CS106 Data Structure & Java Fundamentals

BitTiger丨来自硅谷的终身学习平台

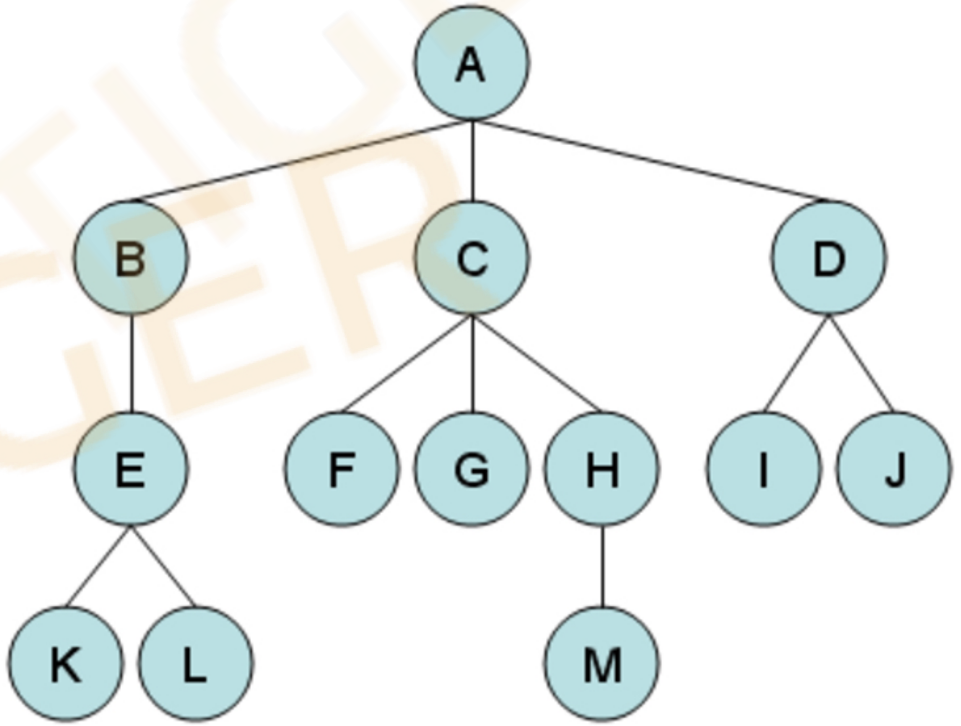
Claire & Bob

Chapter 3 - Data Structures

- ArrayList
- LinkedList
- HashTable/Map/Set
- Tree
- Stack/Queue
- Heap

Tree

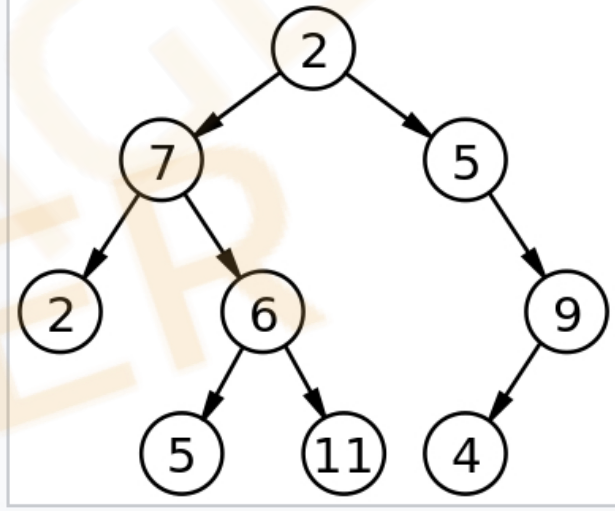
Nodes and edges without having cycles



https://www.cpp.edu/~ftang/courses/CS241/notes/trees.htm

Binary Tree

Every node has 0-2 child node(s)

