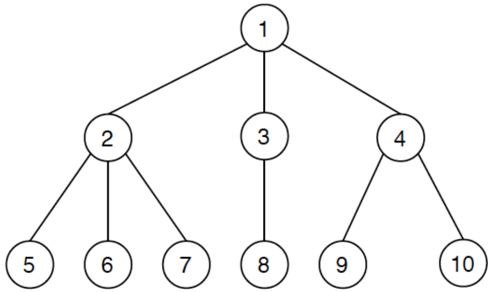


Complete but NOT Full

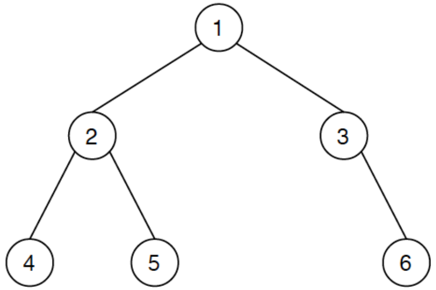


Neither Complete nor Full

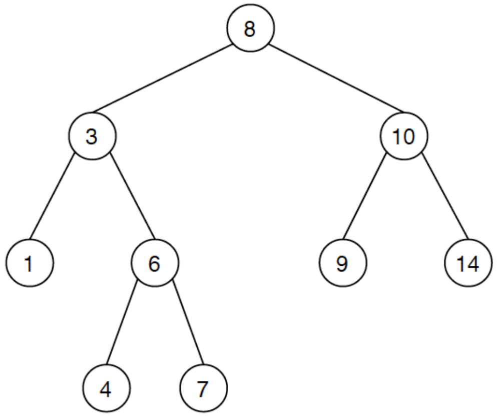
General Tree



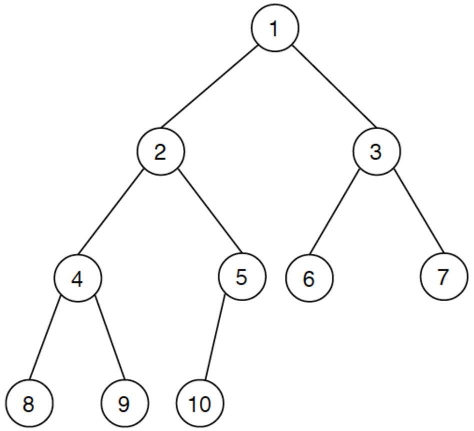
Binary Tree



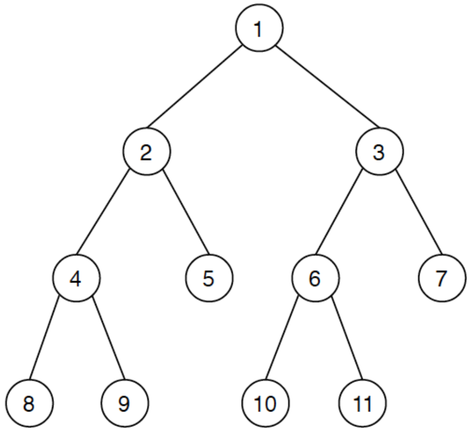
Binary Search Tree



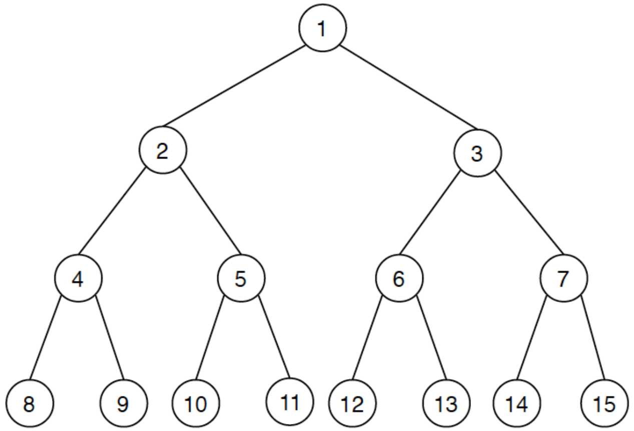
Complete BinaryTree



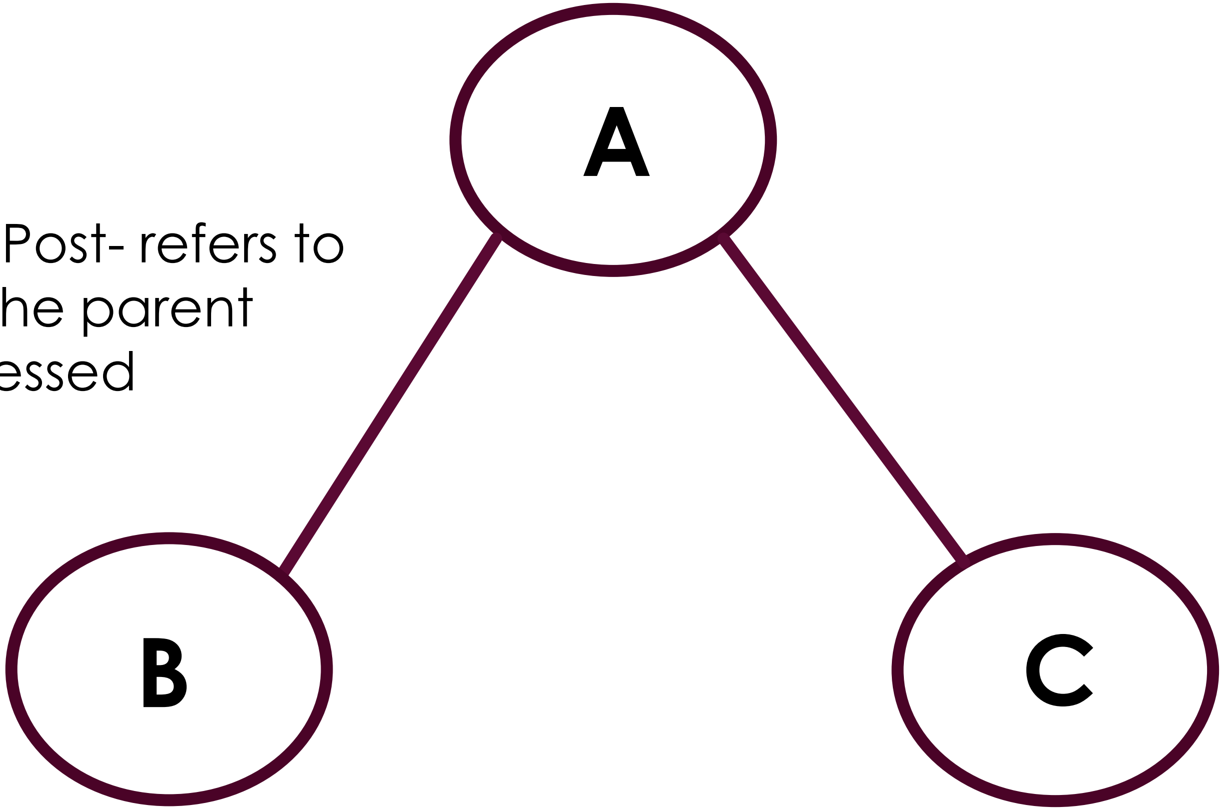
Full BinaryTree



Perfect BinaryTree



Traversal Order



