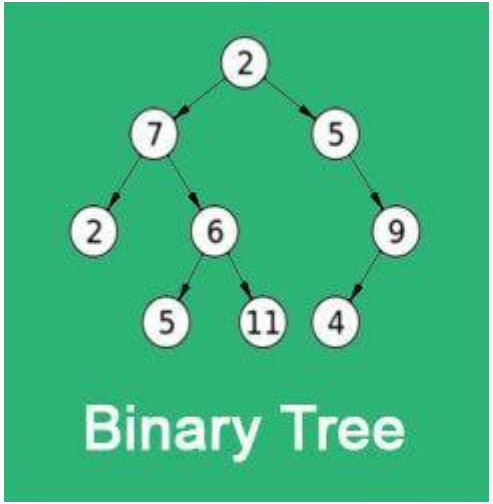
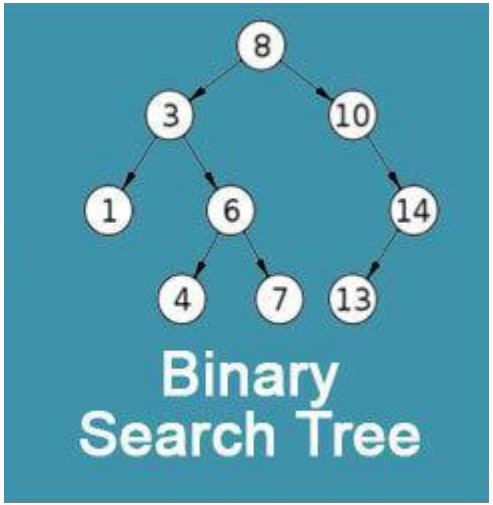
Welcome! CS 225

Binary Search Trees

What is a BST?

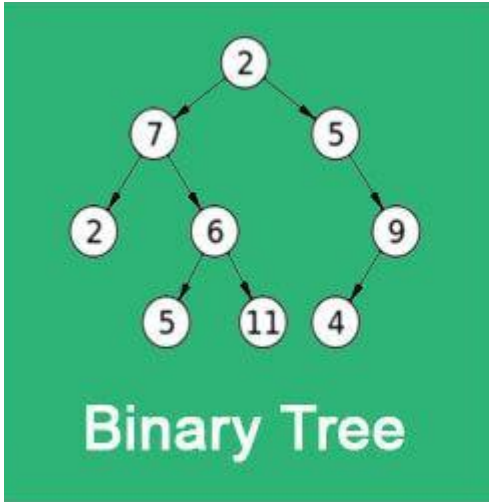
Why a BST?



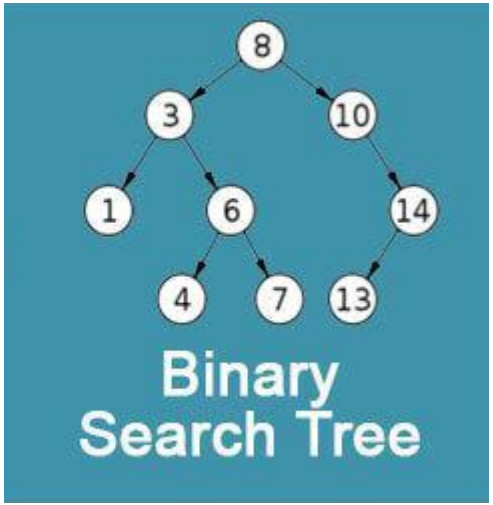


What is a BST?

- Each node has at most 2 children
- Left child has smaller values
- Right child has larger values
- Children are also BSTs
- No repeating nodes

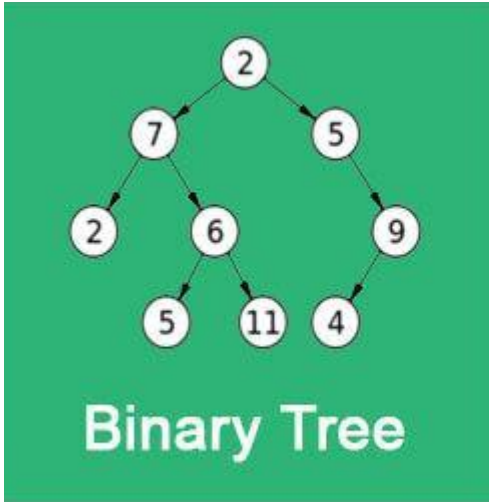


Why a BST?



