**2-3-4 TREES**

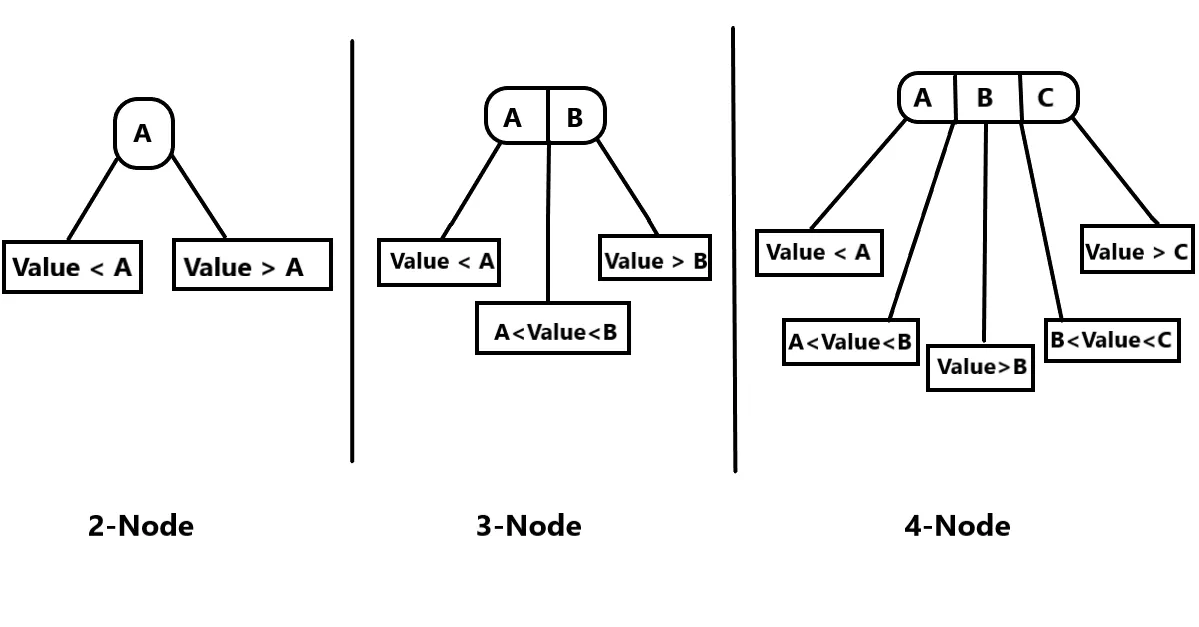
**Introduction**

In this project I’ll talk about 2-3-4 trees. 2-3-4 tree is a balanced search tree. This trees also known as 2-4 trees. 2-3-4 trees are always perfect balanced trees. 2-3-4 trees are useful in indexing and storage in large databases systems.

**Properties**

* Each Node in the tree can store at most 3 values/keys and 4 references or pointers to child nodes.
* The values in each node are Ordered or in Sorted form.
* All leaf nodes are at same level. Hence making it perfectly balanced.
* An Internal (non-leaf) node can either have 2, 3 or 4 children. To be more precise, nodes can be of the following three types.
  1. **2-Node:** Node has two child pointers and 1 data or key value.
  2. **3-Node:** Node has three child pointers and 2 data elements.
  3. **4-Node:** Node has four child pointers and 3 data elements.
* A leaf node can have 2, 3 or 4 items but no children. In other words, a leaf is 2-Node, 3-Node or 4-Node where all pointers hold NULL reference.

So, there is a picture I found in internet. In this picture we can understand clearly what is the 2-3-4 trees.



So, as you see there is a 1 parent and 2 children in first,2 parent and 3 children in second and 3 parent and 4 children in third part of picture.

**Operations over 2-3-4 trees**

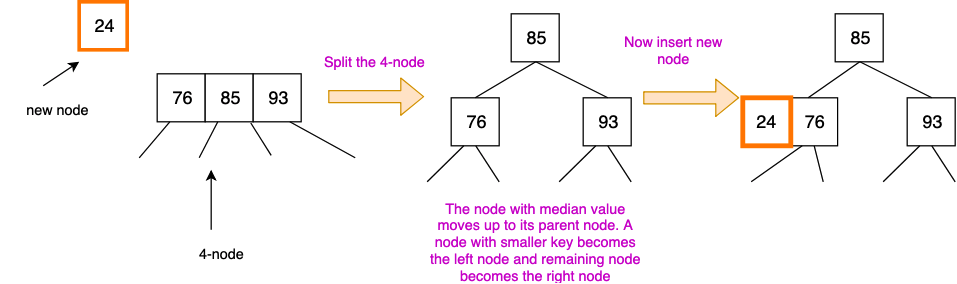
The major operations on 2-3-4 trees are search, insert and delete.

### Search operation

Compare the item to be searched with the keys of the node and move to the appropriate direction. Unlike BST where we move either to the left child or to the right child, we need to make choice among three or four different paths.

Insert operation

IMPORTANT NOTE: A node cannot hold more than three keys. If a node is full before insertion, we split the node so that the new node can be inserted.

The insertion takes place always on the leaf nodes. I repeat again, *we never insert a new node on the internal nodes even if they have room to accommodate*. Therefore, we perform the search operation on the tree until we reach the leaf node. If the leaf node is a 2-node, we insert the item and make it a 3-node. Similarly, if the leaf node is a 3-node, we make it a 4-node. But what if the node is a 4-node? We cannot insert a new node in this node, right? The node is already full. In this case, we split the node that splits into nodes with a smaller number of keys and inserts the new node in the appropriate child node. So, in the picture you can see how to split 4-node and insert new node to the 2-3-4 tree.

### Delete Operation

Delete is a bit tricker than insert operation. Depending upon the location of the node containing the target (x) to be deleted, we need to consider several cases. I am going to explain each of the cases one by one.

Case 1: If x is in a leaf node

This has further two cases.

Case 2: If x is either in a 3-node or 4-node

Delete x. If the node is a 3-node, it becomes 2-node and if the node is a 4-node, it becomes 3-node.

Case 3: If x is in a 2-node

This is called underflow. To resolve this, we need to consider further three cases.

Case 4: If the node containing x has 3-node or 4-node siblings

Convert the 2-node into a 3-node by stealing the key from the sibling. This can be done by left or right rotation. If the left sibling is a 3-node (or 4-node), do the left rotation otherwise do the right rotation.

So that’s all. I’ll share which sources I used. Like you tube and google links.

<https://youtu.be/47u7RU0XNR0> https://youtu.be/uIWLCfT9daI

https://youtu.be/M\_z-qYOY5JQ

https://www.usna.edu/Users/cs/crabbe/IC312/current/units/05/234.html

<https://en.wikipedia.org/wiki/2%E2%80%933%E2%80%934_tree>

https://people.ksp.sk/~kuko/gnarley-trees/234tree.html