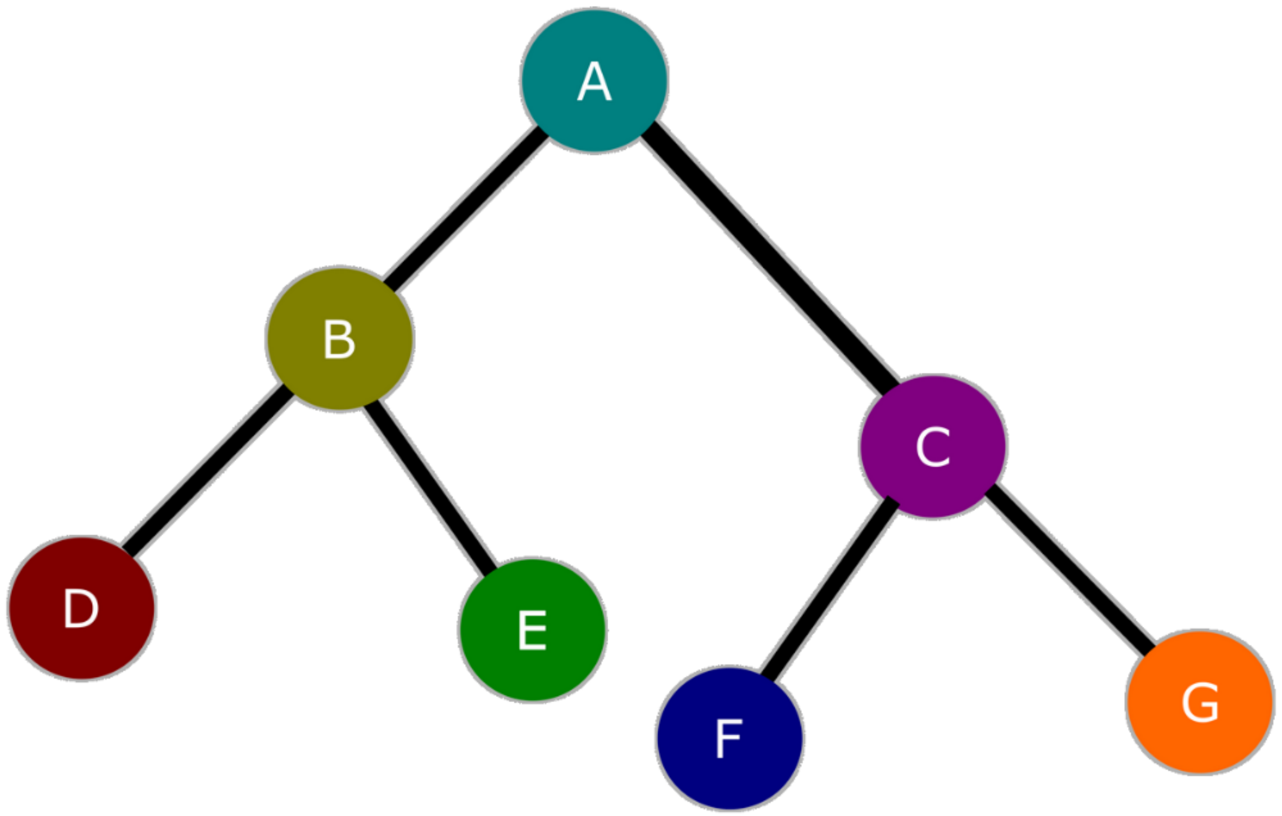
Pre- or Post- refers to when the parent is processed

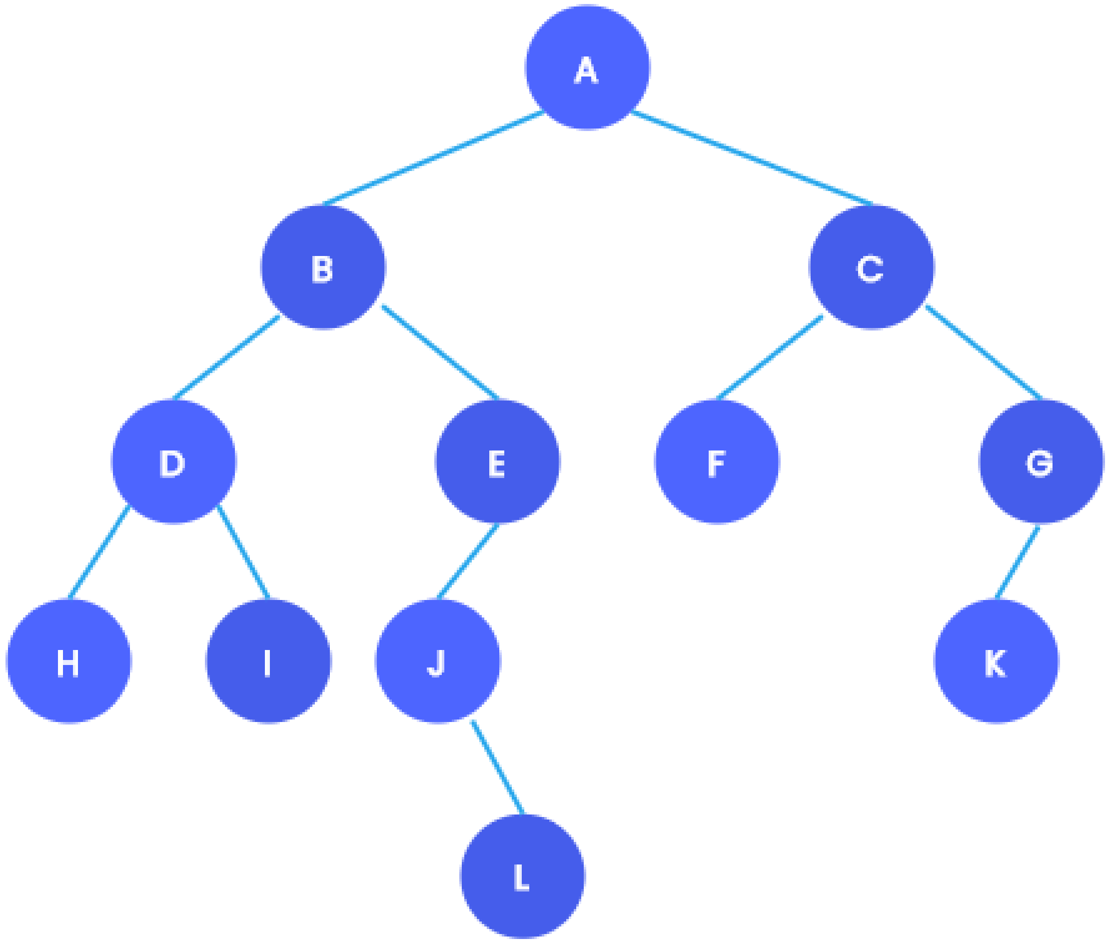
Level-Order Traversal: Level-by-level from left to right

Pre-Order Traversal:
Parent
Left Child (recursively)
Right Child (recursively)

In-Order Traversal:
Left Child (recursively)
Parent
Right Child (recursively)

Post-Order Traversal:
Left Child (recursively)
Right Child (recursively)
Parent


A
B
C
D
E
F
G



Pre-Order:

A, B, D, H, I, E J, L C, F, G, K

In-Order:

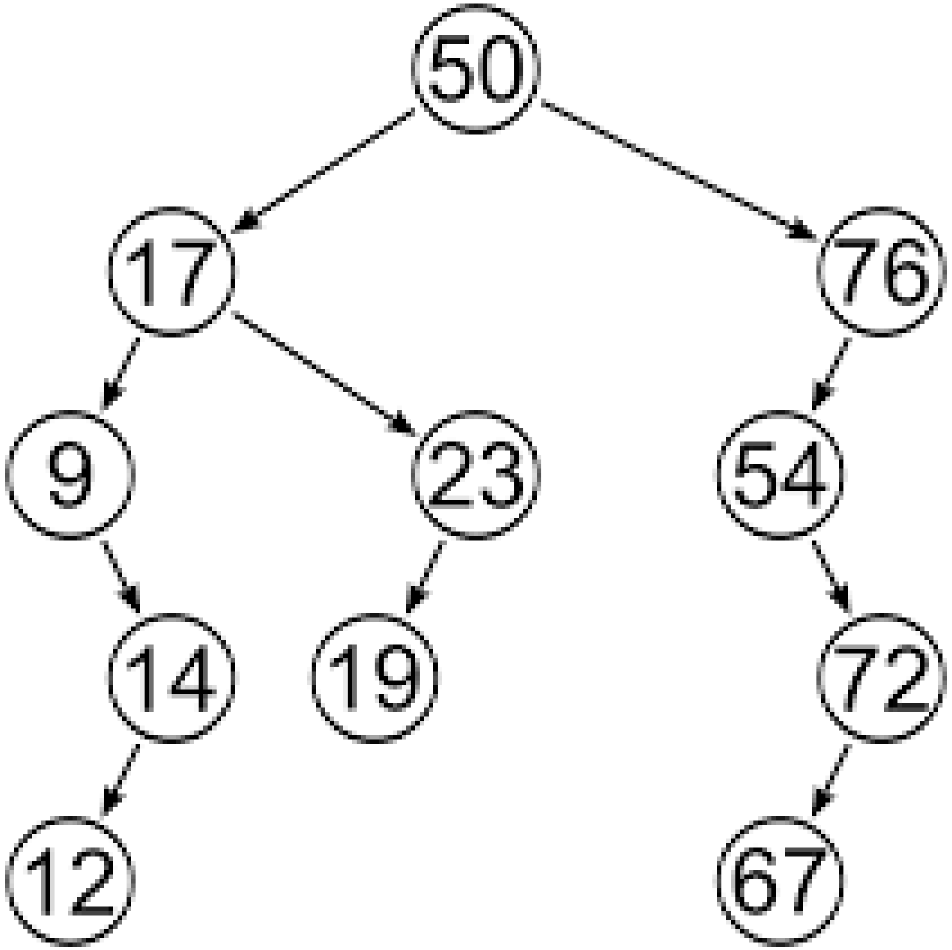
H, D, I, B, J, L, E, A, F, C, K, G

Post-Order:

H, I, D, L, J, E, B, F, K, G, C, A

Binary Search Tree

In-Order Traversal is in <u>increasing</u> values – it's *in order.*



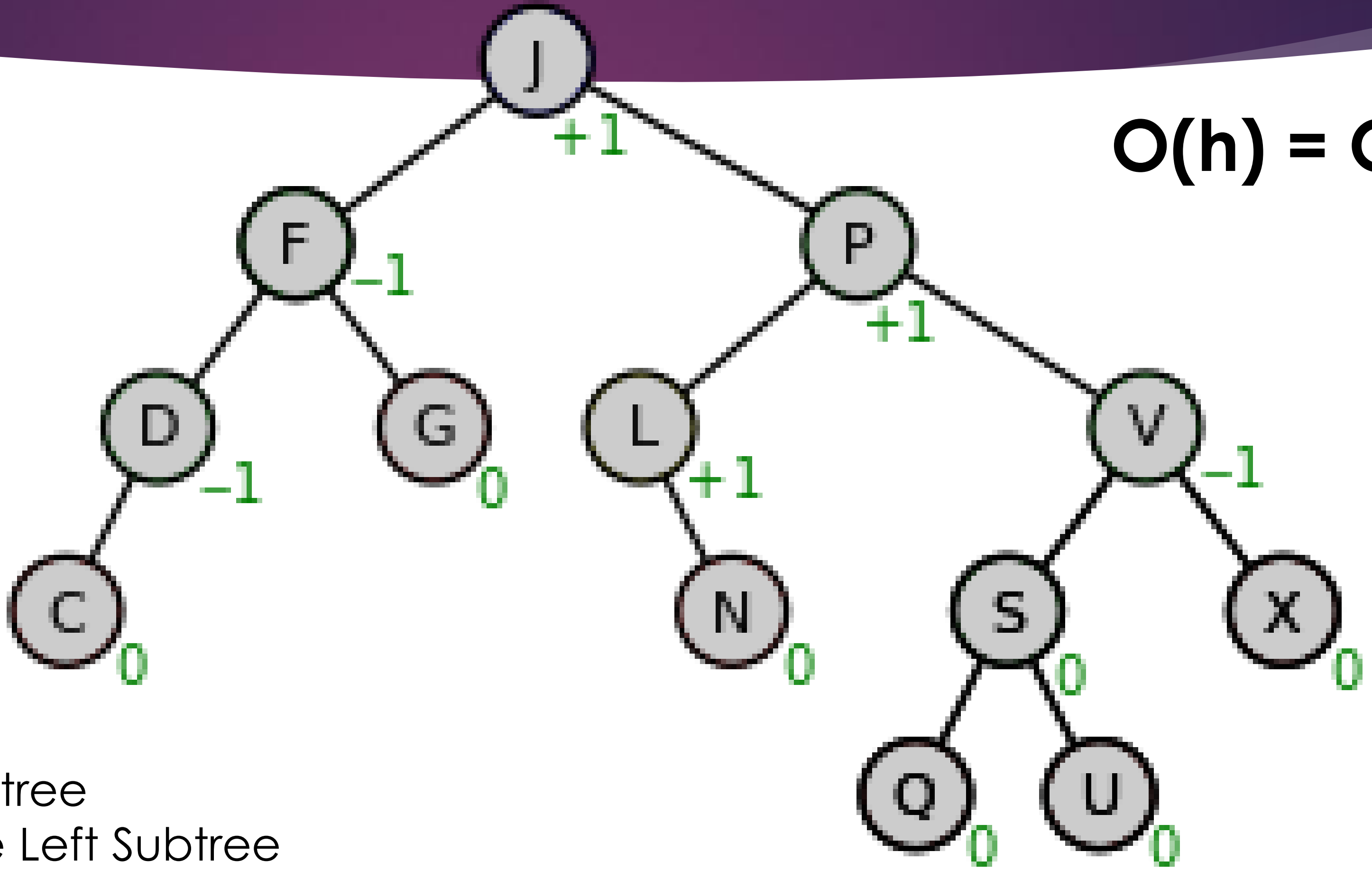
AVL Trees

Georgy Adelson-Velsky

Self-healing binary search trees named for two Russian mathematicians

Evgenii Landis

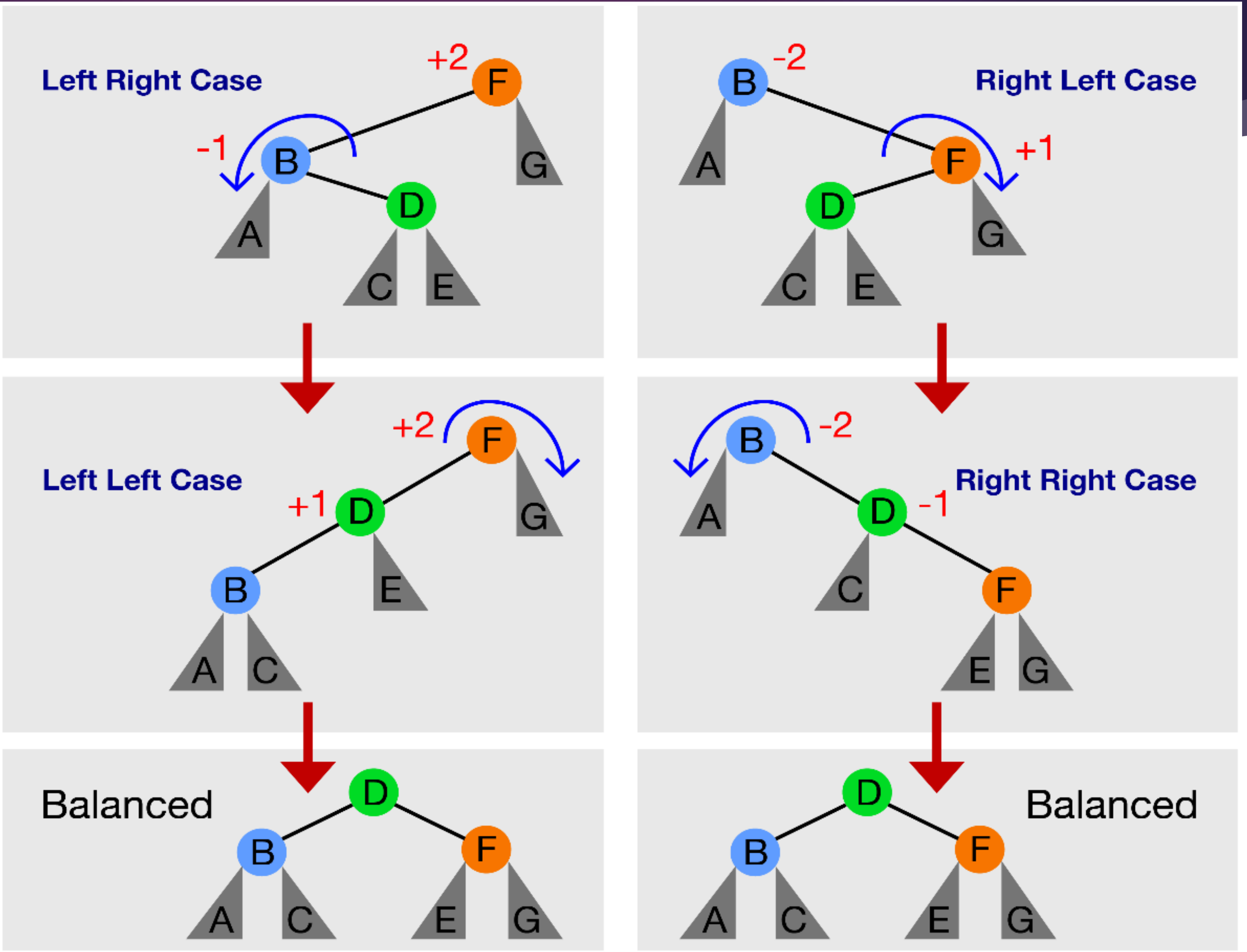
AVL Trees



O(h) = O(log N)

Balance factor:
Height of the Right Subtree
minus the Height of the Left Subtree

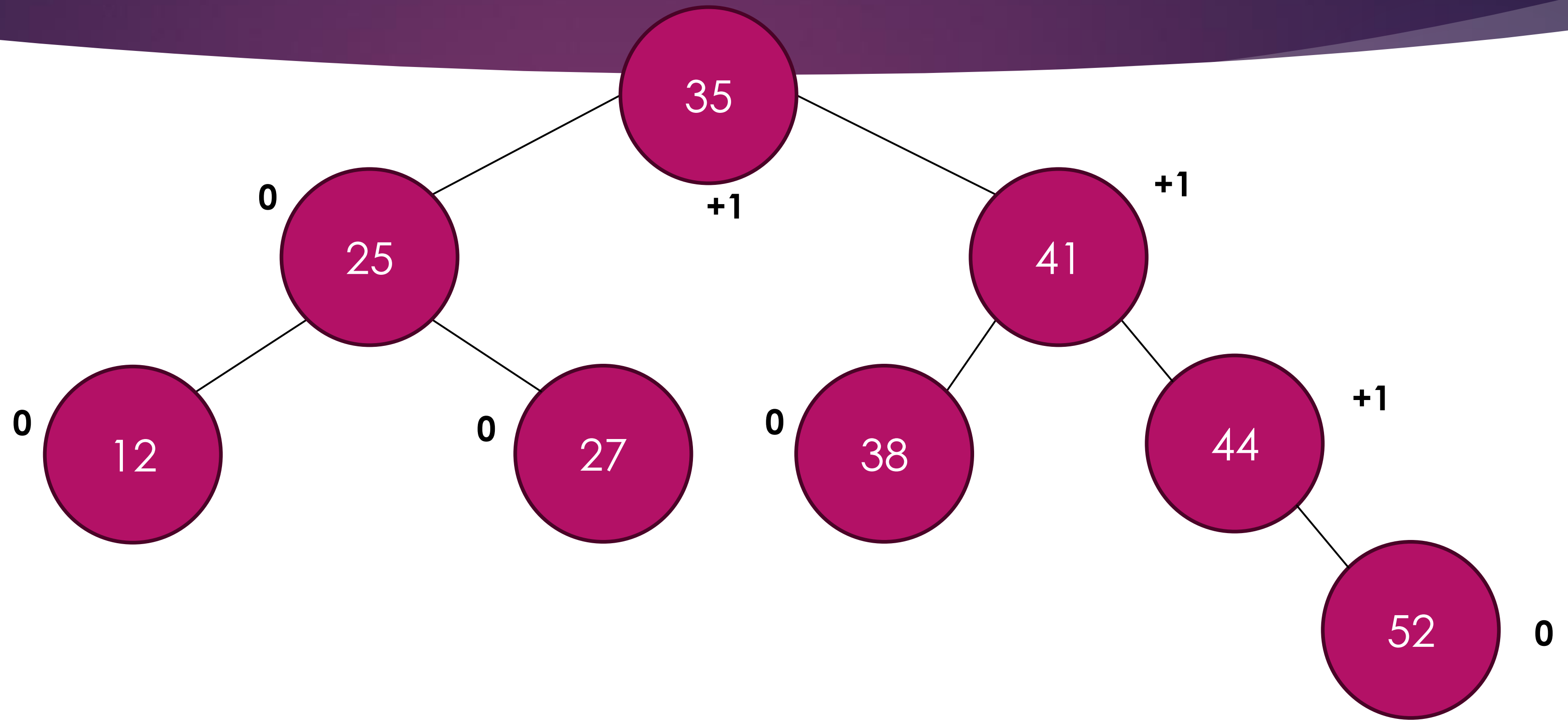
AVL Trees


Left Right Case
+2
F
-1
B
G
D
A
C
E
Right Left Case
-2
B
+1
F
A
D
G
C
E
Left Left Case
+2
F
+1
D
G
B
E
A
C
Right Right Case
-2
B
-1
D
A
C
F
E
G
Balanced
D
B
F
A
C
E
G
Balanced
D
B
F
A
C
E
G