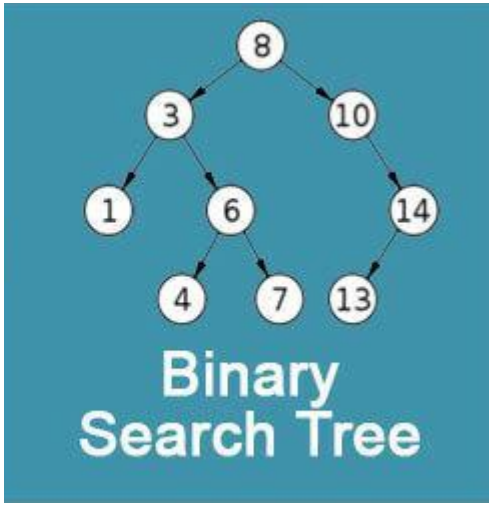
What is a BST?

- Each node has at most 2 children
- Left child has smaller values
- Right child has larger values
- Children are also BSTs
- No repeating nodes



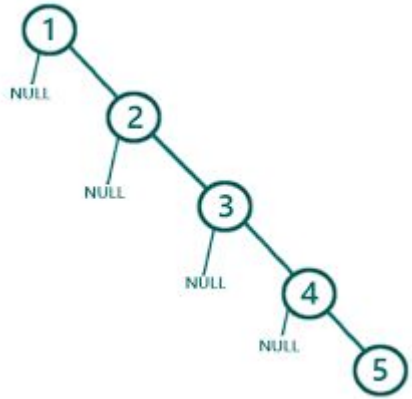
Why a BST?

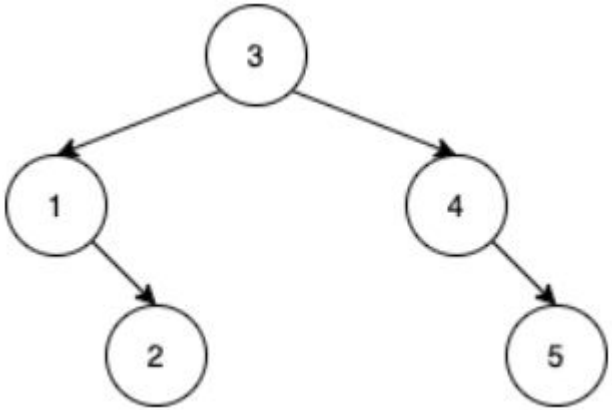
- Insertion, finding and removing are log(n) time in most cases.
- Traversal gives a sorted list



lab_bst (Binary Search Trees)

