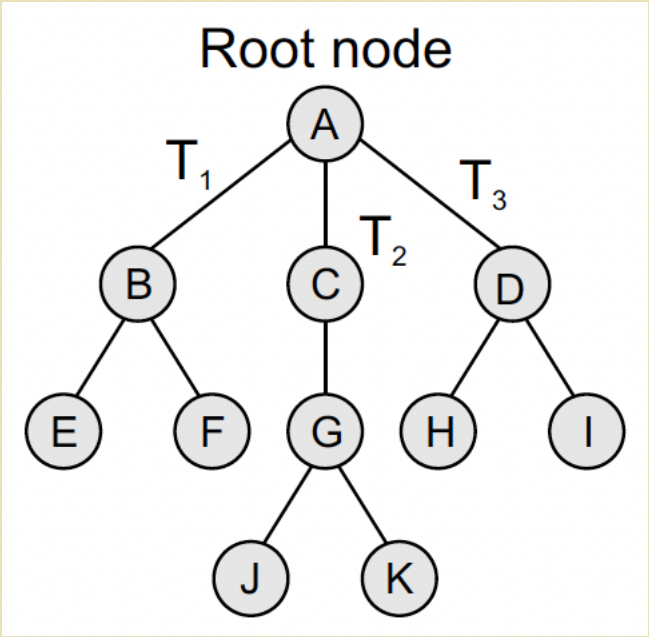
Comparison of Trees

Notes by Yan Yan

Learning Objectives

1. Describe the differences between different trees introduced
2. Remember the time complexities of common tree operations

Tree v.s. Binary Tree



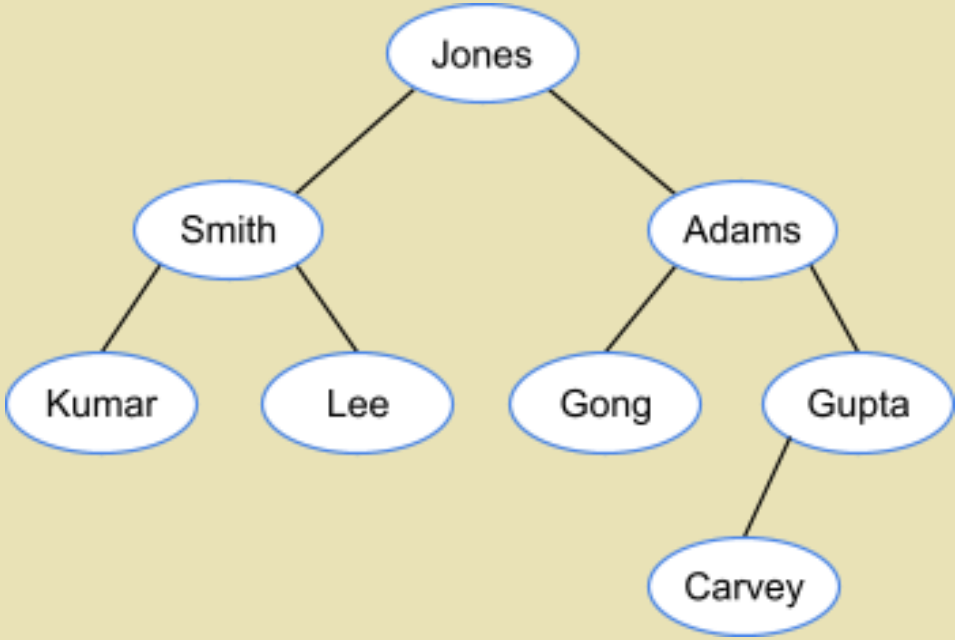
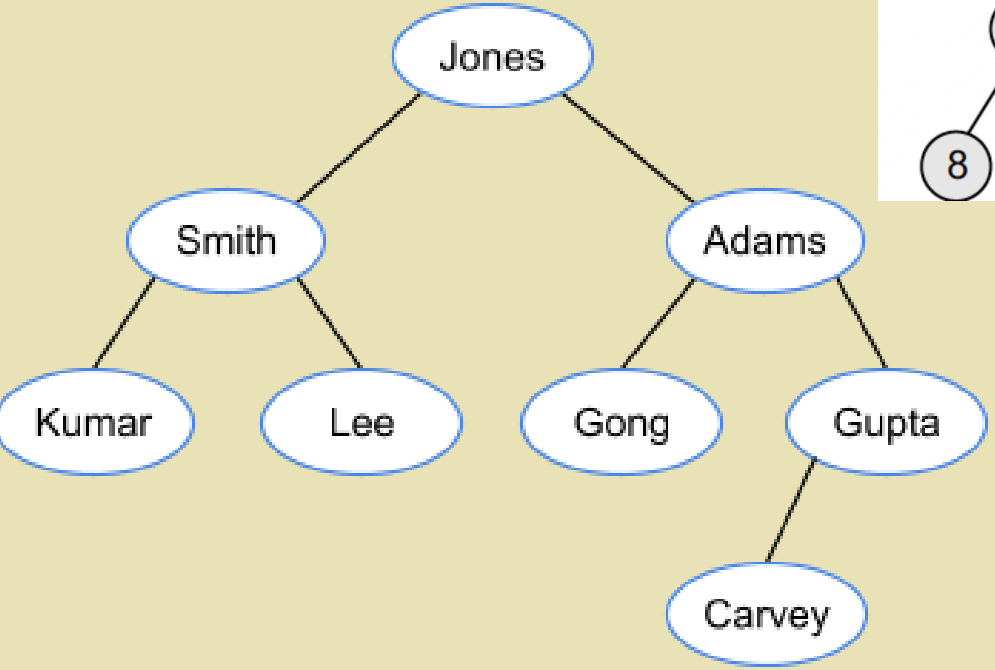
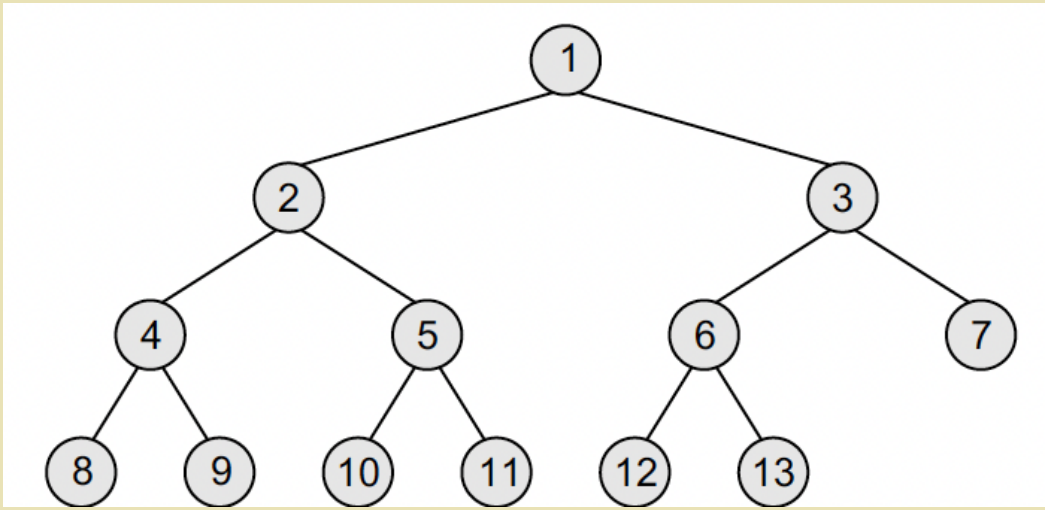