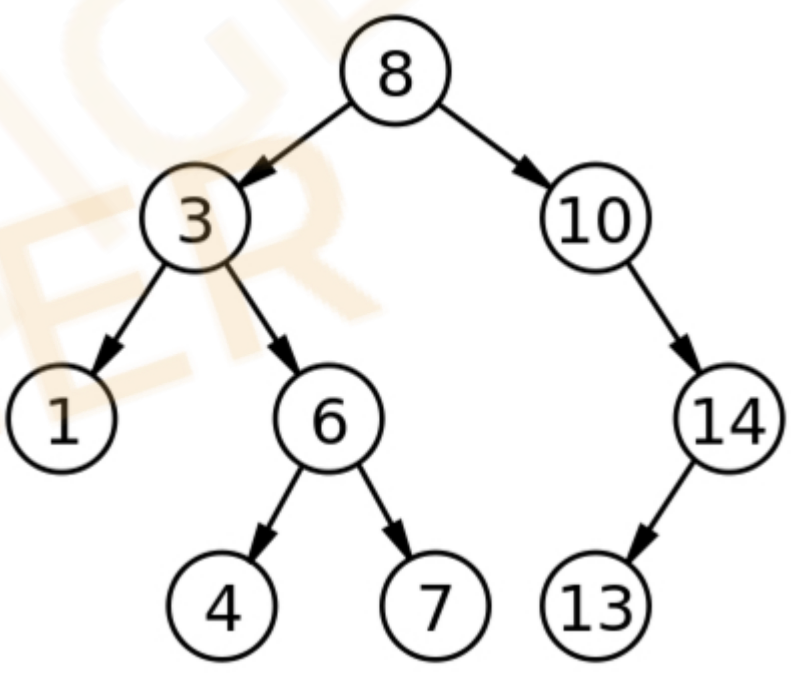


A labeled binary tree of size 9 and height 3, with a root node whose value is 2. The above tree is unbalanced and not sorted.

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

Binary Search Tree (Ordered/Sorted Binary Tree)

BST is a binary tree that fulfills a specific ordering property -
On any subtree:
left nodes < root node < right nodes

