2-3 Trees

**Definition:** A 2-3 Tree satisfies:

A. All non-leaf vertices have 2 or 3 children.
B. All leaves have the same depth.
C. Data stored at vertices satisfies:
   
   - **Vertex with two children:** Contains a single data item that is greater than all data in the left subtree and less than all data in the right subtree.
   - **Vertex with three children:** Contains two data items; the smaller of the two is greater than all data in the left subtree and less than all data in the middle subtree, and the larger of the two is greater than all data in the middle subtree and less than all data in the right subtree.
   - **Leaves:** Contain one or two data items.
Example:
**Generalized Binary Search**

For the MEMBER operation, standard binary search can be generalized to compare with two items at vertices with 3 children (MIN and MAX follow a leftmost or rightmost path).

**Note:**

The height of a 2-3 tree is Between $\lfloor \log_3(n) \rfloor$ and $\lfloor \log_2(n) \rfloor$. Hence, a root to leaf path is $O(\log(n))$ in the worst case.
Inserting into a 2-3 Tree

Idea:

• First search down to find the leaf where the new item belongs and add the new item to that leaf.

• If the leaf only had one item before and now has two, then we are done.

• Otherwise, we now have three items to work with: split off a new leaf to contain one of the items (and attach this new leaf to the parent), leave one item in the existing leaf, and give the third item to the parent.

• Move up to the parent and continue this process until a vertex that previously had only one item and two children is reached (and so it is ok to add a third child and additional item) or until the top of the tree is reached and a new root can be created.
procedure INSERT(d,T):
    Search down to place the new item d in the appropriate leaf v.
    while v has 3 data items \( a \leq b \leq c \) do begin
        if v is the root
            then \( p := \) a new root with v as its leftmost child
        else \( p := \) PARENT(v)
        Remove a and the two leftmost subtrees of v (which are nil if v is a leaf) to form a new subtree and attach this new subtree to p as the left sibling of v.
        Move b to p.
        \( v := p \)
        end
end
Example, insert 24 and 26 into the previous example:
Note:
Other operations such as deleting a vertex and joining and splitting 2-3 trees can also be done in $O(\log(n))$ time.