Figure from book Data Structures Using C, 2nd edition (Chapter 9, Figure 9.1), by Reema Thareja

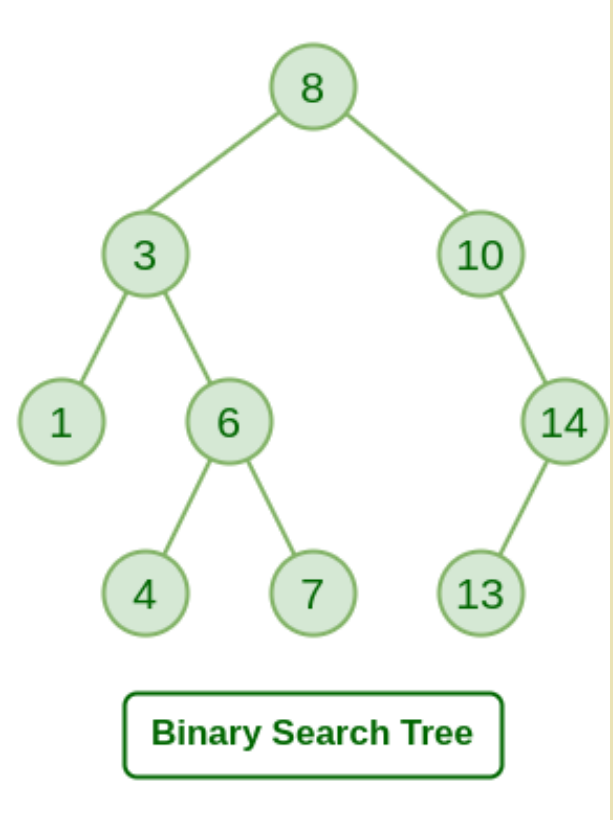
Figure from the zybook
7.1 Binary trees

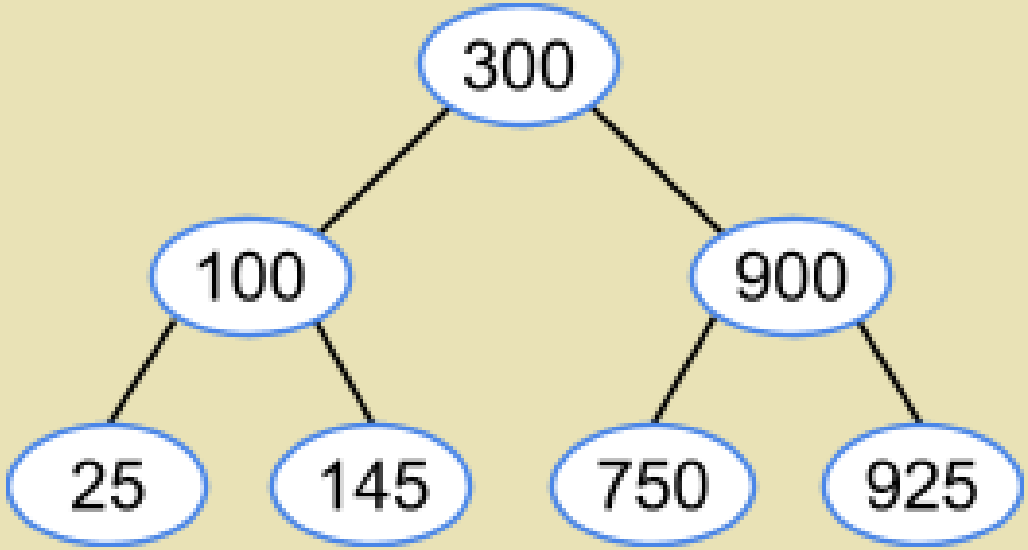
Binary Tree v.s. Complete Binary Tree





BST and Balanced BST



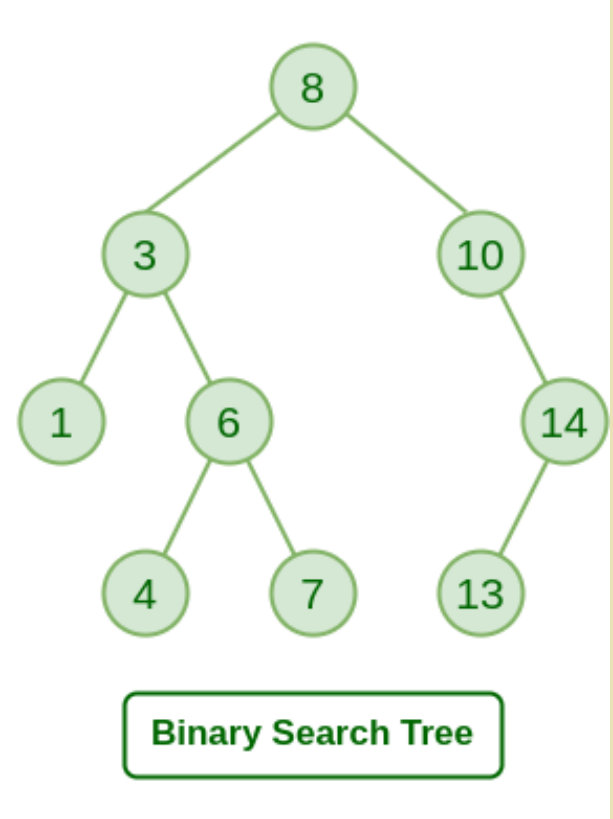


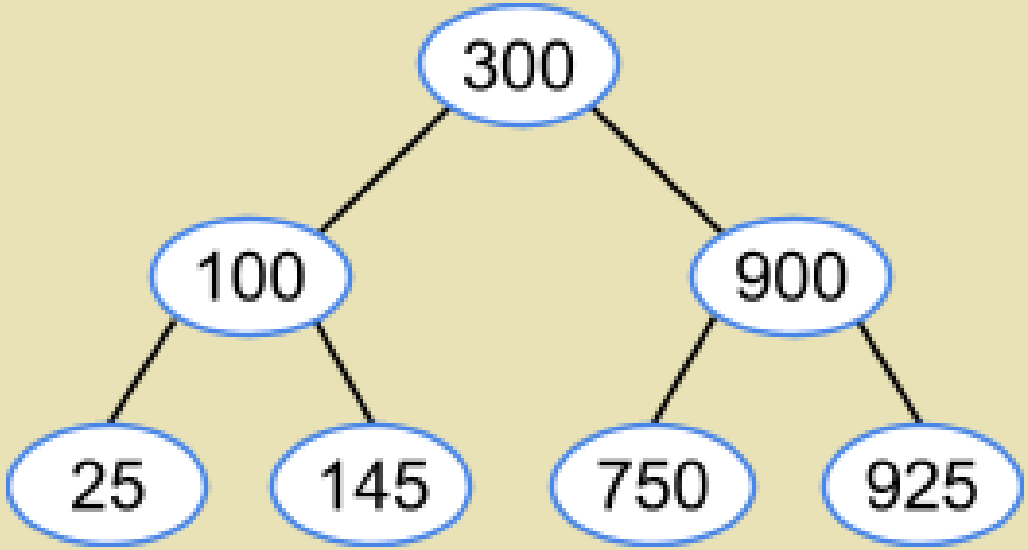
BST may not be a balanced tree

https://www.geeksforgeeks.org/difference-between-binary-search-tree-and-binary-heap/

