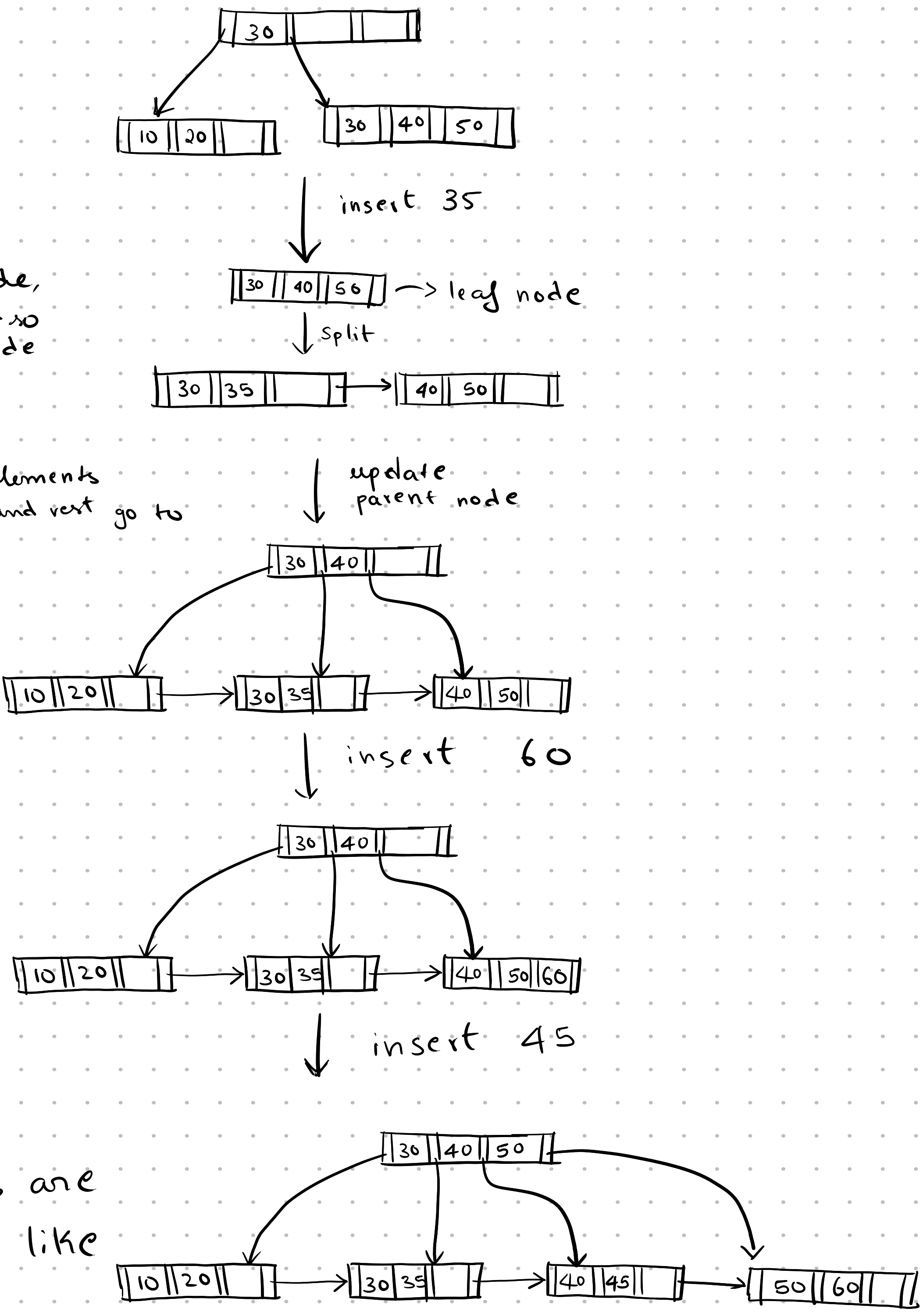
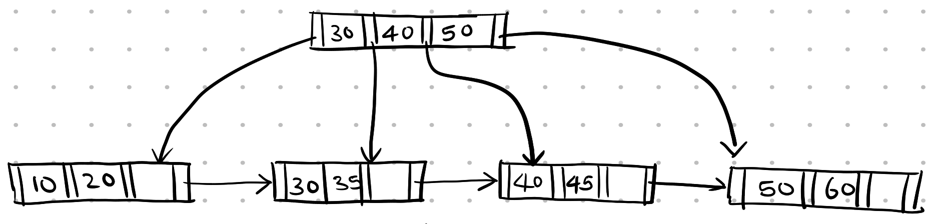
INSERTION

