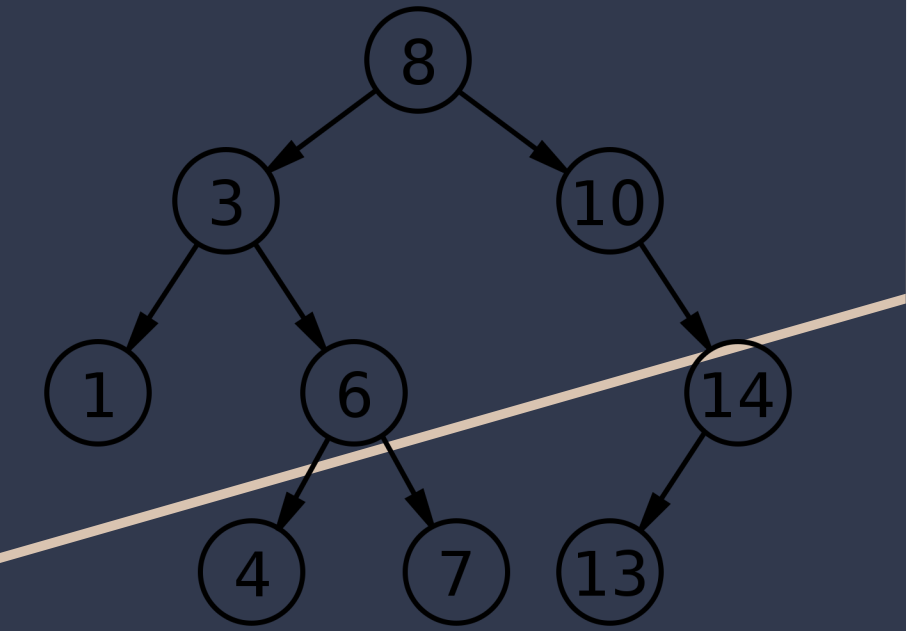
CLC 4: Trees and Heaps

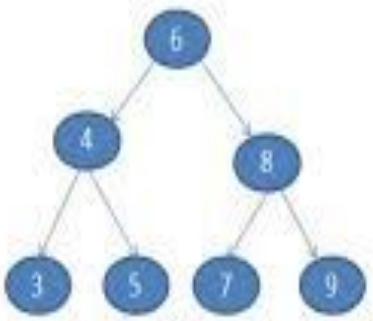
Adam Ringwell and Ryan Scott

What are Binary Search Trees?



- Binary Search Trees are a sorted list of numbers that required an initial root.
- The root of a tree is the first value that has any input pass through it to see where it should end up.
- If a new input is less than the root, it goes left, but if it is greater than or equal to, it goes right
- From here, the data point will pass through each child node to see whether it is less than or greater than or equal to until it becomes a leaf, or a data point with no children

Traversing Through a Tree



Preorder: Root-Left-Right

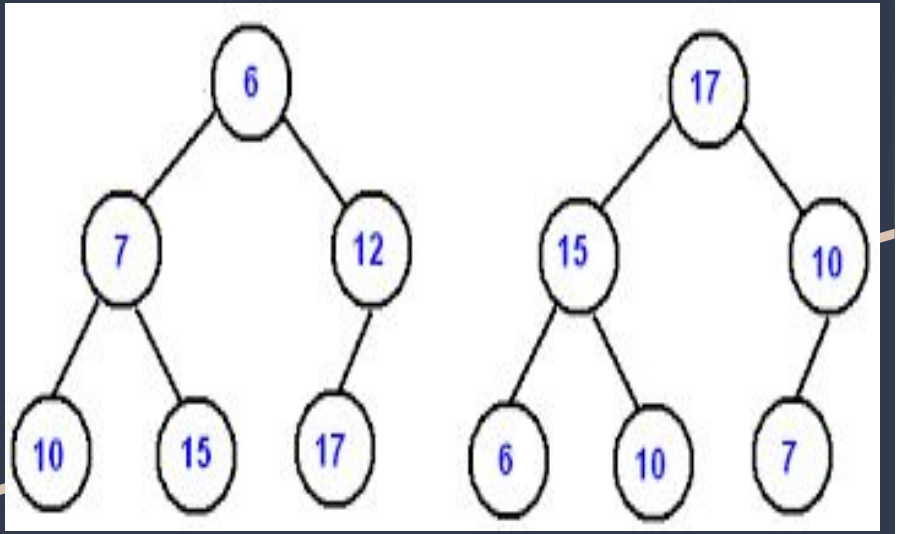
Inorder: Left-Root-Right

Postorder: Left-Right-Root

Preorder : 6, 4, 3, 5, 8, 7, 9
Inorder : 3, 4, 5, 6, 7, 8, 9
Postorder: 3, 5, 4, 7, 9, 8, 6