Figure from 7.3 7.3 Binary search trees

BST and AVL Tree



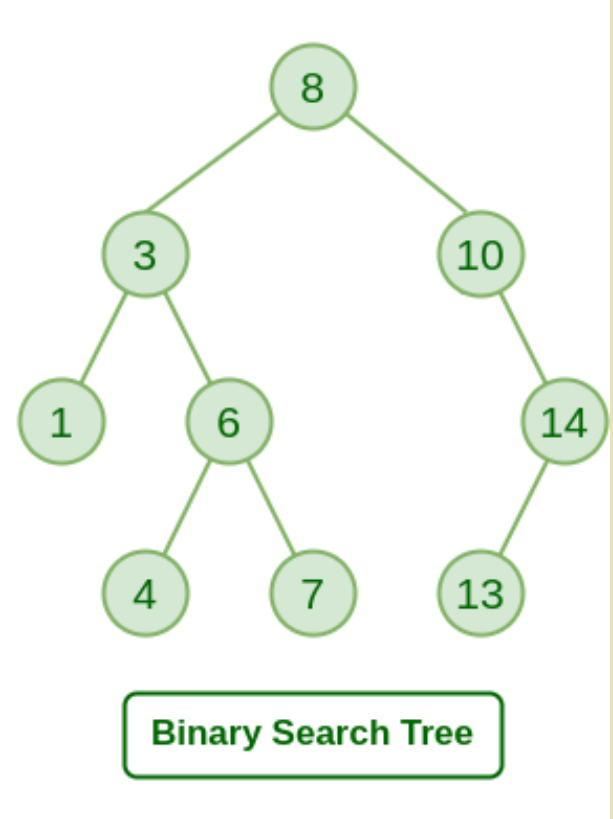


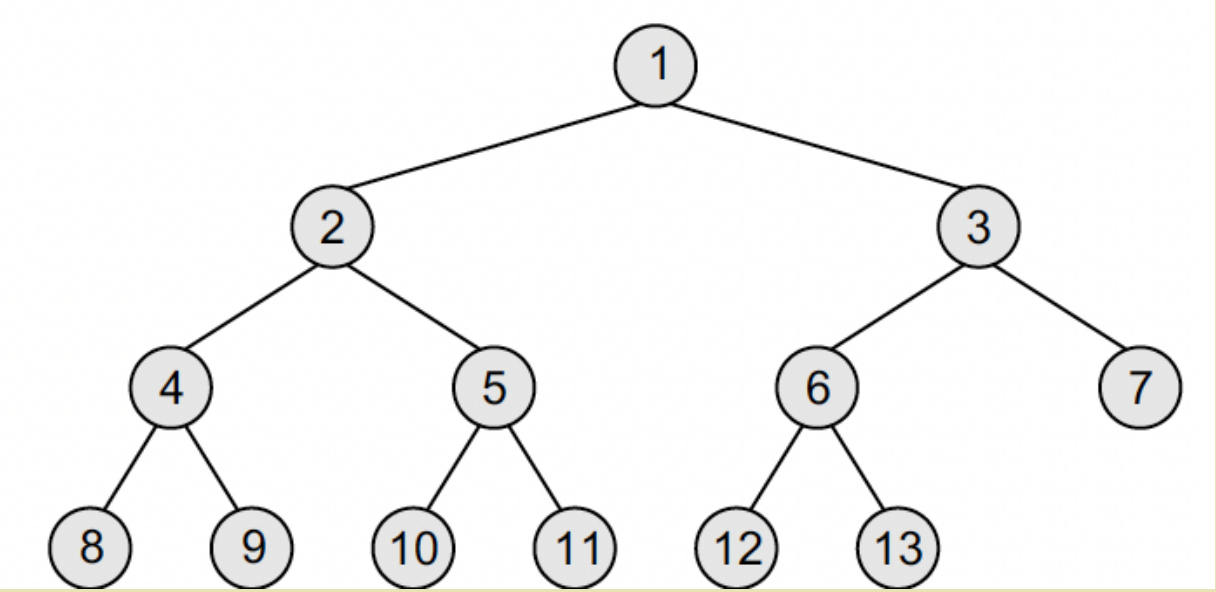
AVL Tree

https://www.geeksforgeeks.org/difference-between-binary-search-tree-and-binary-heap/