find the leaf node, leaf node is full so split the leaf node

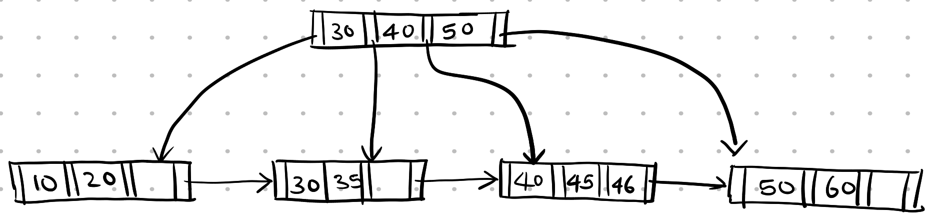
first ⌈n/2⌉ (ceil) elements will goto left node and rest go to new right node

leaf nodes are split just like before





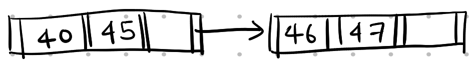
↓ insert 46



↓ insert 47

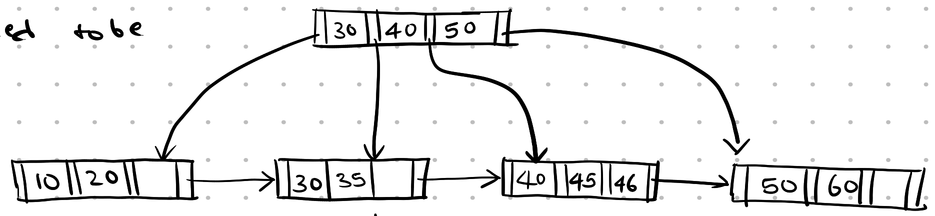
find the node "| 40 | 45 | 46 |"

↓ split



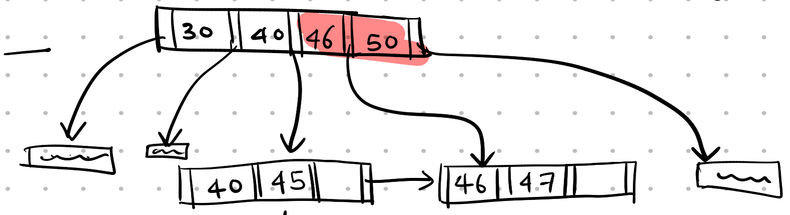
↓ update the parent Node

here the parent node is full and its need to be split it,

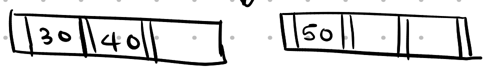


↓ extend (Only Conceptual)

