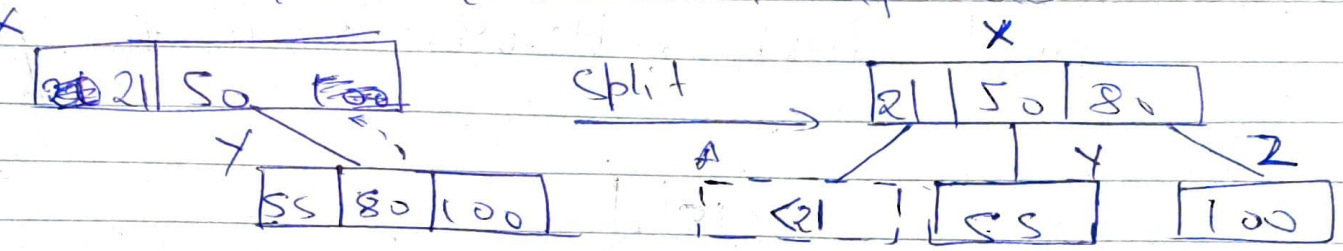
Insertion is implemented using proactive insertion algorithm where before going down, we split the current node if it is full.

Advantage

1. Do not traverse a node twice
2. Always have a free space in the leaf node. (new key is always inserted at leaf node)



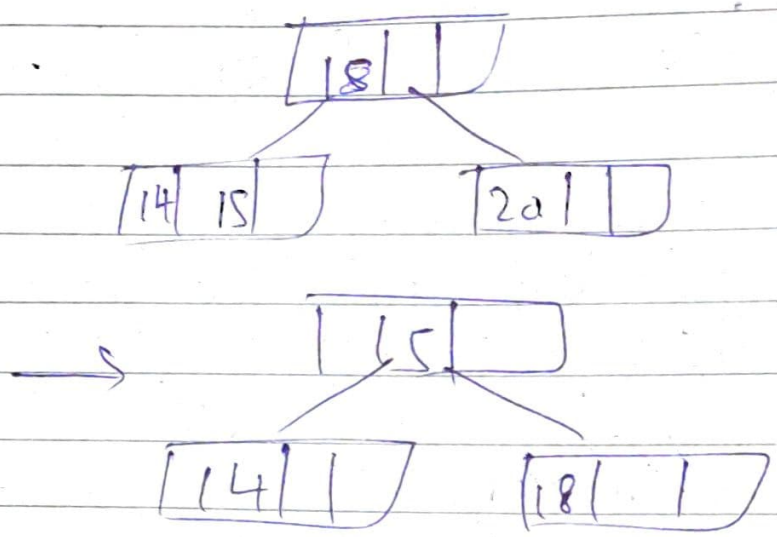
1. Initialize x as root
2. While x is not leaf do following
   a) Find the child of x that is going to be traversed next. Let the child be y.
   b) If y is not full, change x to point to y
   c) If y is full, split it and change x to point to one of the two parts of y. If k is smaller than mid key in y, the set x as first part of y else second part of y
3. The loop in step 2 stops when x is leaf. x must have space for 1 extra key as we have been splitting all nodes in advance

Deletion

1 Leaf node with more than one key (trivial)

2 Internal node, look for inorder predecessor and swap. This will become like 1st case

3 Deleting from leaf node having only one key. So, all leaf nodes should be at same level, and this would be violated so, perform something like rotation to ~~have~~ hold multi-way search tree property



Borrow from adjacent sibling

4 Extension of 3rd point, if we delete from leaf node with only one key, and adjacent sibling is also having only one key, then perform merge operation

All these operations can lead to cascading