Figure from 7.3 7.3 Binary search trees

BST v.s. Heap



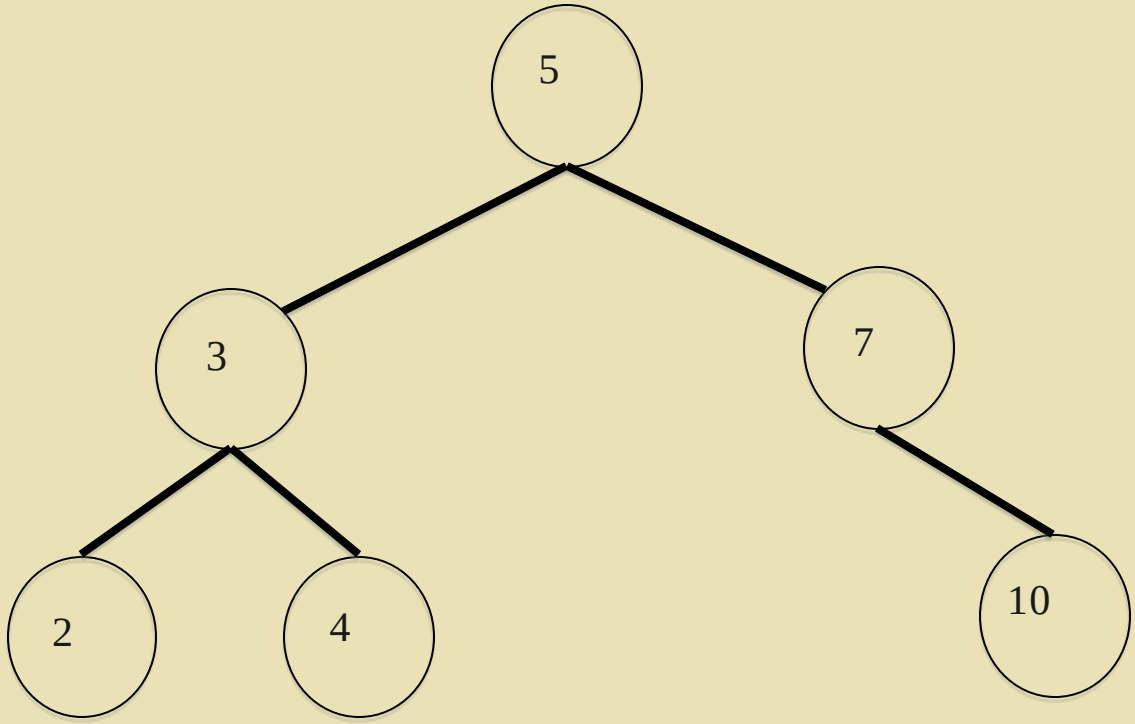


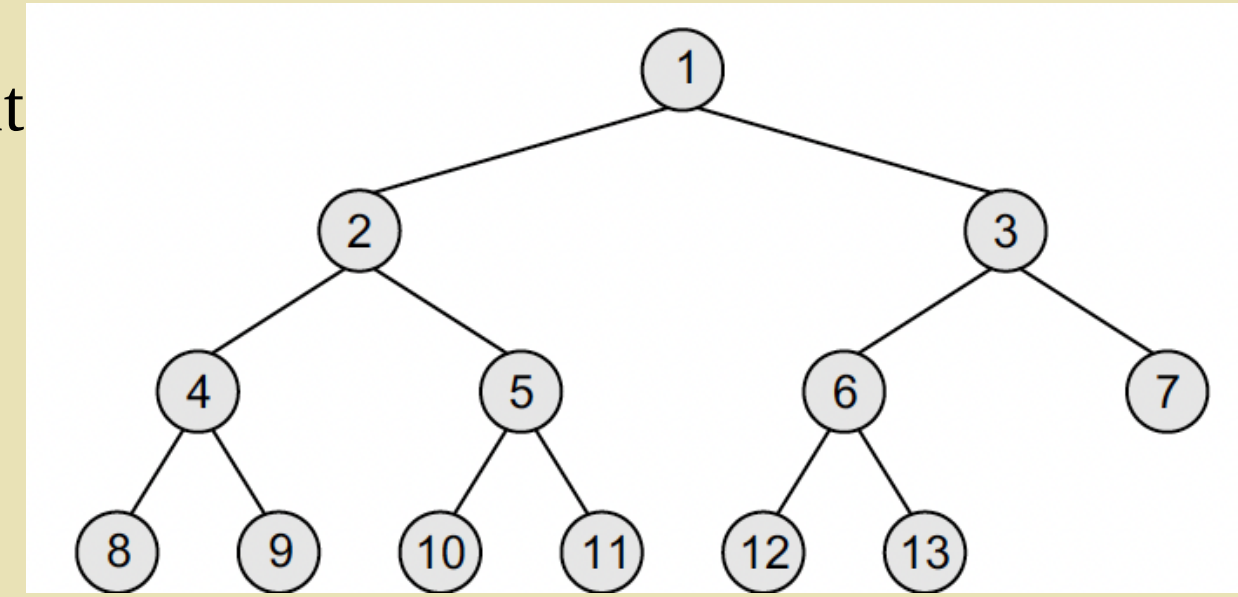
BST may not be a complete tree

https://www.geeksforgeeks.org/difference-between-binary-search-tree-and-binary-heap/