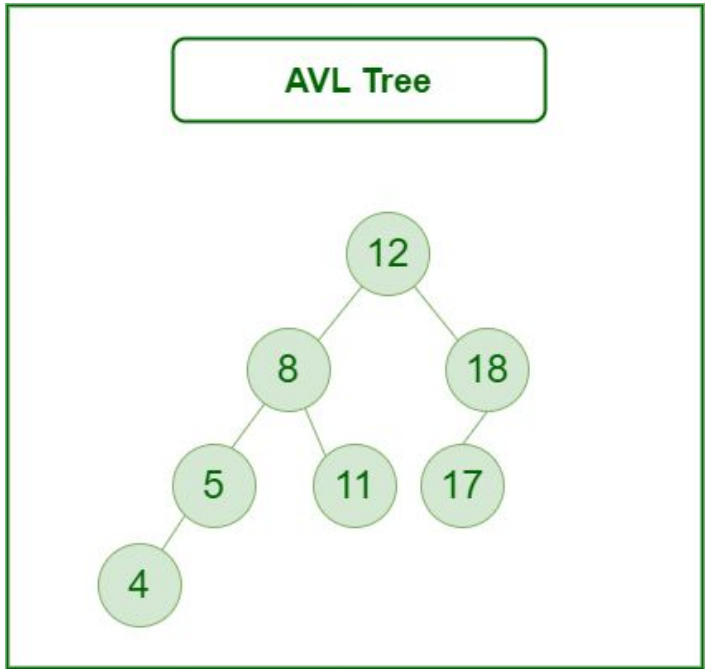
- Worst case Vs Expected case

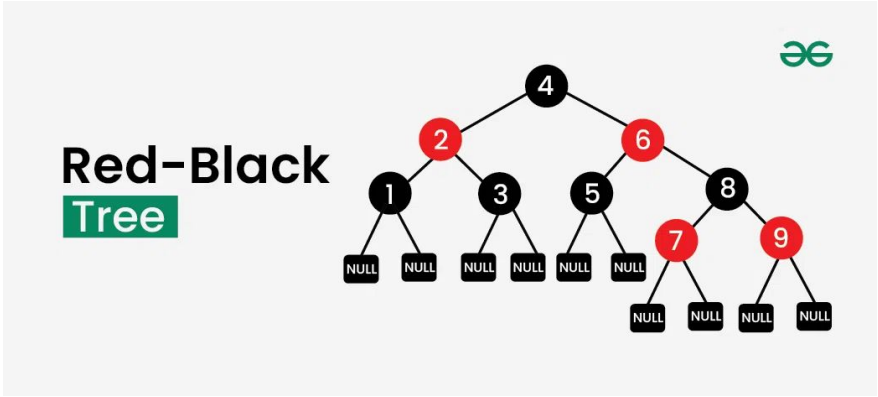


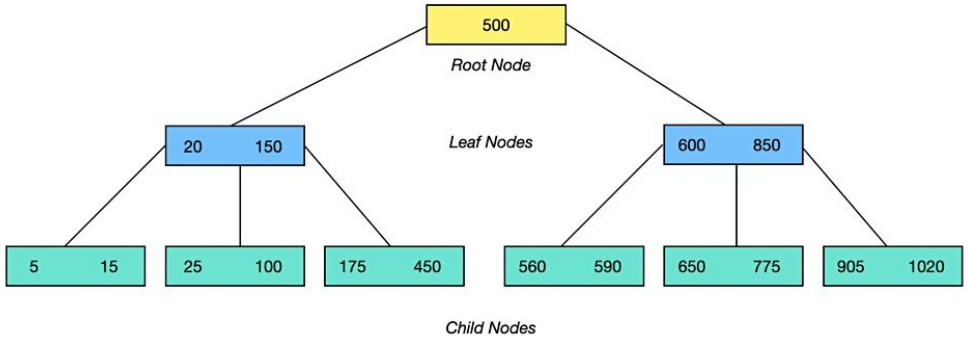


How could this be improved?