AVL Tree Rotation

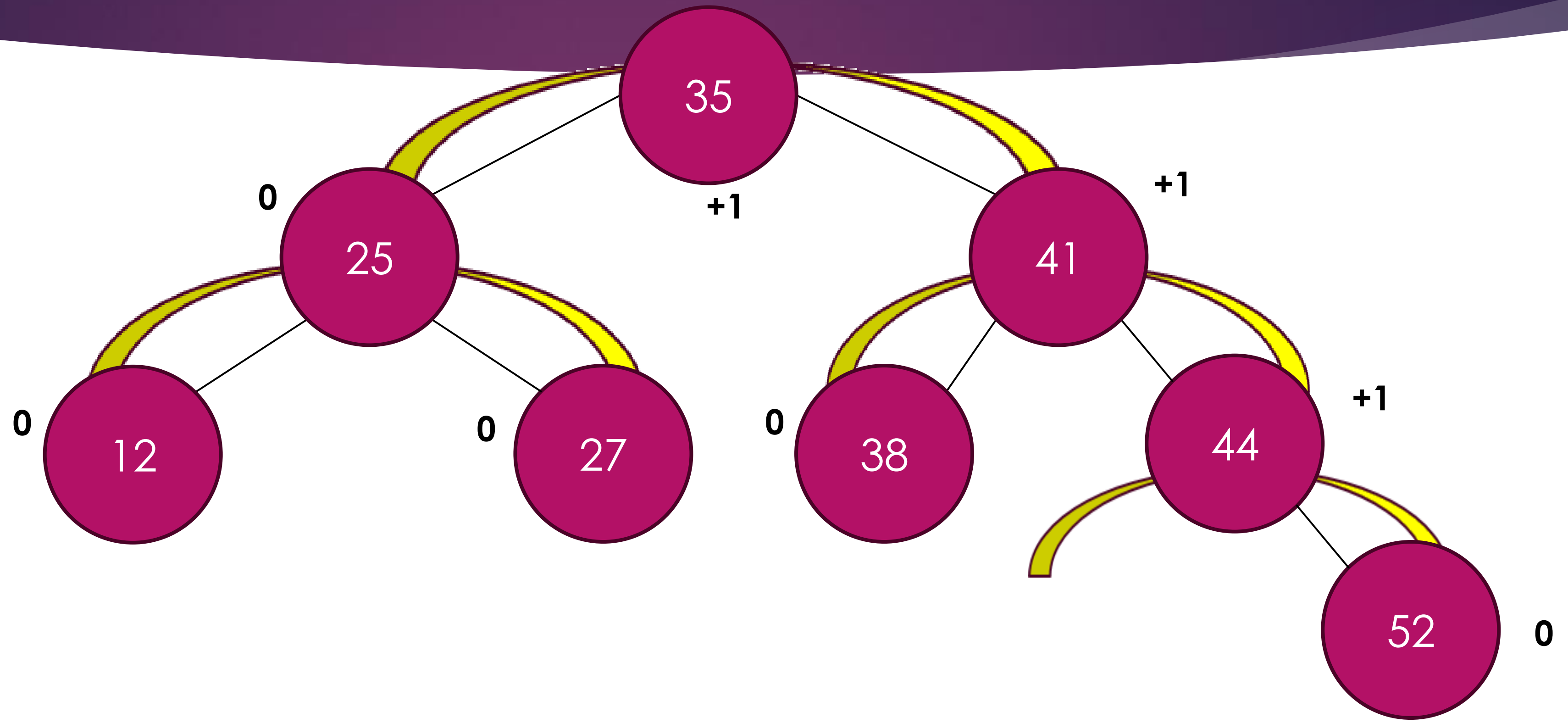
- An AVL Tree is a self-balancing Binary Search Tree (BST)
- Follows the same insertion rules as BST
- In-Order Traversal is *in numerical order*
- Balance Factor is the height difference between the right and left subtrees
- Balance Factor must be -1, 0, or +1
- Rotations must preserve the in-order traversal
- Rotations happen at the lowest level first

AVL Tree Rotation



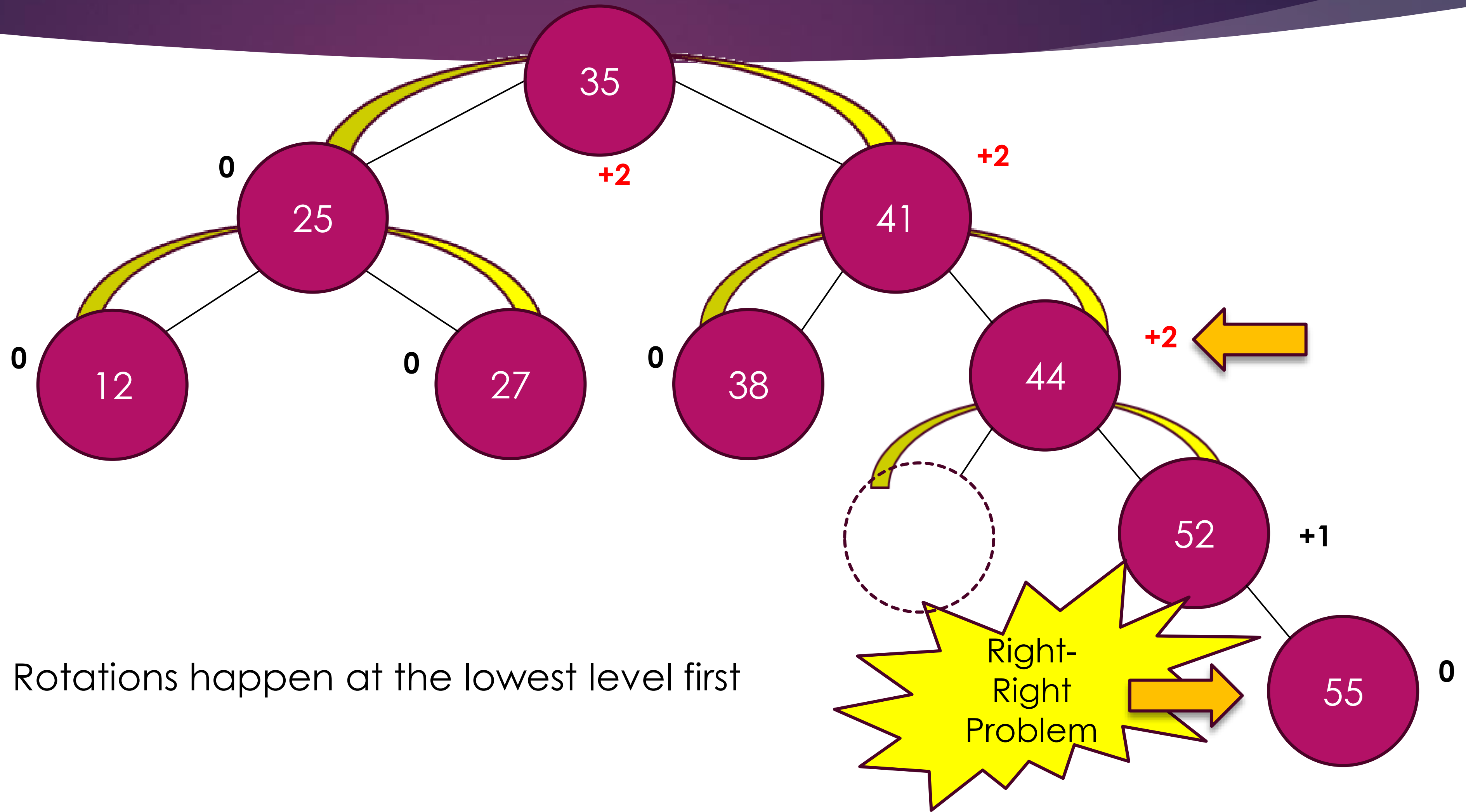
Balance Factor: Height difference between left and right subtrees
Balanced if factor is -1, 0, or +1

