**Binary Search Trees**

viernes, 21 de abril de 2023

11:43 p. m.

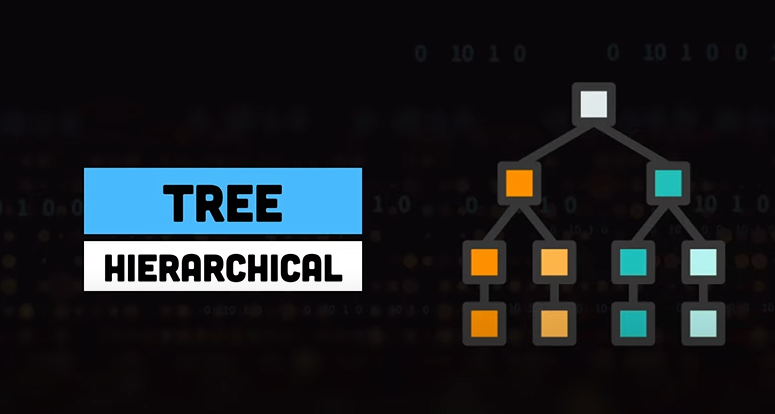
Binary Search Trees

The Roadmap Intro: [Tree Data Structure | Illustrated Data Structures](https://www.youtube.com/watch?v=S2W3SXGPVyU)

Section.io ref: <https://www.section.io/engineering-education/implementing-binary-search-tree-using-python/>

Geeksforgeeks ref: <https://www.geeksforgeeks.org/binary-search-tree-data-structure/?ref=gcse>

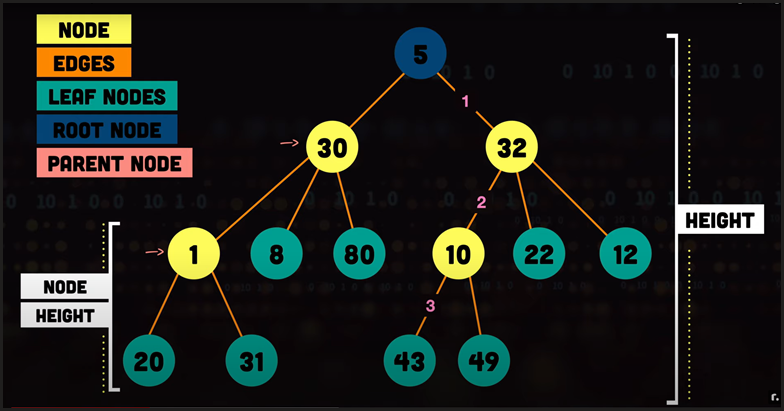
Different from Linked lists, Queues, Stacks and Hash Tables, which all are linear data structures, trees are not. Trees are Hierarchical data structures, and may be useful to represent Organizational Hierarchy, directory structures or Database Indexes.