Self balancing trees



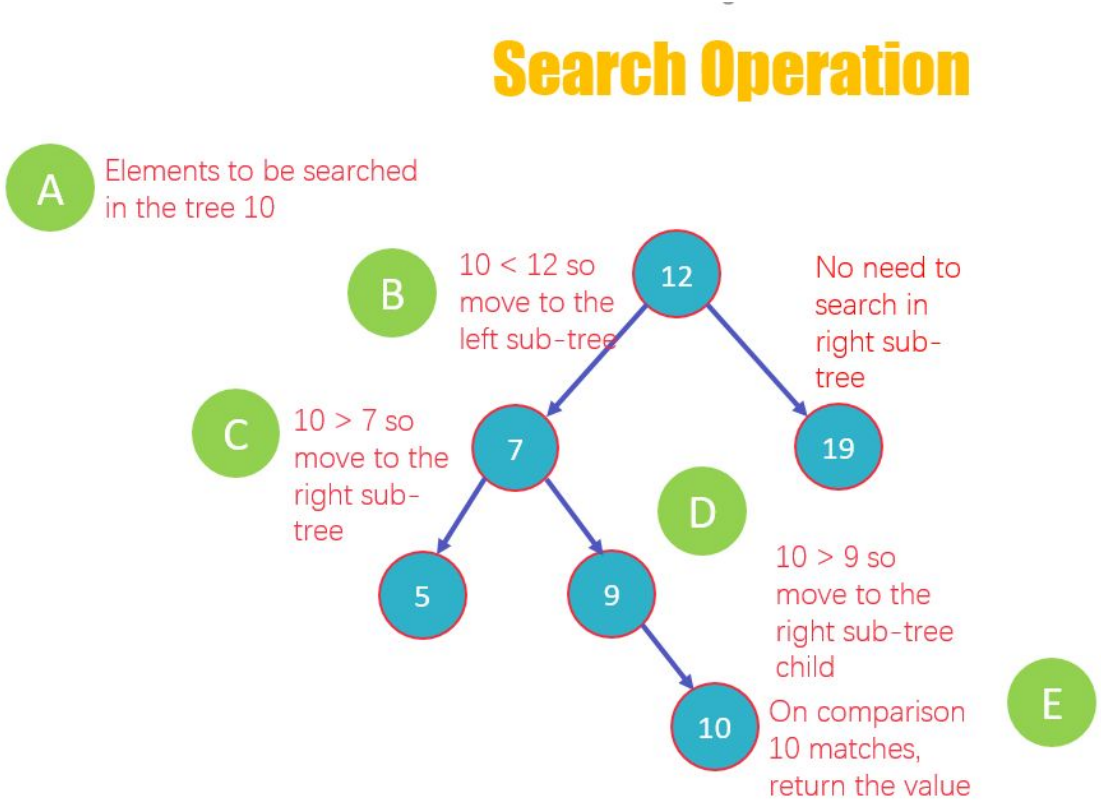




B Trees

Basic Functionality

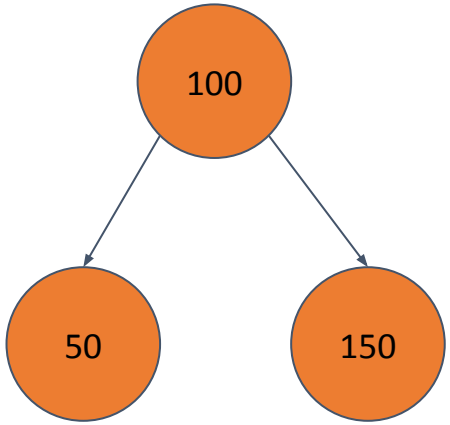
Find



- Tip: Return a reference to the pointer instead of the pointer. This ensures that when you use this function in insert and remove, the returned reference can be edited directly

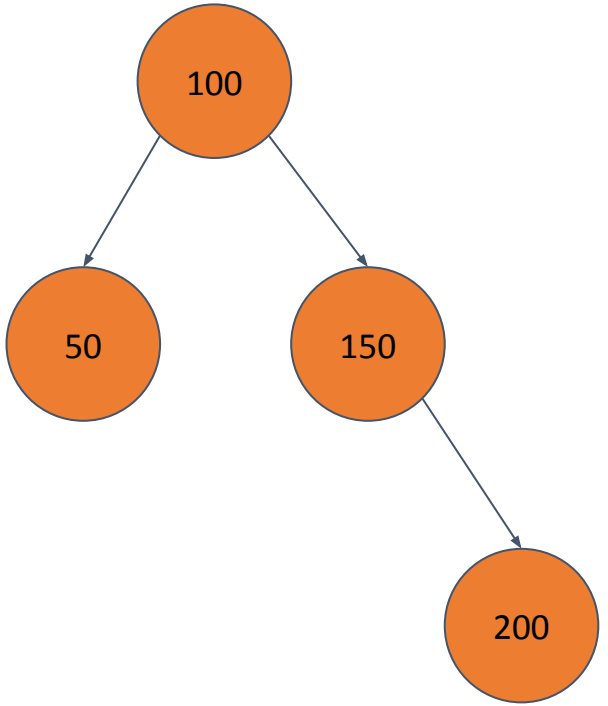
Insert

- Where would 200 be inserted?