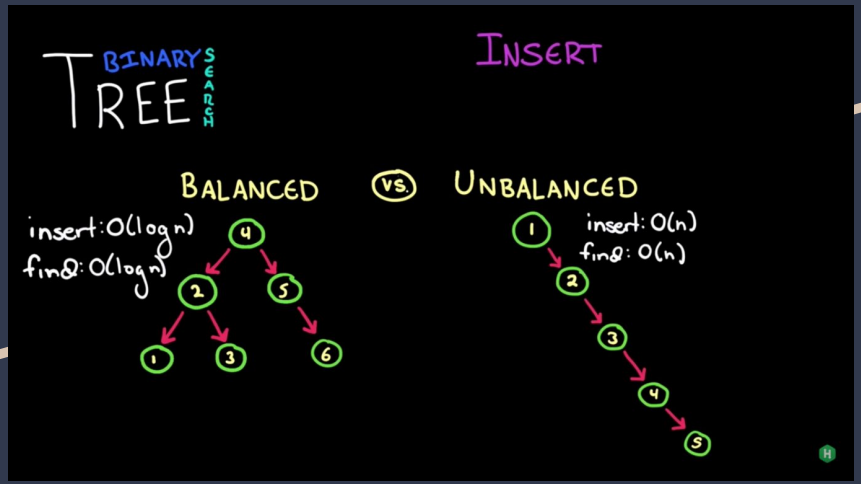
- There are three different ways to traverse, or search and print out the data,through a tree: Inorder(LPR), Preorder(PLR), and Postorder(LRP).
- Inorder will go left, print the data, then go right. This is the most useful since it prints it in order
- Preorder will print the data, go left then right
- Post order will go left then right, then print the data

What are Heaps?



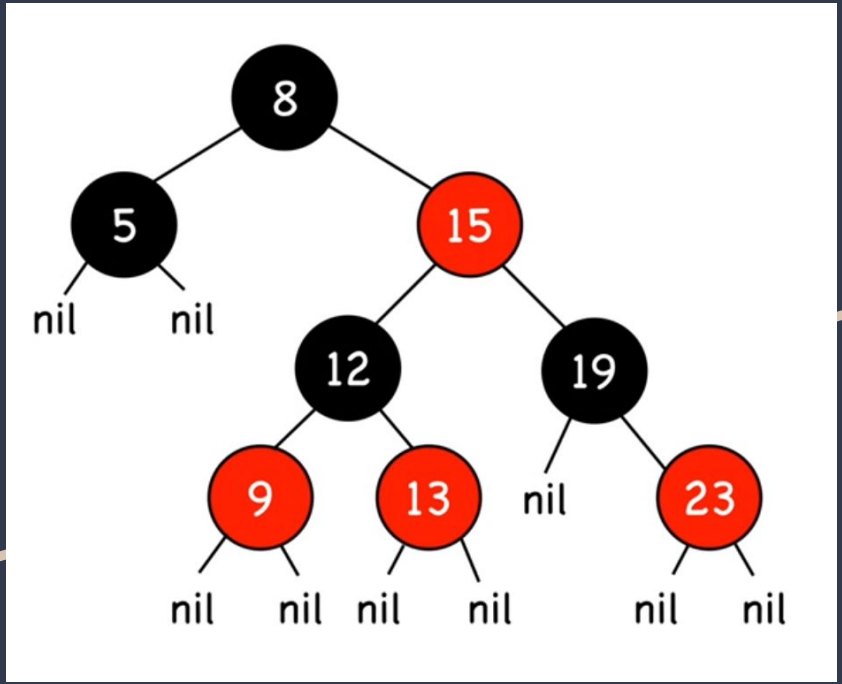
- Heaps are similar to trees, but instead of having a constant root, it is either the minimum data point or the maximum.
- The two heaps are called the Min Heap and Max Heap
- Whenever a new data value is inputted, it goes in the next NULL spot from left to right
- If the new data point is in a Min Heap and is smaller than its parent data value, it will ascend until it has a parent value smaller than itself (This is opposite for Max Heaps)

What is a Balanced Tree? And Why is it Wanted?



- Balanced Trees make sure that the number of data points are equal on both the left and the right of the tree.
- An Unbalanced Tree is extremely one-sided and can cause a tree to become inefficient. The more linear a tree is, the more unbalanced it is
- A Balanced Tree will be the easiest to find each data point

What are Red-Black Trees?



- A Red-Black Tree is like a Binary Search Tree, but it is guaranteed to be balanced.
- A Red-Black Tree must have either red or black data points, must have a black root and black leaves, all red parents have black children data points, and all paths from the root to the leaves have an equal amount of black data points

How do Red-Black Trees Work?

- Whenever a data point is inserted or removed, the structure of the tree will change
- A tree may rotate, or put subtrees in different areas. Trees can be rotated left or right, making the data point move in the desired location
- Whenever a new data point is inserted, the color is set to red. If the new point becomes the root, it turns black. If it has a red parent and uncle, then the parent, grandparent, and uncle switch colors. If the parent is an opposite direction from its parent than its child, the parent and the child will rotate, but only if the uncle is a black data point.

Big-O Notation

- Binary Search Tree: (O(n))
- Heap: (O(logn))
- Red-Black Tree: (O(logn))

References

https://www.bigocheatsheet.com/

https://www.youtube.com/watch?v=oSWTXtMglKE

https://www.youtube.com/watch?v=t0Cq6tVNRBA

https://www.youtube.com/watch?v=5IBxA-bZZH8