**What is a Balanced Binary Search Tree?**

* Binary search trees can be **balanced**
  + Subtrees hold nearly equal number of nodes
  + Subtrees are with nearly the same height

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Structure** | **Worst case** | | | **Average case** | |
| Search | Insert | Delete | Search Hit | Insert |
| BST | N | N | N | 1.39 lg N | 1.39 lg N |
| 2-3 Tree | **c** lg N | **c** lg N | **c** lg N | **c** lg N | **c** lg N |
| Red-Black | 2 lg N | 2 lg N | 2 lg N | lg N | lg N |
| AVL Tree | 1.44 lg N | 1.44 lg N | 1.44 lg N | lg N | lg N |

**What are B-Trees?**

* [**B-trees**](https://en.wikipedia.org/wiki/B-tree)are generalization of the concept of ordered binary search trees – see the [**visualization**](https://www.cs.usfca.edu/~galles/visualization/BTree.html)
  + B-tree of order **b** has between **b** and **2\*b** keys in a node and between **b+1** and **2\*b+1** child nodes
  + The keys in each node are ordered increasingly
  + All keys in a child node have values between their left and right parent keys
* B-trees can be efficiently stored on the hard disk

**B-Trees vs. Other Balanced Search Trees**

* B-Trees hold a **range of child nodes**, not single one
  + B-trees do not need re-balancing so frequently
* B-Trees are good for **database indexes**
  + Because a single node is stored in a single cluster of the hard drive
  + Minimize the number of disk operations (which are very slow)
* B-Trees are almost perfectly balanced
  + The count of nodes from the root to any **null** node is the same

**2-3 Trees**

**Definition**

* A 2-3 search tree can contain:
  + Empty node (**null**)
  + 2-node with **1 key** and **2 links** (children)
  + 3-node with **2 keys** and **3 links** (children)
* As usual for BSTs, all items to the left are smaller, all items to the right are larger.

**2-3 Tree Properties**

* Unlike standard BSTs, 2-3 trees **grow from the bottom**
* The **number of links** from the root to any **null** node is the same
* Transformations are **local**
* Nearly **perfectly balanced**
* Inserting **10 nodes** will result with height of the tree **2**
  + For normal BSTs the height can be **9** in the worst case

**AVL Trees**

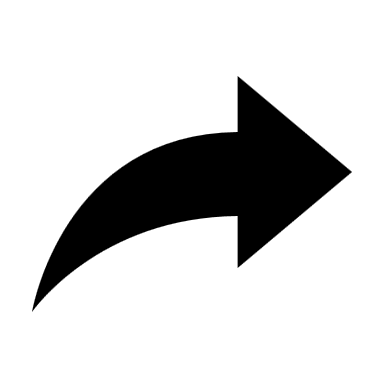
* [AVL tree](https://en.wikipedia.org/wiki/AVL_tree) is a self-balancing binary-search tree ([visualization](https://www.cs.usfca.edu/~galles/visualization/AVLtree.html))
  + Height of two subtrees can **differ by at most 1**

**AVL Tree Rebalancing**

* Height difference is measured by a balance factor (BF)
* **BF(Tree)** = **Height(Left)** – **Height(Right)**
* BF of any node is in the range **[-1, 1]**
* If BF becomes **-2** or **2** 🡪 rebalance
* Rebalancing is done by retracing
* Start from inserted node's parent  
  and go up to root
* Perform **rotations** to restore balance

**Right Rotation**

* Set (x) to be child of (y)
* Set Right Child of(y) to be Left Child of (x)



**AVL Tree Insertion Algorithm**

* Insert like in ordinary BST
* Retrace up to root
  + Modify balance / height
  + If balance factor ∉ [-1,1]  
     🡪 rebalance

**Double Right Rotation**

* Rotate Right (node) with negatively balanced Left Child