assume the parent node is extended — and is then Split :-



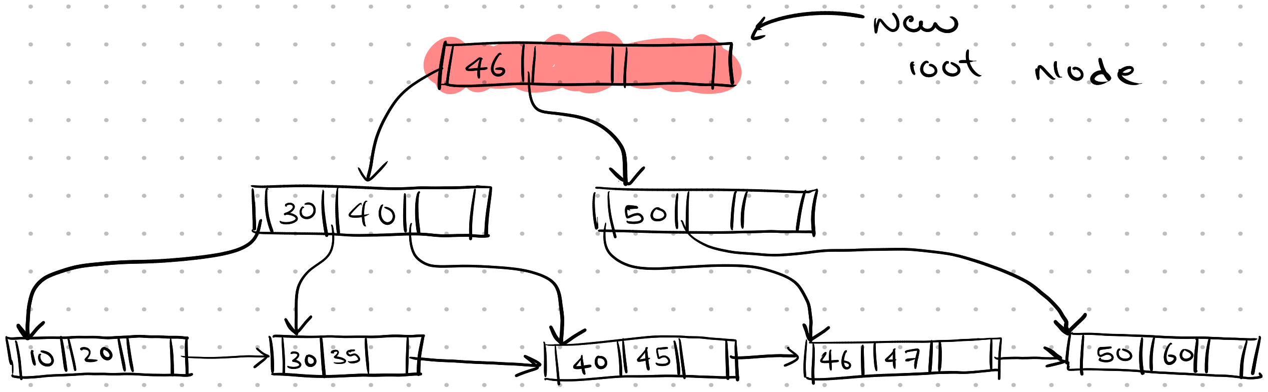
↓ Split



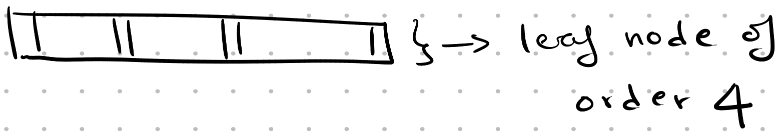
left right

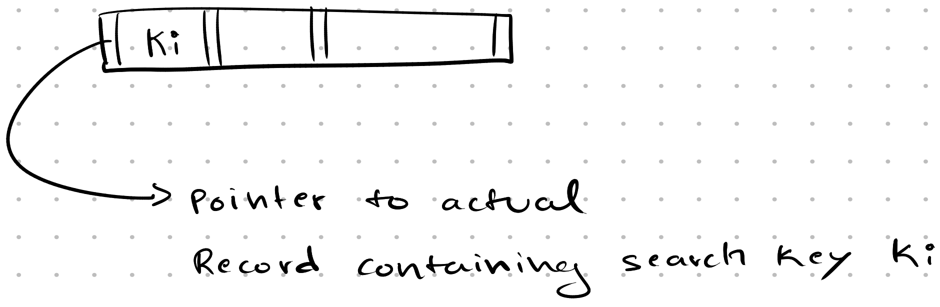
↓ move 46 to parent node

here search key value present b/w pointers which are in kept in the left non leaf-node & the pointers that were moved to right in this case "46", is moved to its parent node but ther is no parent node so we create a new node which incerases the depth of the B-tree