AVL Tree Rotation



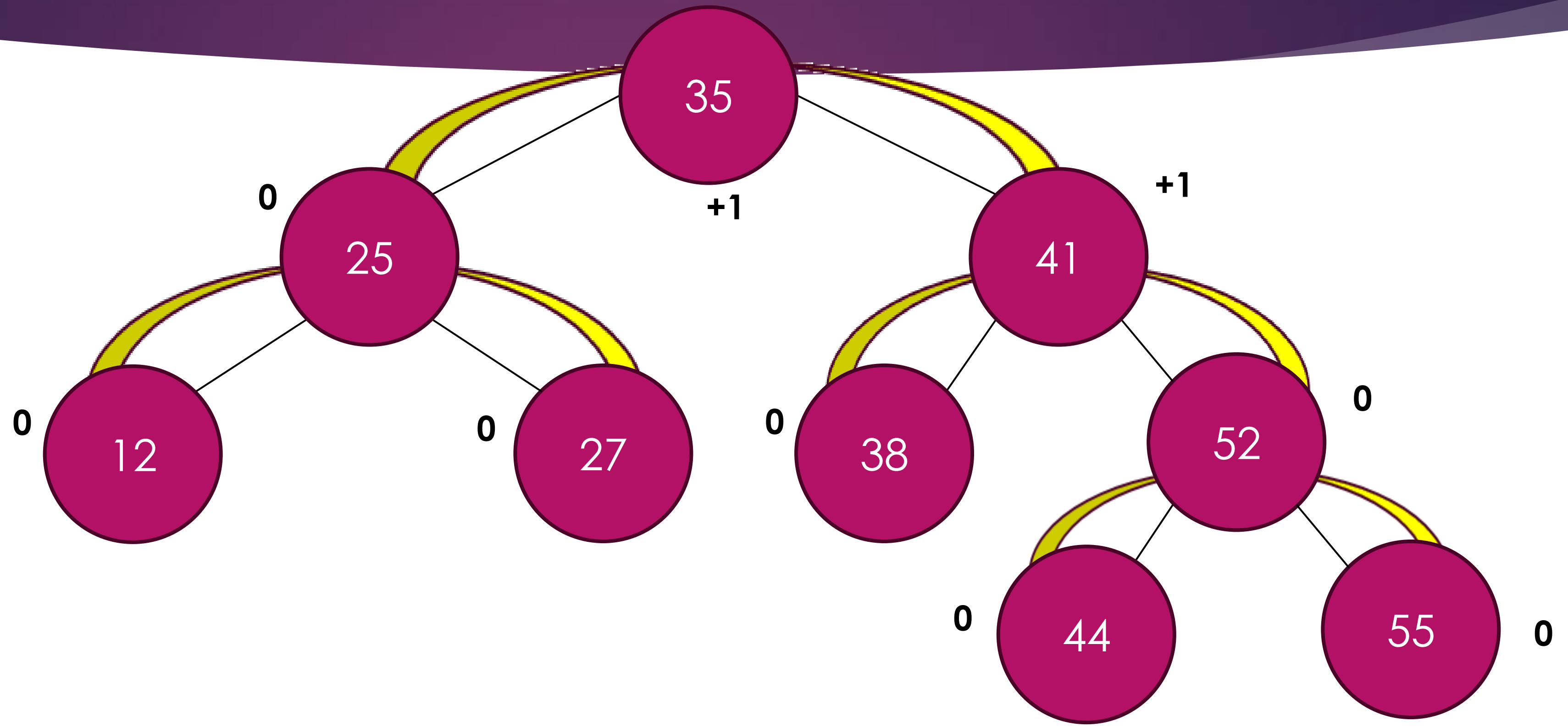
Rotations must preserve in-order traversal
Single Rotation can only take place along the in-order arc