Interfaz de usuario gráfica, Sitio web

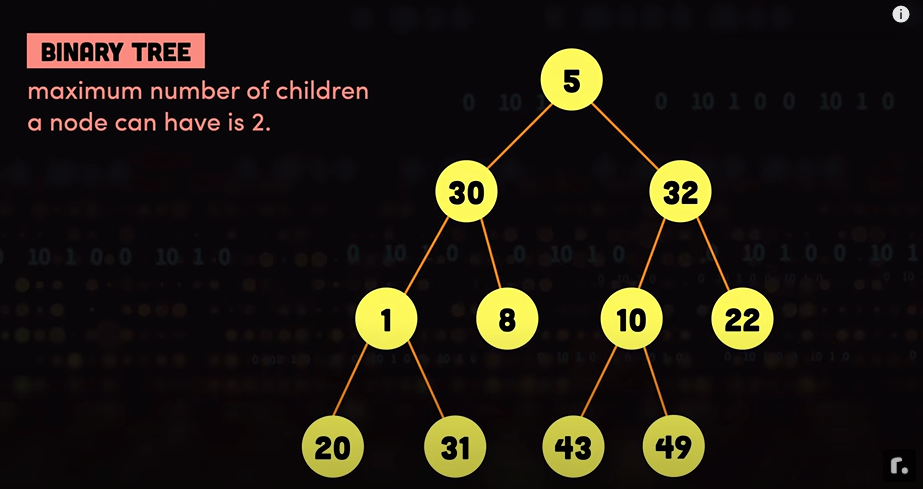
Descripción generada automáticamente

Tree's terminology

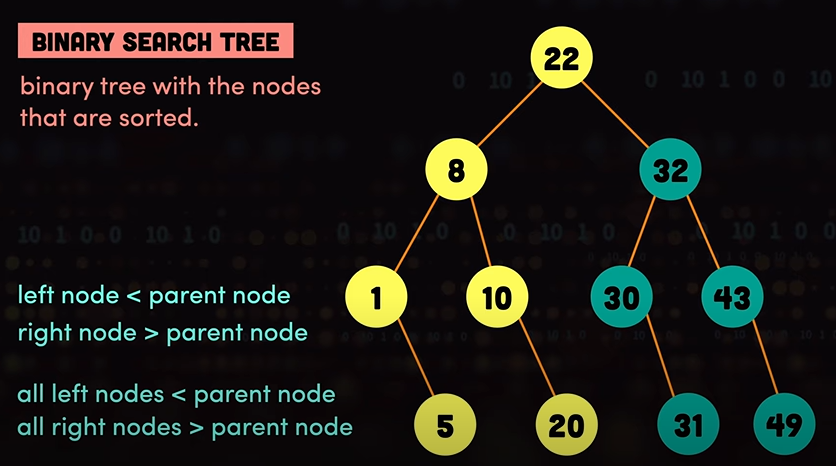


Types of trees

Binary Tree

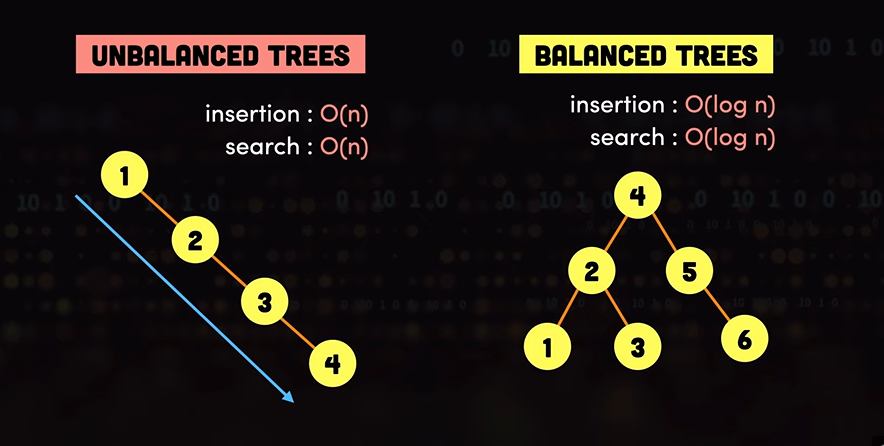


Binary Search Tree

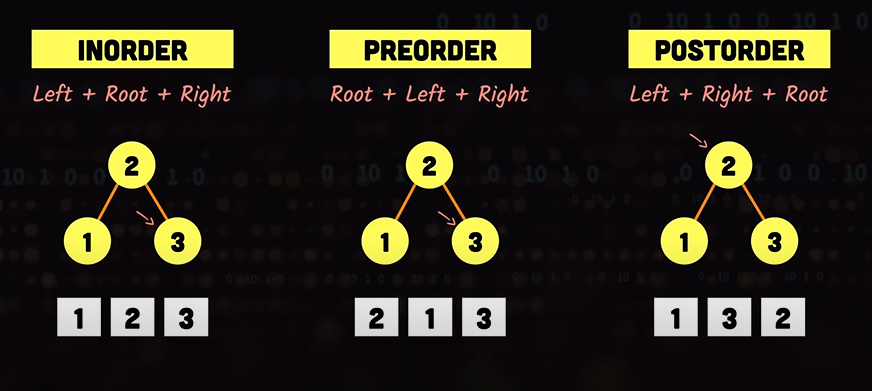


Look up in Trees

The way to look up for an element within a Binary Search Tree is starting a the root and if the value looked for is greater or smaller than the current node (the root, at the start) go you right or left, respectively, until finding the element. Compl: O(n)



Traversal in Trees



Strengths and Weaknesses

**Strengths**

**Efficient Search:** BSTs offer efficient searching, especially in balanced trees. On average, you can find an element in O(log n) time, making them faster than linear search in arrays.

**Ordered Data:** BSTs maintain elements in sorted order. This is useful when you need to maintain a sorted collection and perform operations like finding the minimum and maximum values, range queries, or performing in-order traversals.

**Simple to Implement:** BSTs are relatively easy to implement. The basic operations, such as insertion, deletion, and searching, are straightforward to understand and code.

**Dynamic Structure:** BSTs allow for dynamic data insertion and deletion while maintaining their balance (in self-balancing BSTs). This makes them suitable for applications where data changes frequently.

**Memory Efficiency:** BSTs can be memory-efficient compared to other data structures like hash tables. They only store data and pointers, making them suitable for memory-constrained environments.

**Weaknesses**

**Unbalanced Trees:** The major weakness of BSTs is that they can become unbalanced, leading to worst-case time complexities of O(n) for search, insert, and delete operations. This can happen when data is inserted in a sorted order, creating a skewed tree.

**Dependent on Input Data:** The performance of a BST heavily depends on the order in which elements are inserted. If data is inserted in a way that creates an unbalanced tree, it can lead to inefficient operations.

**Lack of Guaranteed Balancing:** While self-balancing BSTs like AVL trees and Red-Black trees exist to address the balancing issue, standard BSTs do not guarantee balance. Maintaining balance often requires extra overhead.

**Memory Overhead:** BSTs have a memory overhead due to the need to store pointers for each node. This overhead can be significant when dealing with a large number of nodes.

**Limited for Multidimensional Data:** BSTs are suitable for one-dimensional data. For multidimensional data, specialized data structures like k-d trees or quadtrees are more appropriate.

**Slower than Hash Tables:** For certain operations, like direct access (i.e., no need to search), hash tables provide faster performance than BSTs.

Heap Use Cases

Auto-Complete Suggestions: BSTs can be employed to provide auto-complete suggestions in text editors and search engines. As the user types, the BST can be used to search for words or phrases that match the entered prefix.

File Systems: File systems often use BSTs to keep track of file structures. Directories and files can be organized in a hierarchical manner using BSTs.

Priority Queues: BSTs can be adapted to create a priority queue with efficient insertion and removal of the element with the highest priority.

Network Routing: In computer networks, BSTs can be used for IP routing tables to efficiently find the best route for data packets based on their destination IP addresses.

Natural Language Processing: BSTs are used in linguistic and language processing applications for organizing and searching word frequency data, stemming information, or creating linguistic trees for sentence analysis.