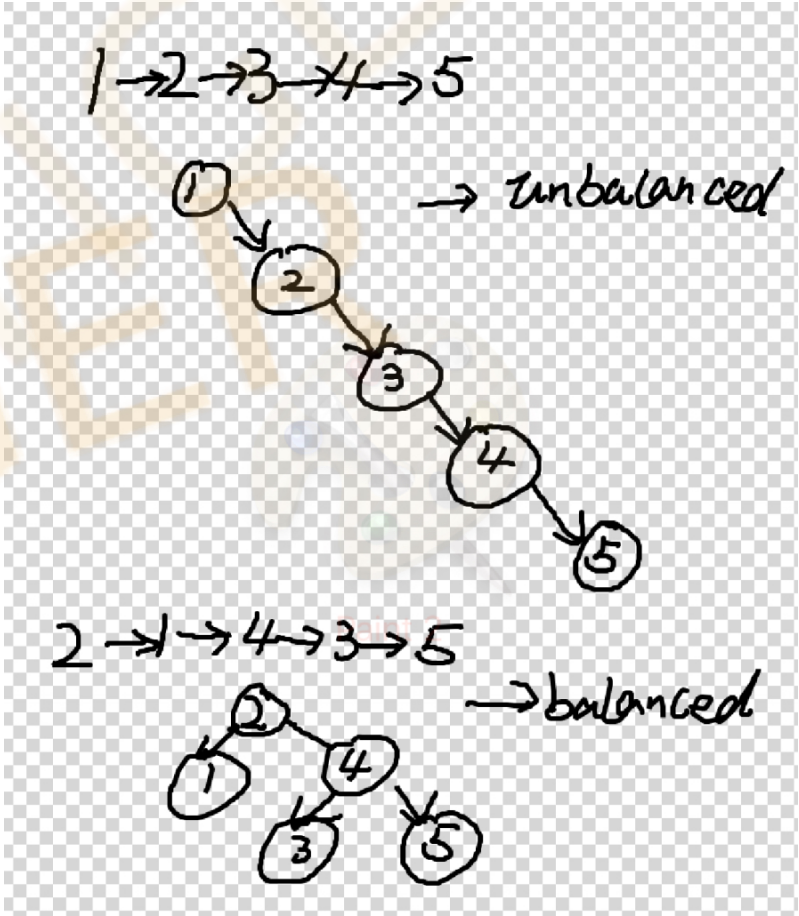


https://en.wikipedia.org/wiki/Binary_search_tree

Balanced v.s. Unbalanced BST

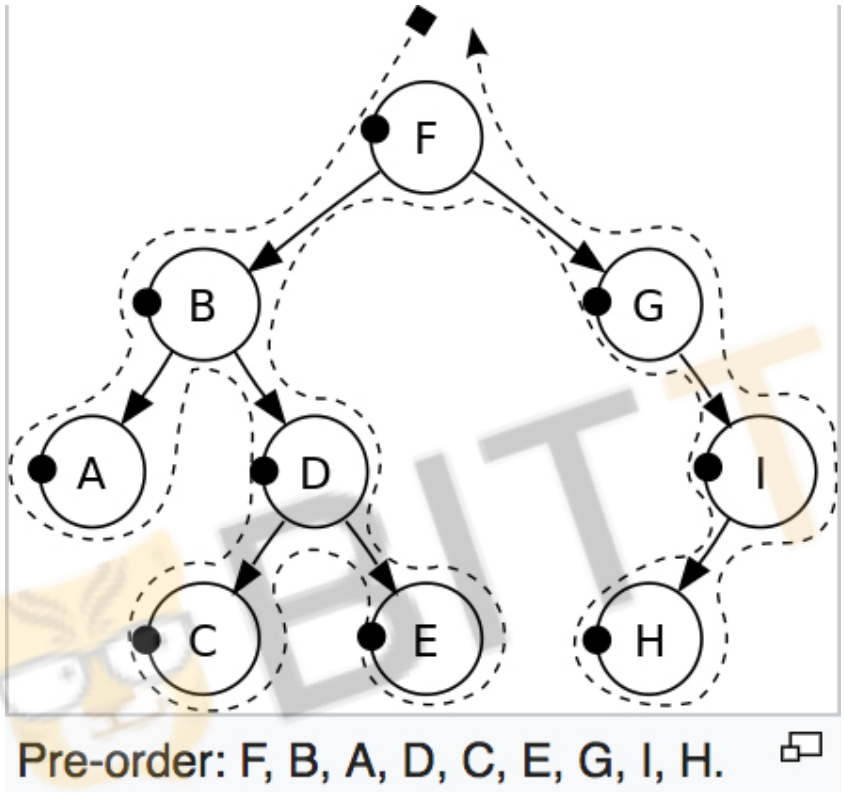
	Average/Balanced	Worst/Unbalanced
Space	O(n)	O(n)
Search	O(logN)	O(n)
Insert	O(logN)	O(n)
Delete	O(logN)	O(n)