AVL Tree v.s. Heap

- Both balanced
- Heap is completed
- Max-heap: parent >= children
- AVL tree: left child< parent< right child

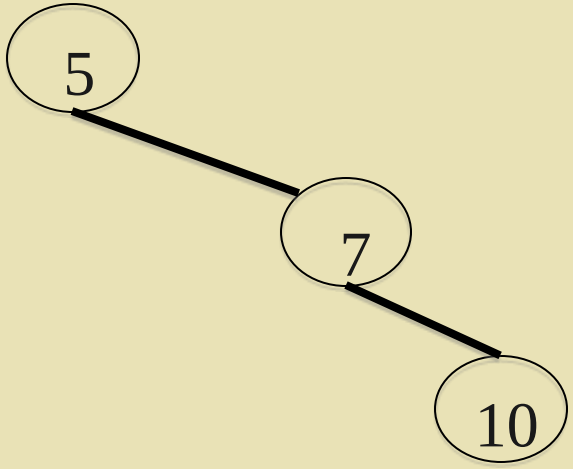




https://www.geeksforgeeks.org/difference-between-binary-search-tree-and-binary-heap/

Complexity

- BST: worst-case time complexity of insert/delete/search operations is O(h), where h is the height of the Binary Search Tree.
  - In the worst case, we may have to travel from the root to the deepest leaf node. The height of a skewed tree may become n and the time complexity of insertion operation may become O(n).



