Binary Search Trees

Description	Sorted arrays	Linked List
Inserting a new item	O(N)	O(1)
Searching	O(logN)	O(N)
Removing an item	O(N)	O(1)

Binary search trees are going to make all of these operations quite fast, with O(log N) time complexity !!! ~ predictable

Trees

Root node: we have a reference to this, all other nodes can be accessed via the root node
We have nodes with the data and connection between the nodes // edges
In a tree: there must be only a single path from the root node to any other nodes
in the tree
~ if there are several ways to get to a given node: it is not a tree !!!

Binary search trees

- Every node can have at most two children: left and right child
- Left child: smaller than the parent
- Right child: greater than the parent

Why is it good? On every decision we get rid of half of the data in which we

are searching !!! // like binary search

O(logN) time complexity

Resume

Binary search trees are data structures

Keeps the keys in sorted order:

- So that lookup and other operations can use the principle of - binary search !!!
- Each comparison allows the operations to skip over half of the tree, so that each lookup/insertion/deletion takes time proportional to the logarithm of the number of items stored in the tree
- This is much better than the linear time O(N) required to find items by key in an unsorted array, but slower than the corresponding operations on hash tables

Bruna Santos - March 30, 2018 14:47 pm