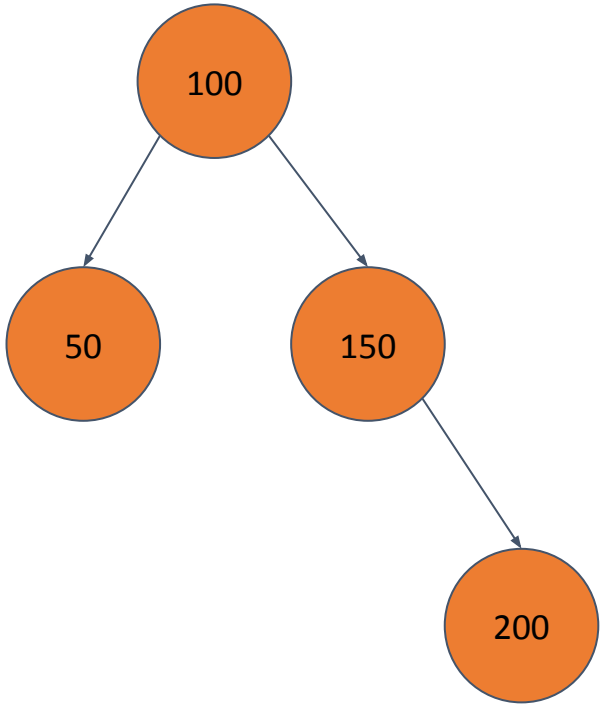
Insert

- Where would 200 be inserted?



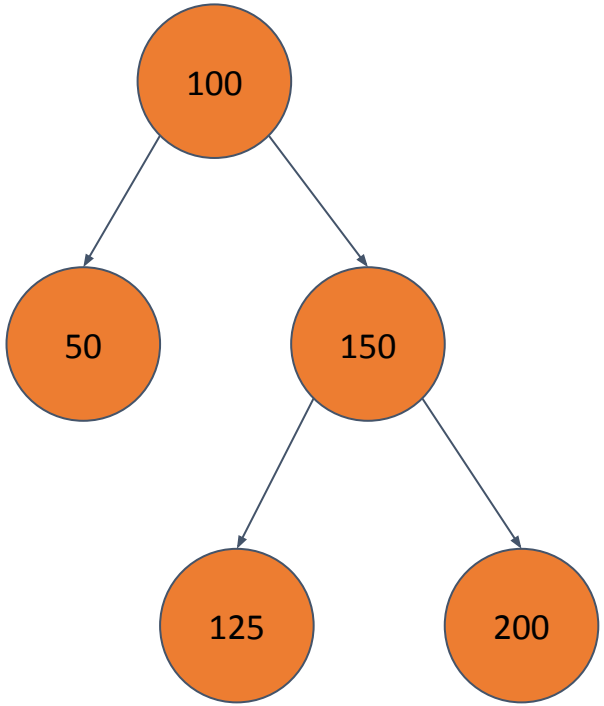
Insert

- Where would 125 be inserted?



Insert

- Where would 125 be inserted?