Assume Balanced BST in interview

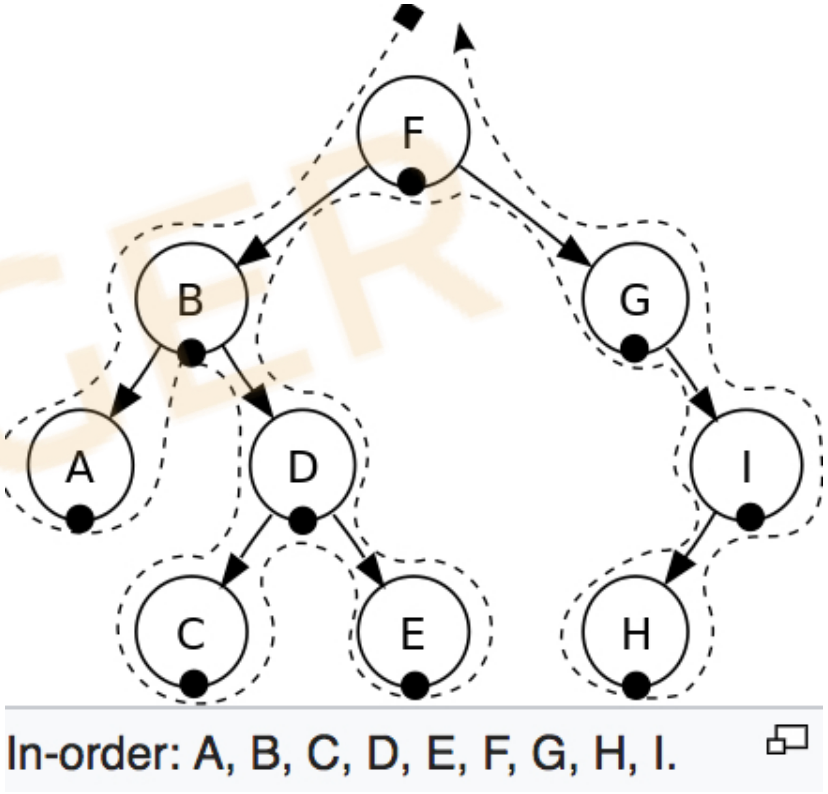


Tree Traversal

Root → left → right



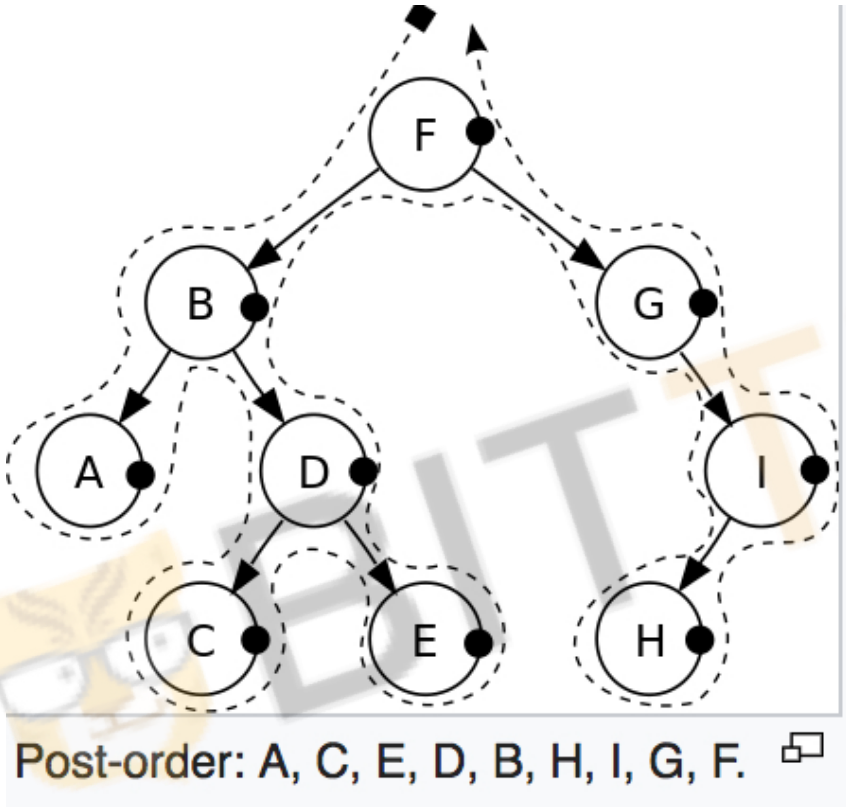
Left → root → right



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

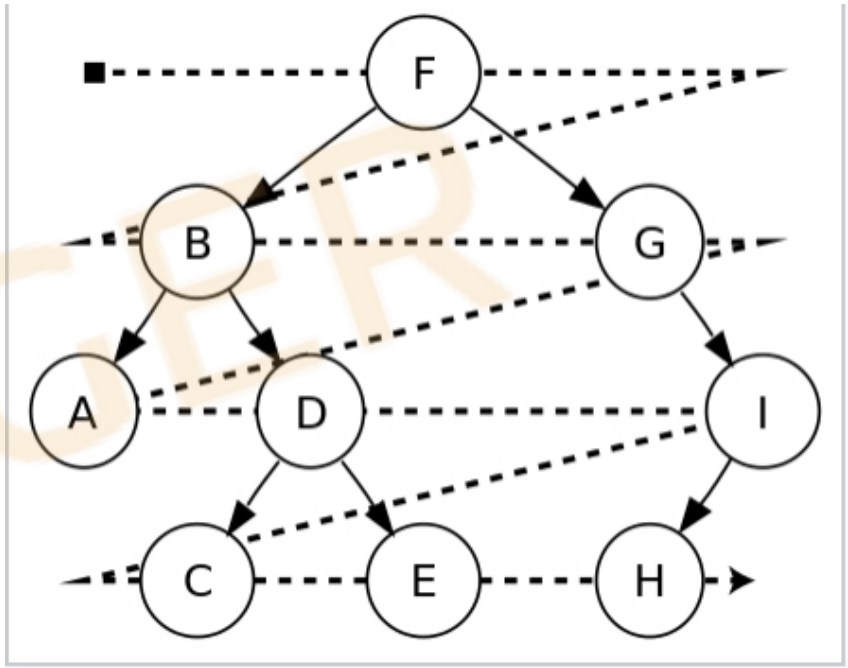
Tree Traversal

Left → right → root



Post-order: A, C, E, D, B, H, I, G, F.

Breadth-first search (BFS)



Level-order: F, B, G, A, D, I, C, E, H.

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

Learning by Doing

Let’s implement a BST

Chapter 3 - Data Structures

- ArrayList
- LinkedList
- HashTable/Map/Set
- Tree
- Stack/Queue
- Heap

Stack v.s. Queue

Stack: LIFO - last in first out
Queue: FIFO - first in first out

Stack in Java

- Create a stack - Deque<Integer> stack = new ArrayDeque<Integer>();
- Peek, pop, push, and more (Stack interface) - https://docs.oracle.com/javase/7/docs/api/java/util/Stack.html
- https://docs.oracle.com/javase/7/docs/api/java/util/Deque.html

Queue in Java

- Create a queue - Queue<Integer> queue = new LinkedList<Integer>();
- Add, remove, offer, poll and more - https://docs.oracle.com/javase/7/docs/api/java/util/Queue.html

