Lecture Notes – Binary Search Tree

After the example of using a binary heap to implement a priority queue (find min or find max), how can we improve so that we can have a tree structure that facilitates searching in *O(log n)* time?

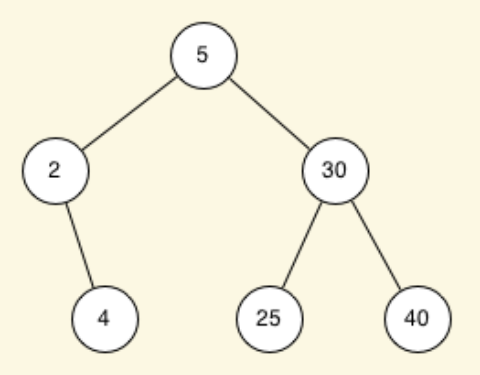
A binary search tree is a binary tree with the following properties:

1. all items in the left subtree is less than the root
2. all items in the right subtree is greater than or equal to the root
3. each subtree is itself a binary search tree

Adding / building a Binary Search Tree (BST):

```
If tree is empty
        Add newNode as root
else
        if newNode's key < root's key
                recursively call this method with leftsubtree
        else
                recursively call this method with rightsubtree
```

Given the random sequence: 5, 30, 2, 40, 25, 4
Let's construct the BST together using the whiteboard. The result will look like the following tree.



Consider adding 19 to an existing BST.
Figure 2 illustrates the process for inserting a new node into a binary search tree. The lightly shaded nodes indicate the nodes that were visited during the insertion process.

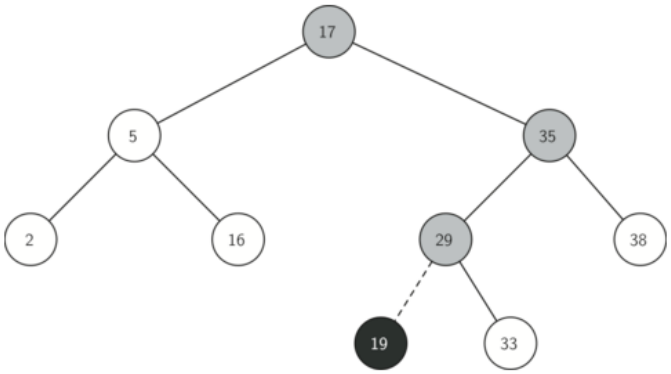


Figure 2: Inserting a Node with Key = 19

For a near balanced BST, search efficiency approaches *O(log n).*

Where is the smallest value in the tree?
Where is the largest value in the tree?
What does an inorder traversal (L-RT-R) yield?

Deleting from a BST:

If the node to be deleted is a leaf, simply adjust the pointer in its parent node to None.

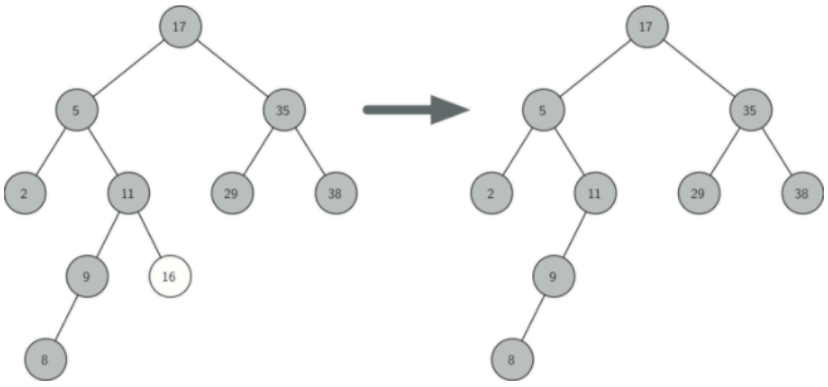


Figure 3: Deleting Node 16, a Node without Children

If the node to be deleted has one subtree, pull the subtree up to take the place of the deleted node.