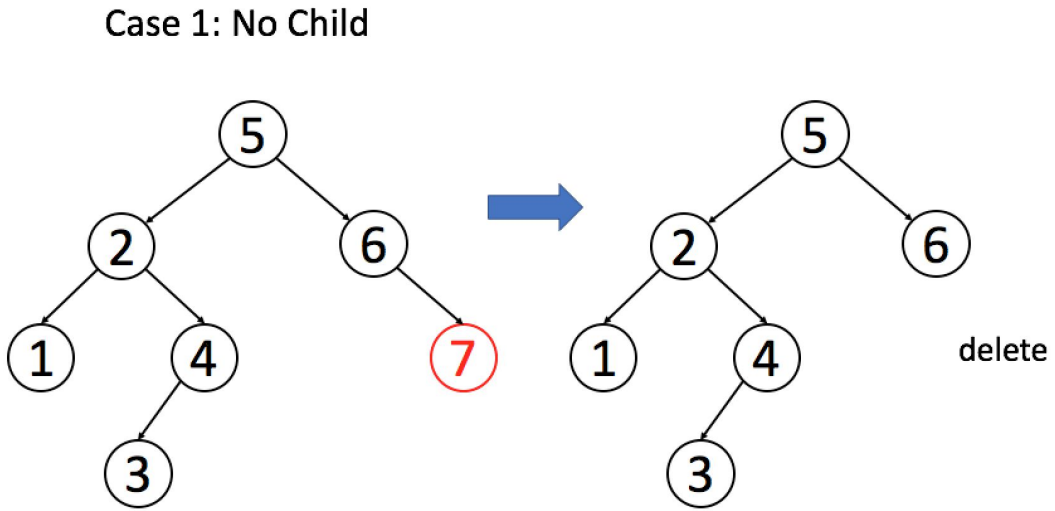


Remove

Case 1 (No child):

1. Find the target node (passed as a reference to the parent's pointer)
2. Delete the node
3. Set child pointer of parent to NULL

Delete 7

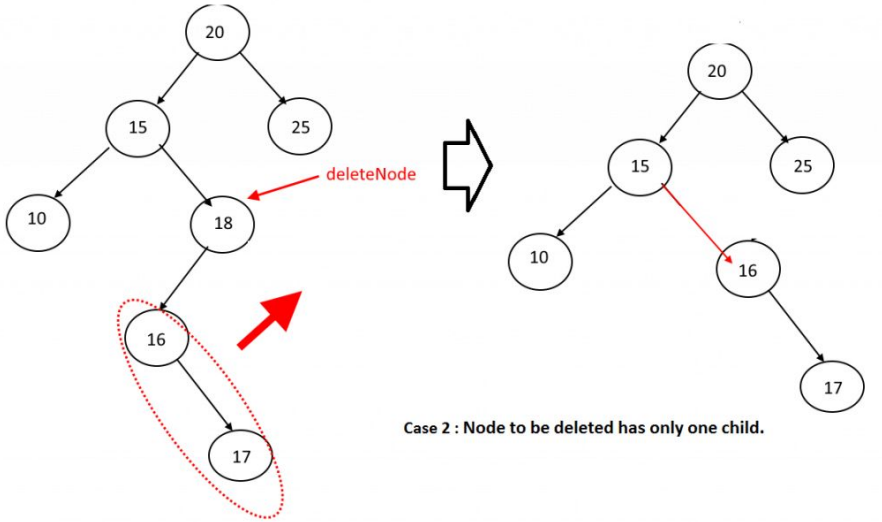


Remove

Case 2 (1 child):

1. Find the target node (passed as a reference to the parent's pointer)
2. Make a temporary pointer to the target
3. Set the parent's node to the target's child
4. Delete the target