Learning by Doing

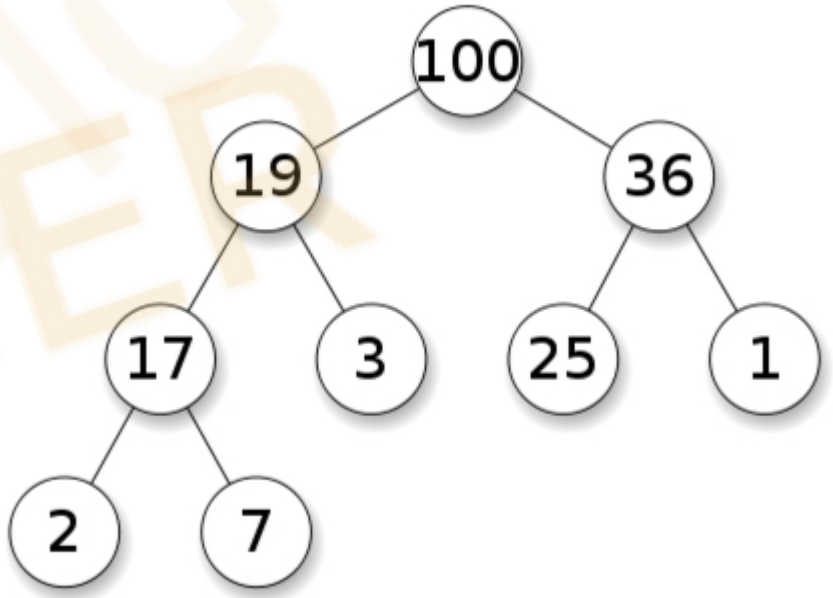
Let’s implement Stack/Queue by ourselves...

Chapter 3 - Data Structures

- ArrayList
- LinkedList
- HashTable/Map/Set
- Tree
- Stack/Queue
- Heap

Heap

- **Max heap**: parent >= children
  (Note that 19 is not greater than 25)
- **Min heap**: parent <= children



https://en.wikipedia.org/wiki/Heap_(data_structure)

Min Heap: Insert

- Add to the next empty spot (looking top to bottom, left to right)
- **Bubble up** to the right place
- **Worst** time complexity: O(logN)
- On **average**, the newly inserted element does not travel very far up the tree - O(1)
- Proof: http://wcipeg.com/wiki/Binary_heap

Min Heap: Delete

- Swap the root with the last added element
- Bubble down - Compare with left and right child, swap with the smaller one
- Time Complexity - O(logN)

Heap Implementation

- Implemented as Tree? - Overhead of node class

Heap Implementation

- Implemented as Tree? - Overhead of node class
- Implemented by Array? - How to get to left and right child ?

Heap Implementation

- Implemented as Tree? - Overhead of node class
- Implemented by Array? - How to get left and right child？
- Parent index: (i - 2)/2
- Left child index: i * 2 + 1
- Right child index: i * 2 + 2

Learning by Doing

Implement a heap..

Chapter 4 - Algorithms

- Binary Search
- Sort (quickSort/mergeSort)
- DFS & BFS
- Dynamic Programming

Chapter 4 - Algorithms

- Binary Search
- Sort (quickSort/mergeSort)
- DFS & BFS
- Dynamic Programming

Linear Search vs Binary Search

Search target: 51

99	56	47	8	34	69	55	72	51

Linear Search

Search target: 51

Time complexity: n/2 -> O(n)

99	56	47	8	34	69	55	72	51

Binary Search

Search target: 51

To do a binary search, the array must be **SORTED**!

99	56	47	8	34	69	55	72	51

Binary Search

Search target: 51

Middle index = (start + end) / 2 = (0 + 8)/2 = 4

8	34	47	51	55	56	69	72	99

Binary Search

Search target: 51

Middle index = (start + end) / 2 = (0 + 8)/2 = 4

8	34	47	51	55

Binary Search

Search target: 51

Middle index = (start + end) / 2 = (0 + 4)/2 = 2

8	34	47	51	55

Binary Search

Search target: 51

Middle index = (start + end) / 2 = (0 + 4)/2 = 2

47	51	55

Binary Search

Search target: 51

Middle index = (start + end) / 2 = (0 + 2)/2 = 1

47	51	55

Binary Search

Search target: 51

Time complexity: **O(logN)**

Learning by Doing

Coding Challenge - FirstBadVersion

Learning by Doing

Let’s get to advanced level…

MediumInTwoSortedArray

Search

Linear Search: O(n)

Binary Search: O(logN)