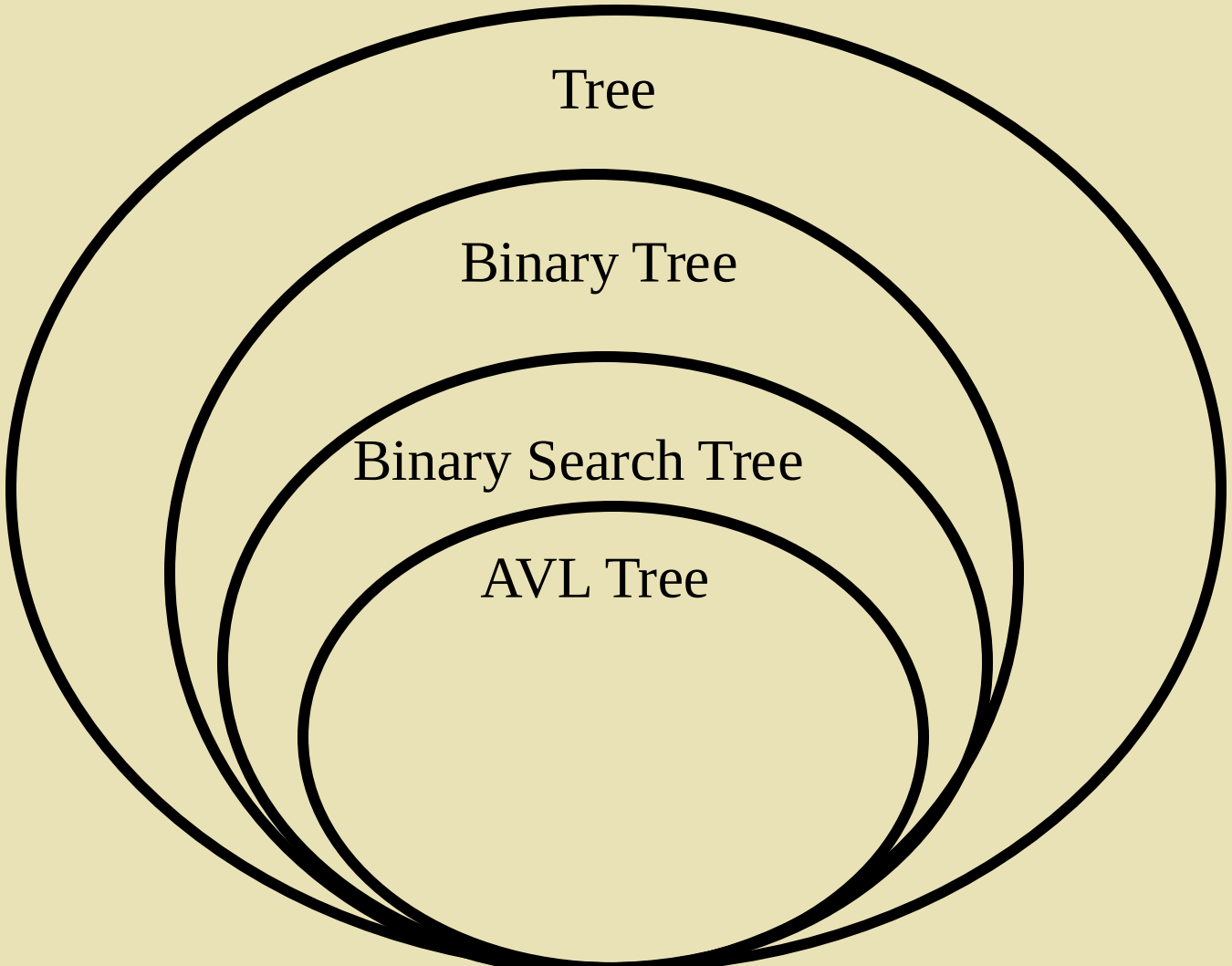
Complexity

- AVL tree: time complexity of insert/delete/search operations is O(log(n)) , where n is the number of nodes.
  - insertion only requires only one rebalance, deletion can require many but still no more than the tree's height
- Heap: time complexity of insert/delete/search operations is O(log(n)) , where n is the number of nodes.
  - O(number of swap) = O(height of tree)

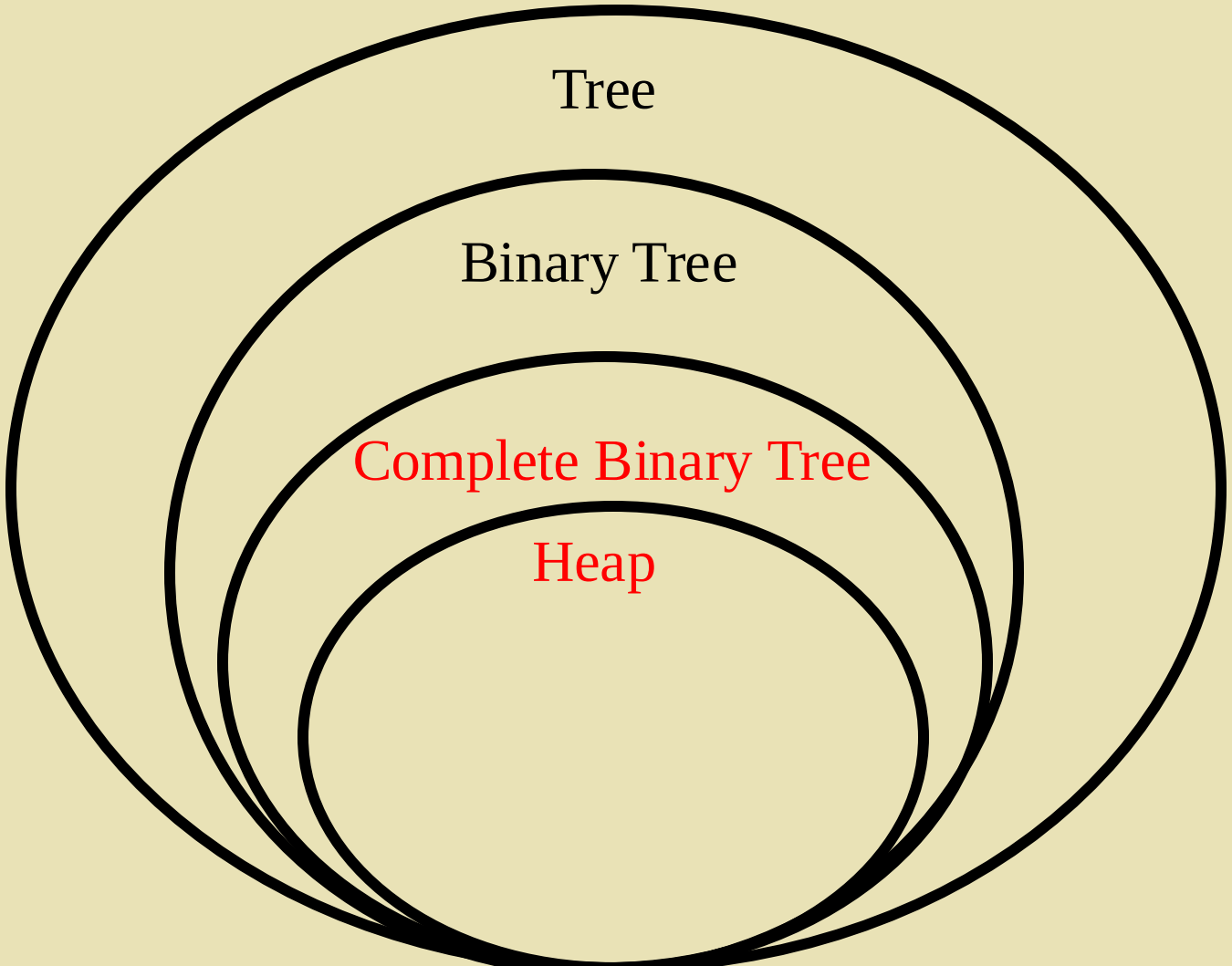
Heap	Tree
Usually, Heap is of two types, Max-Heap and Min-Heap.	A tree can be of various types for eg. binary Tree, BST(Binary Search tree), AVL tree, etc.
Heap is ordered.	Binary Tree is not ordered but BST is ordered.
Insert and remove will take O(log(N)) time in the worst case.	Insert and remove will take O(N) in the worst case in case the tree is skewed.
Finding Min/Max value in Heap is O(1) in the respective Min/Max heap.	Finding Min/Max value in BST is O(log(N)) and Binary Tree is O(N).
Heap can be built in linear time complexity.	BST: O(N * log(N)) and Binary Tree: O(N).