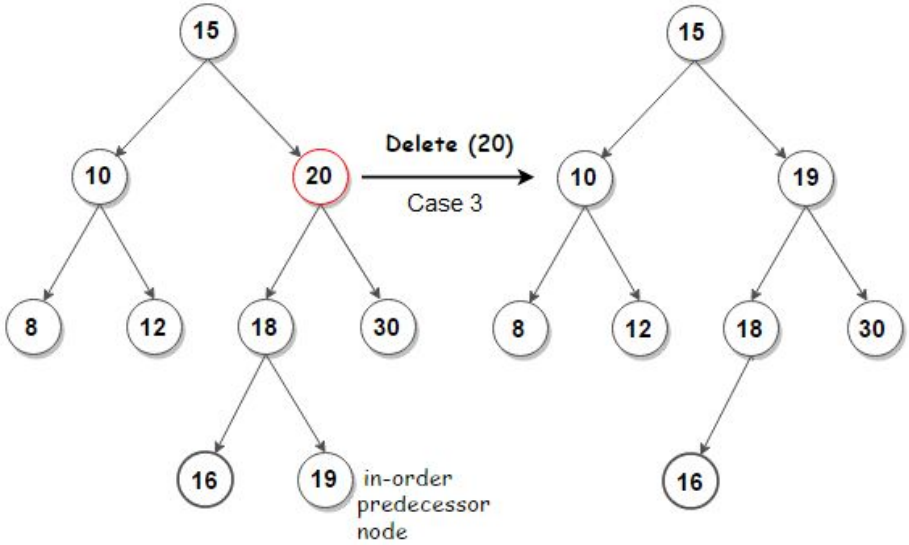
Delete 18



Remove

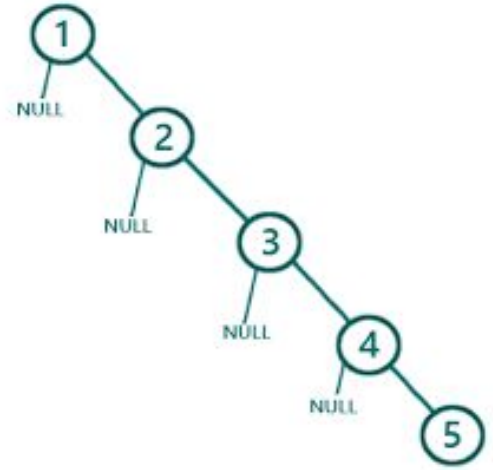
Case 3 (2 children):

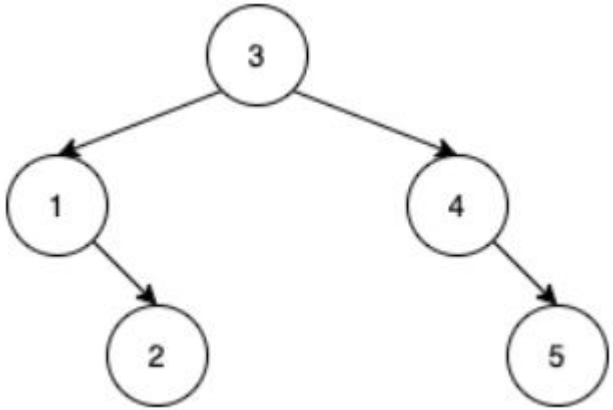
1. Find the target node (passed as a reference to the parent's pointer)
2. Find the target's In Order Predecessor (IOP)
3. Swap the target with the IOP
4. Recursively call on the target's new location



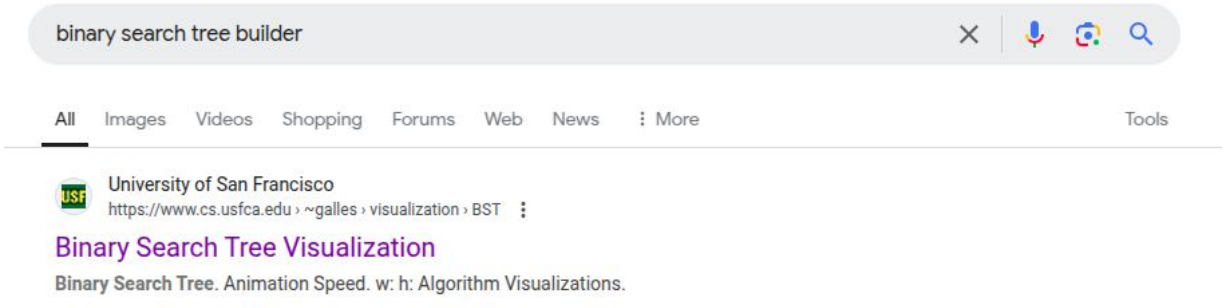
Worst case Vs Average case

- What is the time complexity of find, insert and remove in a BST?
  - O(h)
- What is h in terms of n?
  - There is no guarantee for that; it depends on the order in which the elements were inserted into the BST
- Consider a BST with numbers 1 to 5, but inserted in the following order [1, 2, 3, 4, 5]. What is the tree you get?
- Now consider the order [3, 1, 4, 2, 5]. What is the tree you get?
- Clearly, height of the tree depends on order of insertion





Helpful Resources



Case 1 (No child):

1. Find the target node
2. Delete the node
3. Set child pointer of parent to NULL

Case 2 (1 child):

1. Find the target node
2. Make a temporary pointer to the target
3. Set the parent's node to the target's child
4. Delete the target

Case 3 (2 children):

1. Find the target node
2. Find the target's In Order Predecessor (IOP)
3. Swap the target with the IOP
4. Recursively call on the target's new location