Trees and Heaps

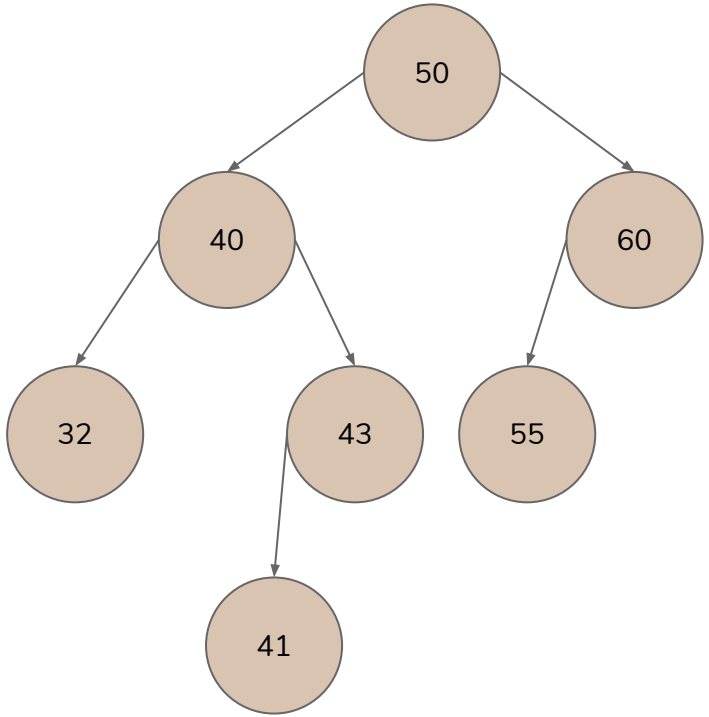
Leap@CMU 2017

“A Tree is a data structure made up of nodes, vertices, or edges”

-Wikipedia

Terminology

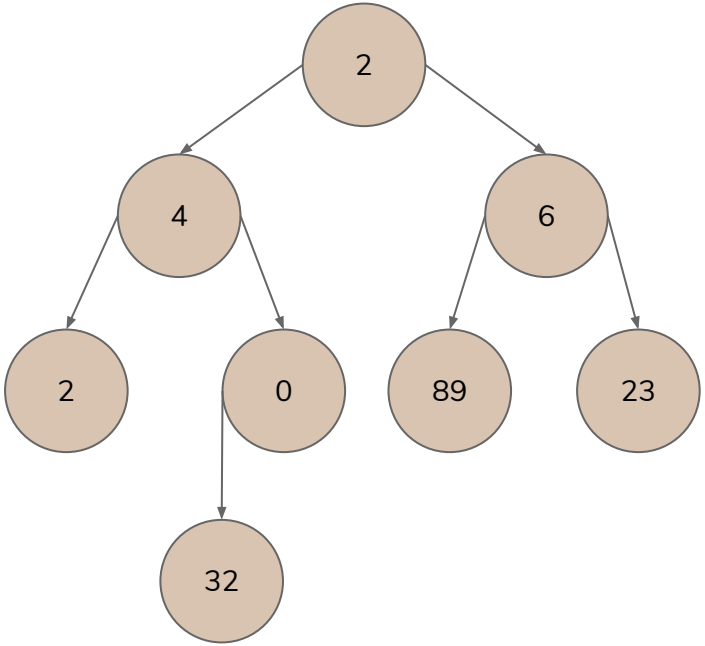
- Points that contain values are known as **nodes**
- The arrows connecting nodes are **edges**
- A **parent node** connects to **child nodes** with edges
- The top node is the **root**
- The **depth** of a node is its distance from the root in edges



Traversing a Tree

- Traversing a tree means listing all of the nodes in a given tree in a logical manner
- **Depth-first traversing** means listing down the tree to the deepest node before going across
- **Breadth-first traversing** means you list layer-by-layer and only moving down when the rightmost node has been hit

Worked Example



Depth First: [2, 4, 2, 0, 32, 6, 89, 23]

Breadth First: [2, 4, 6, 2, 0, 89, 23, 32]

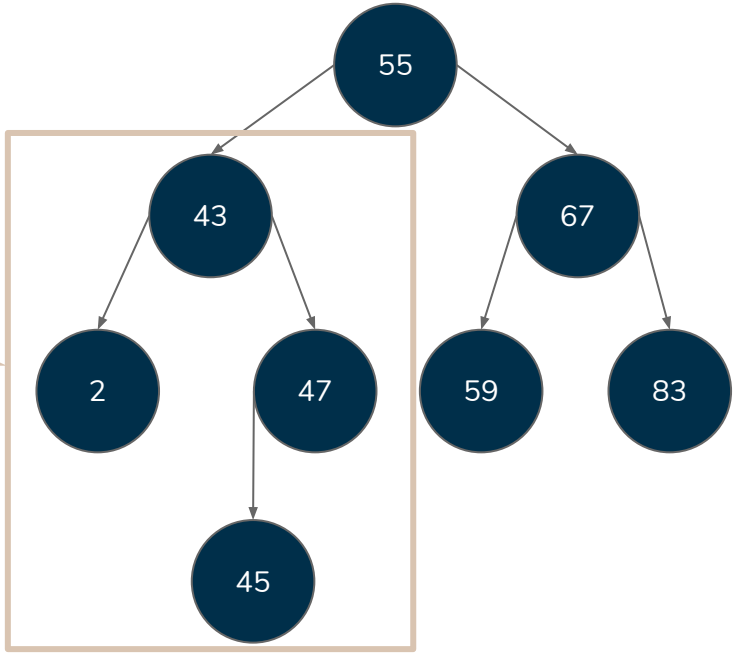
A Binary Search Tree is a special type of tree