https://www.geeksforgeeks.org/comparison-between-heap-and-tree/

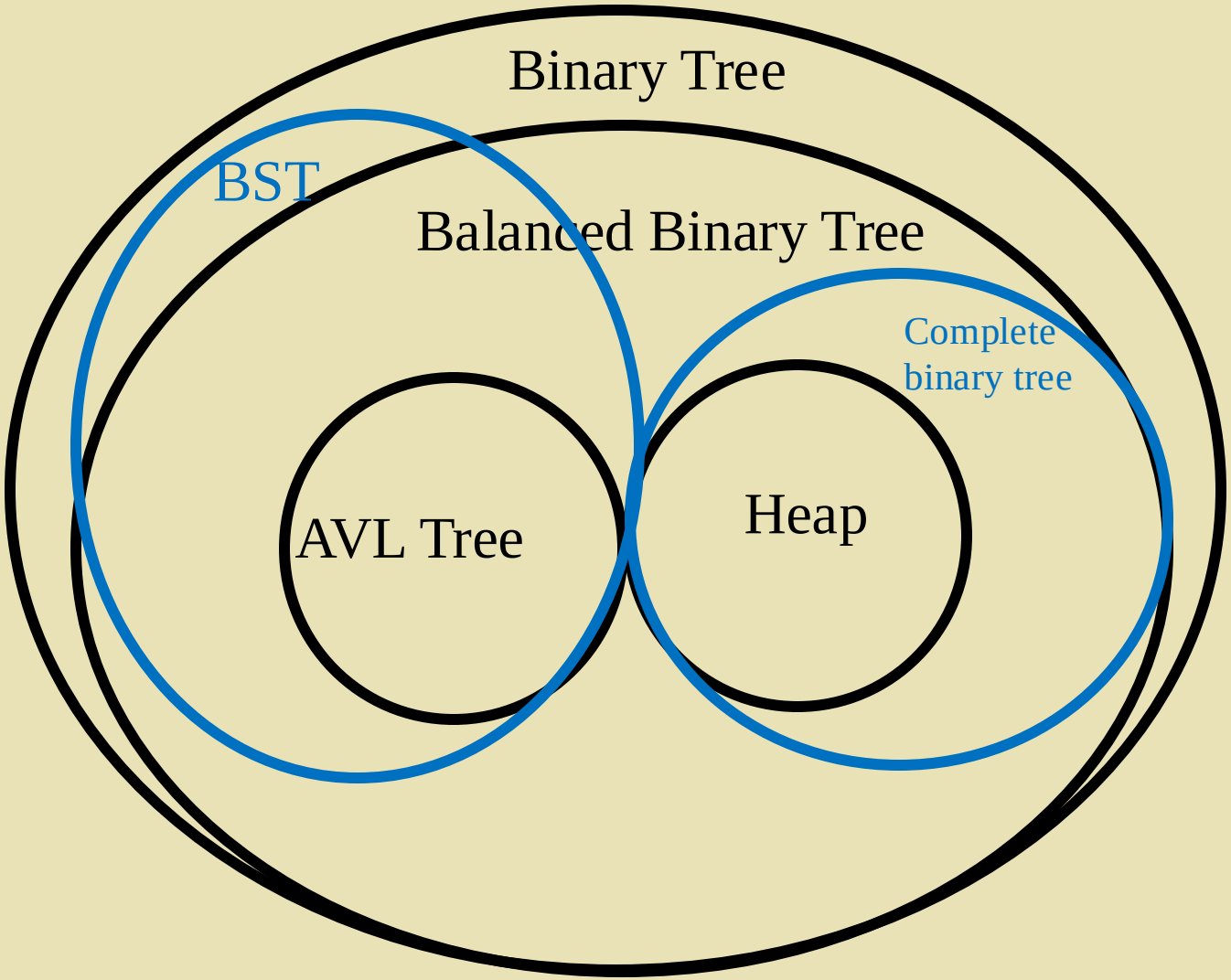
Comparisons



Comparisons



Comparisons



References and Useful Resources

- Binary search tree and heap differences https://www.geeksforgeeks.org/difference-between-binary-search-tree-and-binary-heap/
- Compare Heap and Tree https://www.geeksforgeeks.org/comparison-between-heap-and-tree/

That’s about this lecture!