AVL Tree Rotation: Single Rotation

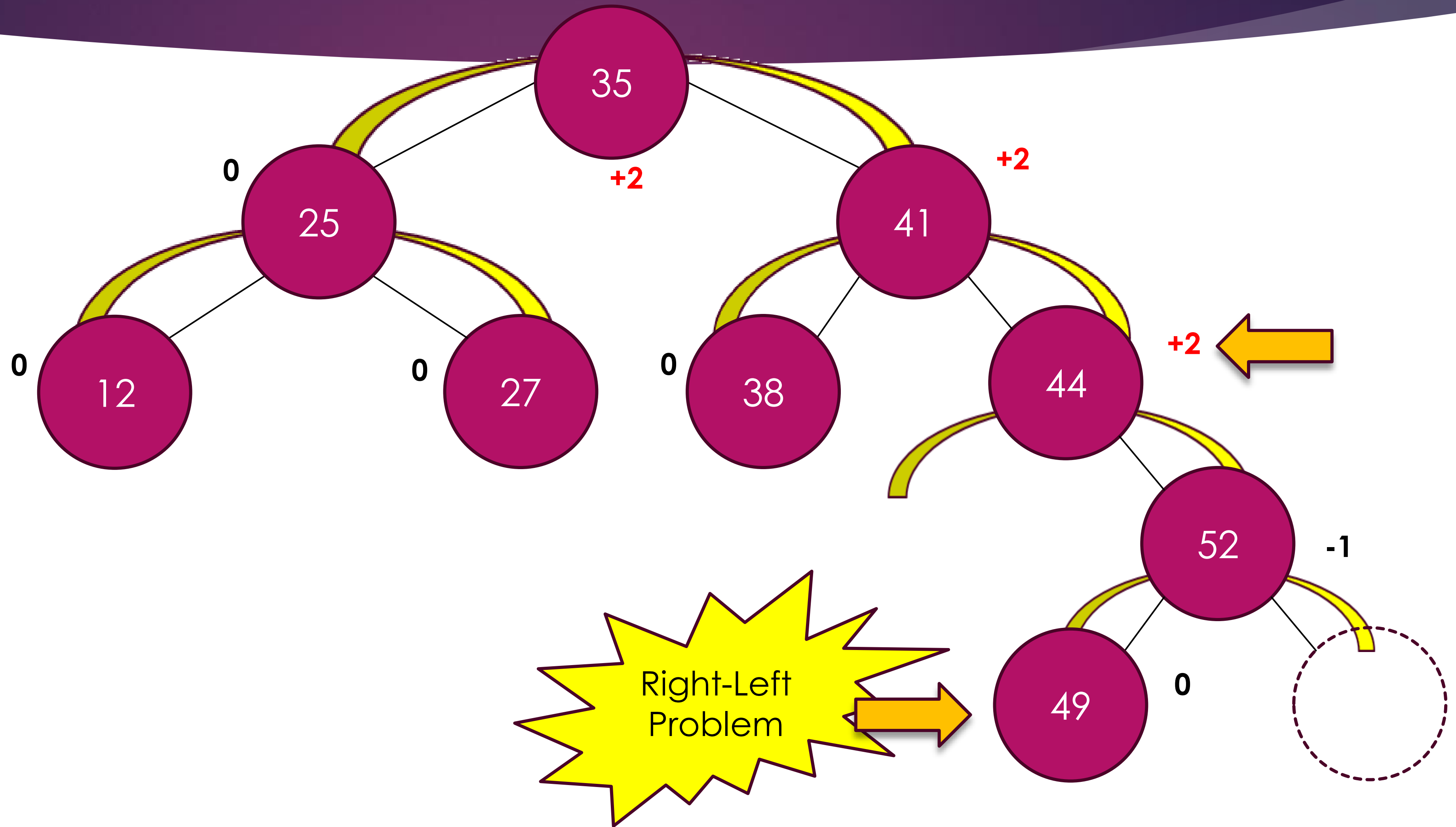


Rotations happen at the lowest level first

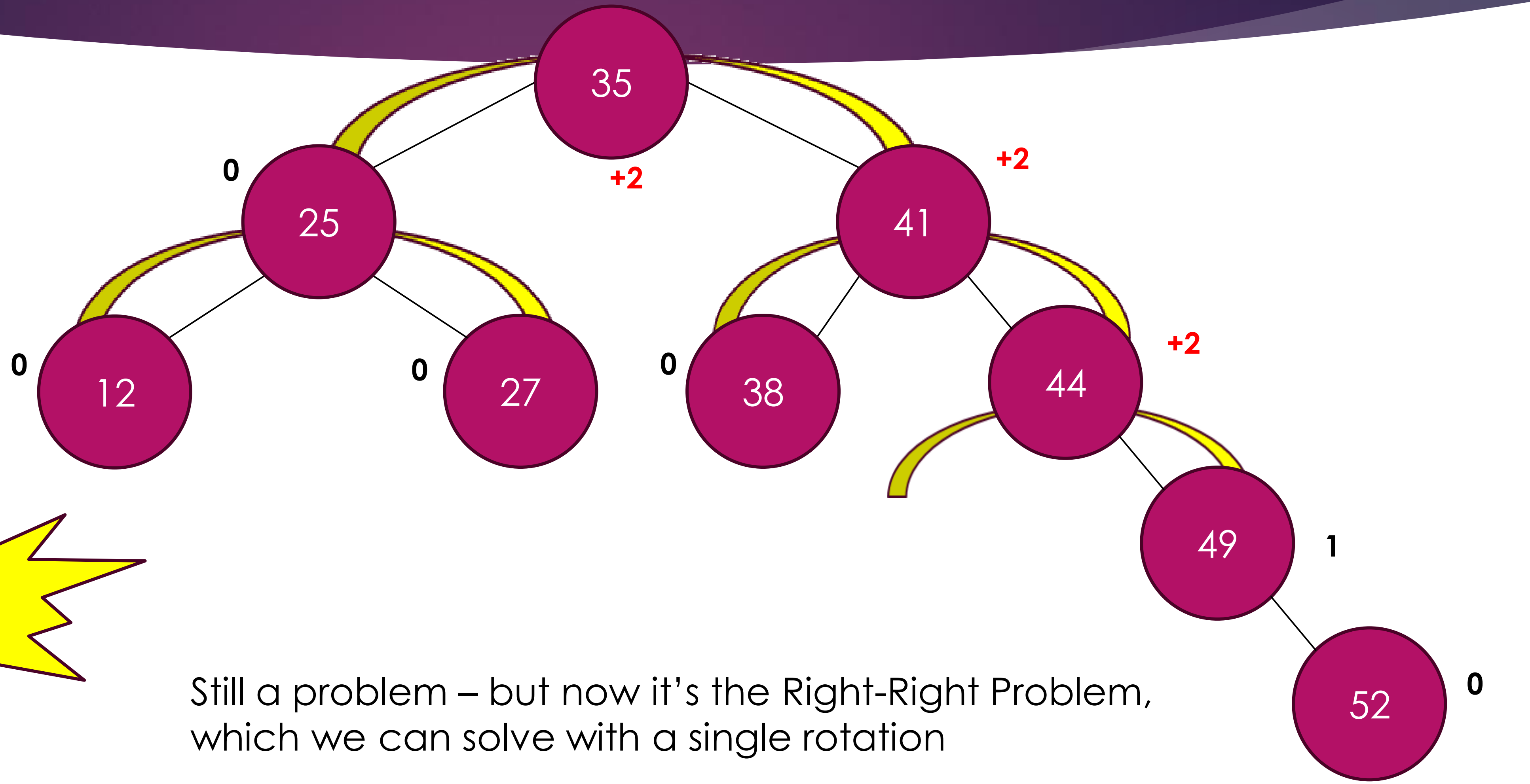
AVL Tree Rotation: Single Rotation



AVL Tree Rotation: Double Rotation

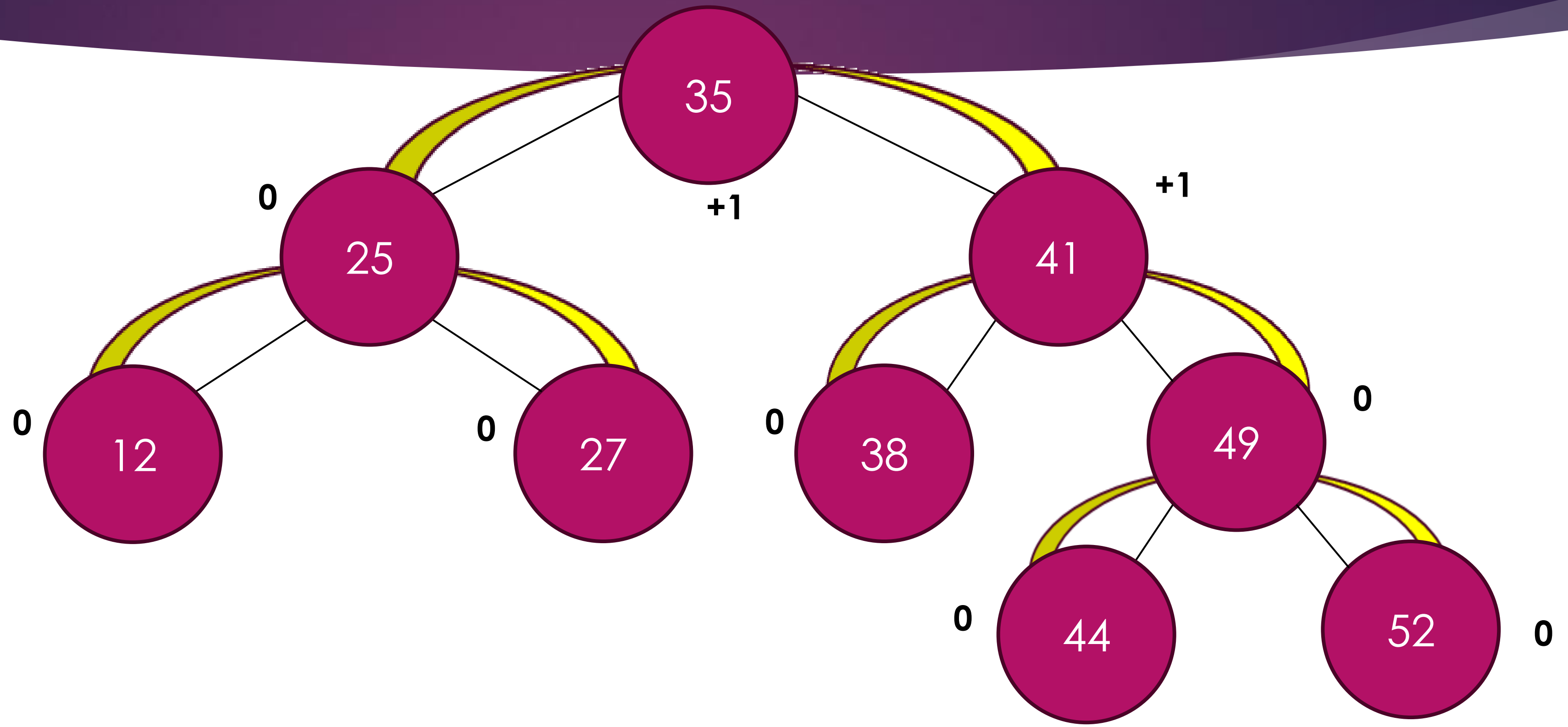


AVL Tree Rotation: Double Rotation

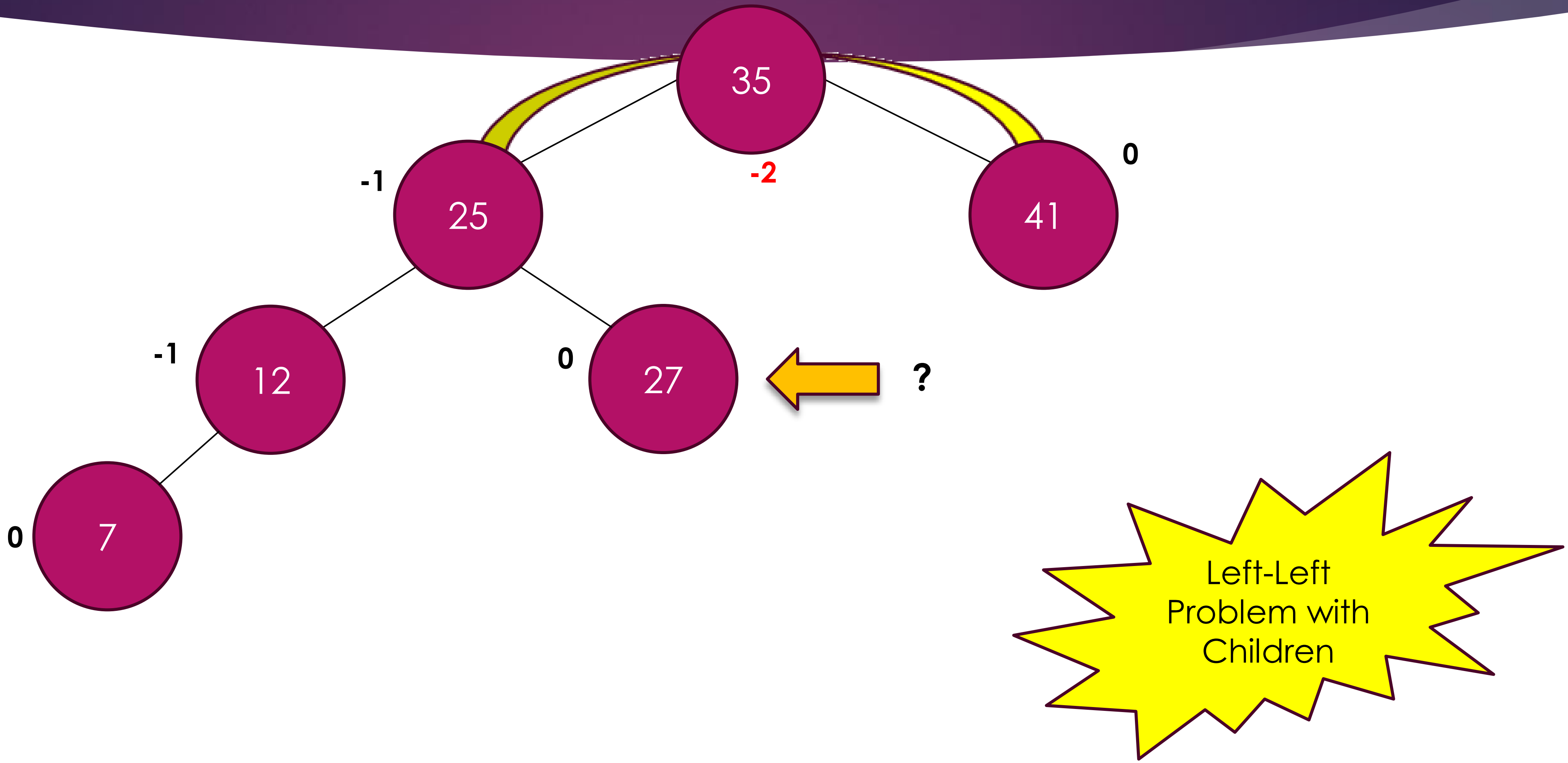


Still a problem – but now it's the Right-Right Problem, which we can solve with a single rotation

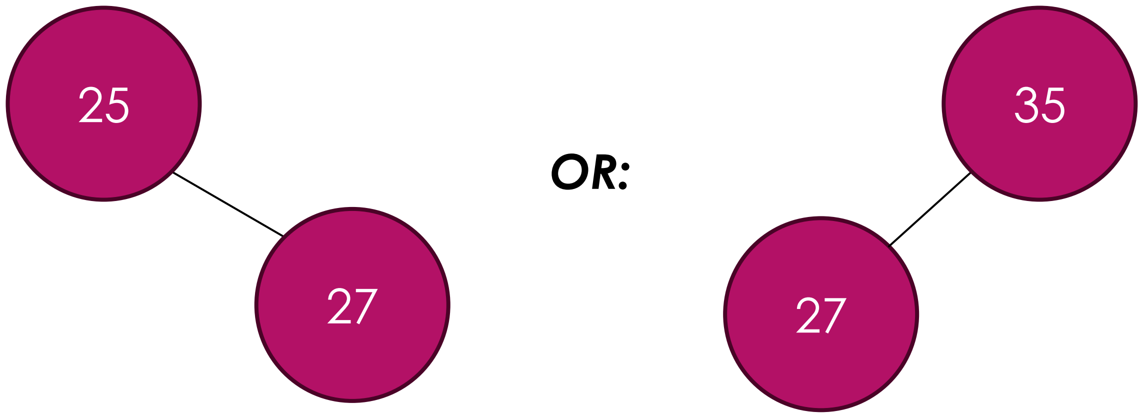
AVL Tree Rotation: Double Rotation



AVL Tree Rotation: Rotation with Children

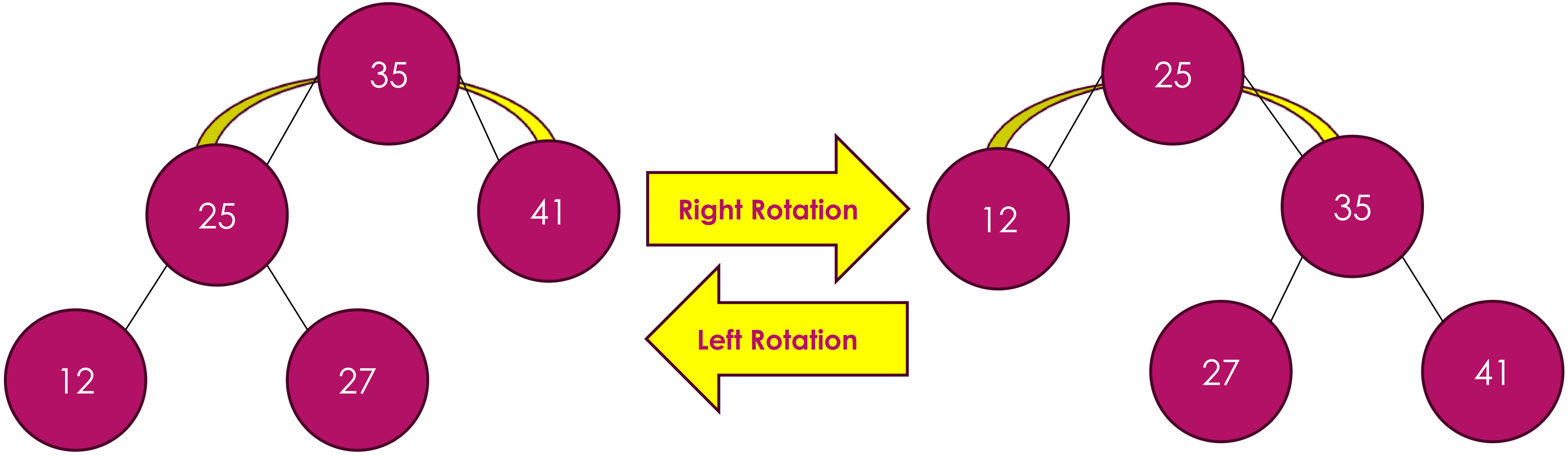


AVL Tree Rotation: Rotation with Children



Rotation works because 27 could be a child of either node

AVL Tree Rotation: Rotation with Children



Always maintain the traversal order

AVL Rotation

https://en.wikipedia.org/wiki/AVL_tree

AVL Rotation

- Right-Right Problem: Left Rotation (single)
- Left-Left Problem: Right Rotation (single)
- Right-Left Problem: Double Rotation
  - Right Rotation (one level down)
  - Left Rotation
- Left-Right Problem: Double Rotation
  - Left Rotation (one level down)
  - Right Rotation
- Rotation with Children: Conflicting subtree moves to the former parent

AVL Rotation

If the order of numbers in an In-Order Traversal changed after a Rotation, you did it wrong.