Binary Search Tree Properties

- Can only be implemented with comparable data (the data can be sorted)
- Each parent node can have up to 2 child nodes (one less than, one greater than)
- Every node to the right of a parent node must be greater than it
- Every node to the left of a parent node must be less than it
- For each node, the “subtree” or section that is created if the child node was treated as the root is also a binary search tree

“Subtree” Example

This is still a Binary Search Tree

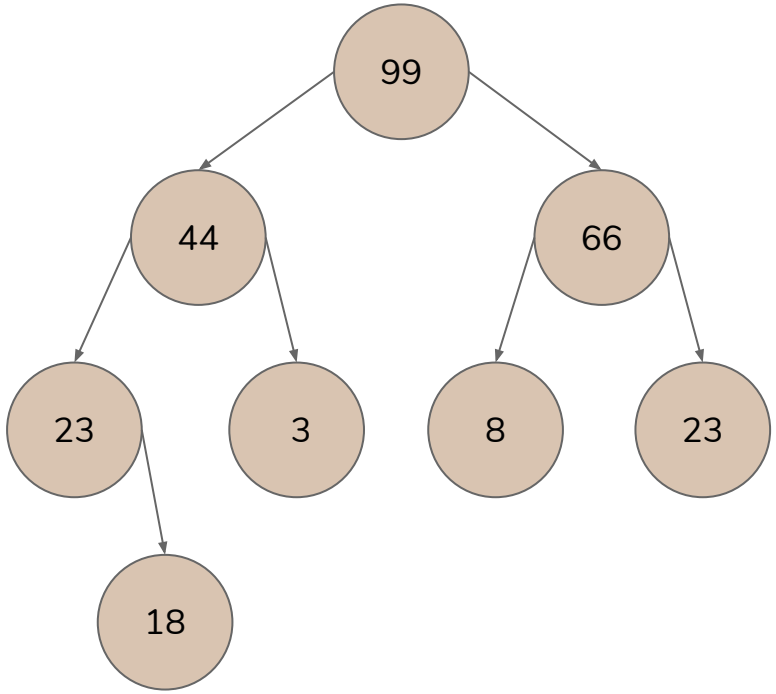


Java requires that you implement your own trees

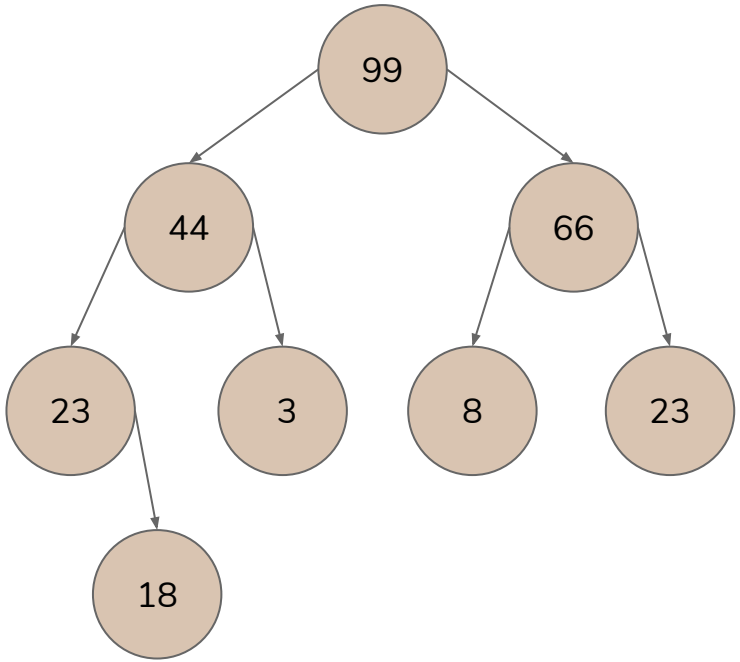
A Heap is another type of tree

What is a Heap?

Heaps are a specific type of tree with the explicit property that the root is the most extreme value (either largest or smallest) and all subsequent child nodes are less extreme (smaller in the first case, larger in the second) than the parent.



Pretty Print



```
{99{44{23{18},3},66{8,23}}
```

Project Guidelines

1. Implement a working tree
2. Implement a binary search tree
3. Add and search methods
4. Delete Method
5. Check if a tree is a heap
6. Pretty-print the trees in the console

*Note: It would help to think with an OOP mindset