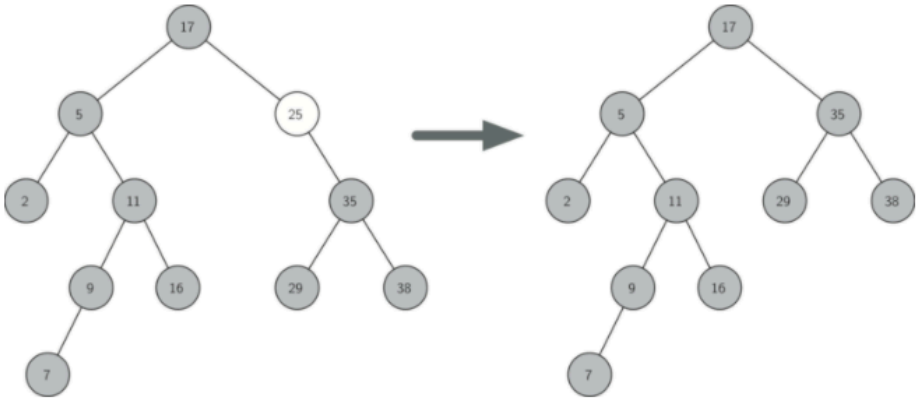


Figure 4: Deleting Node 25, a Node That Has a Single Child

If the node to be deleted has both left and right subtree, find its inorder successor, copy successor's value into node to be deleted, then delete successor from tree. It is also possible to use the inorder predecessor. Same logic follows.
Note that whether the inorder successor or predecessor is selected, this selected replacement node will always be either a leaf or node with only one child.

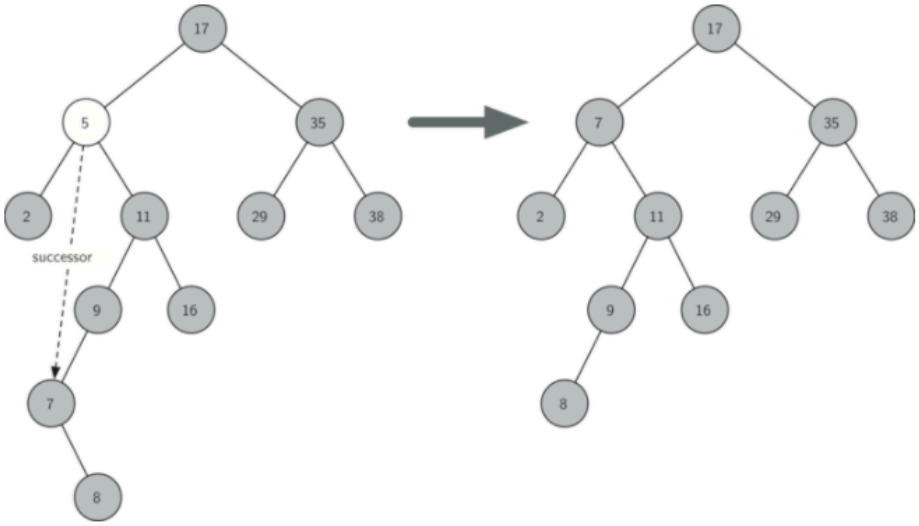
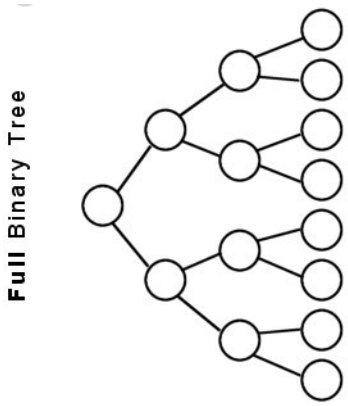


Figure 5: Deleting Node 5, a Node with Two Children

* All image from etext 7.13

Binary Search Tree Analysis

Consider the binary search algorithm where each iteration goes through the list, eliminates half the batch, when we view the list turned 90 degrees, the operation resembles a binary tree structure.



In a binary search tree, the more balanced / complete / compact the tree is, the closer it approaches *O(log n)* efficiency.

The more skewed the binary search tree is, the more inefficient it is.

What is the worst case?
The shape of the BST depends on how the nodes were inserted into the tree, i.e. it depends on the order they arrive and are added into the tree
Consider if the input data are arriving at a sorted order. The resulting tree becomes a linear structure, and is operating at *O(n)* efficiency.

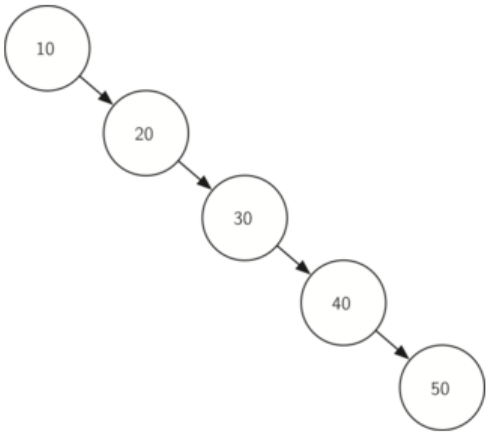


Figure 6: A skewed binary search tree would give poor performance

image from e-text section 7.14

This leads to the next topic of tree balancing.