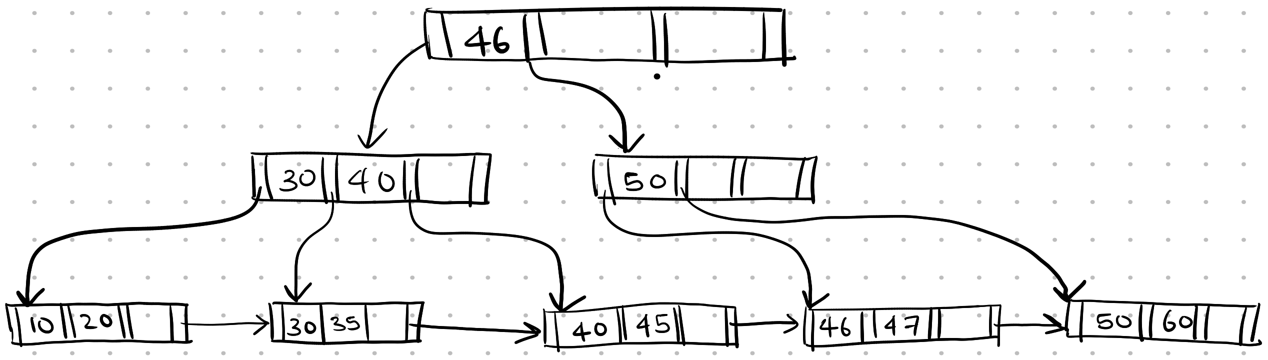


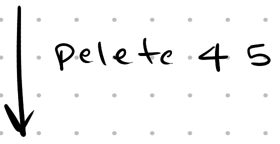
Structure of leaf Node



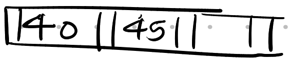


DELETION

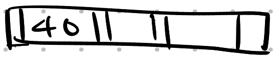




find leaf Node



Delete record entry



Now leaf now has

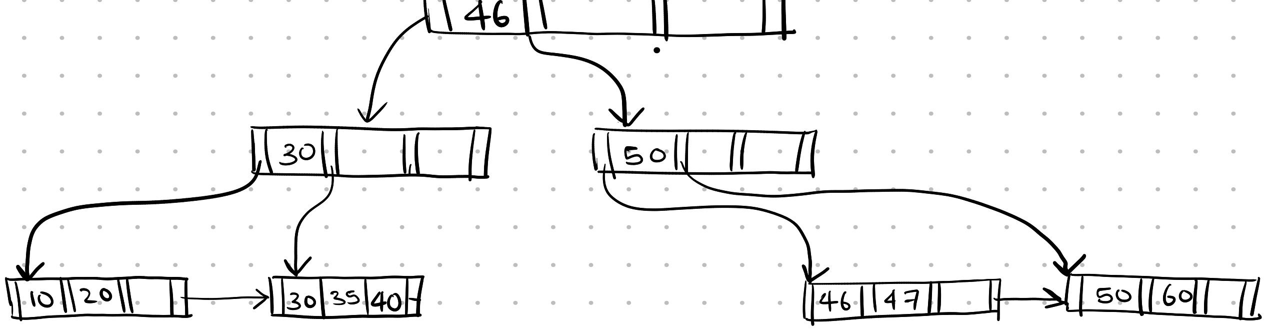
$n=1, \quad n=1 < \lceil (n-1)/2 \rceil$

$< \lceil 3/2 \rceil$

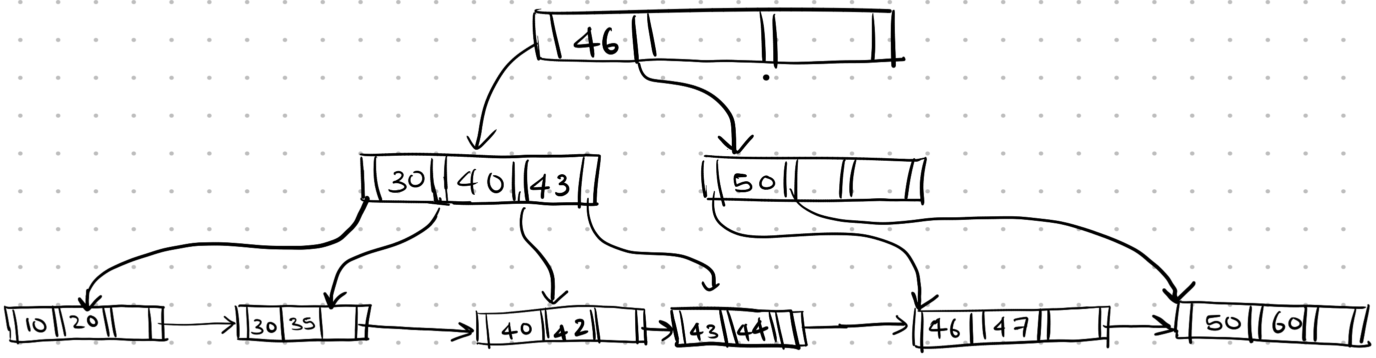
$1 < 2$

so the leaf node must me merged or Redistributed

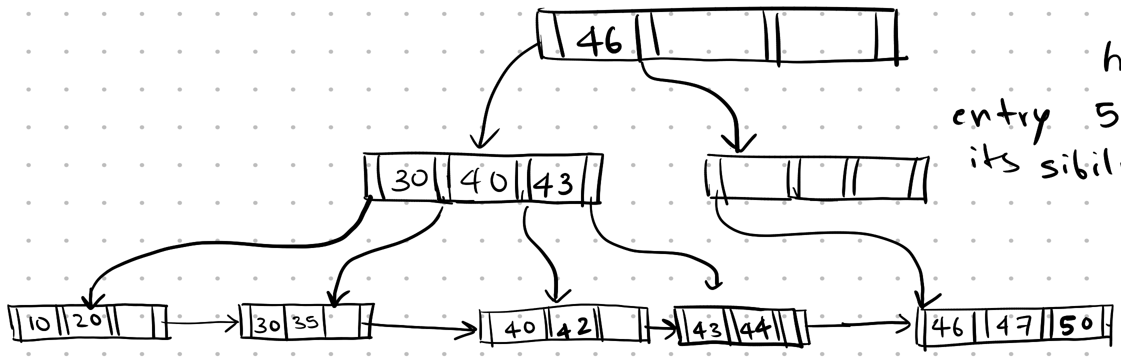
here there is enough space, so the search key "40" is merged with its sibiling, and empty node is deleted and the search key in parent node is also deleted



assume the below b-tree



Delete index entry "60"



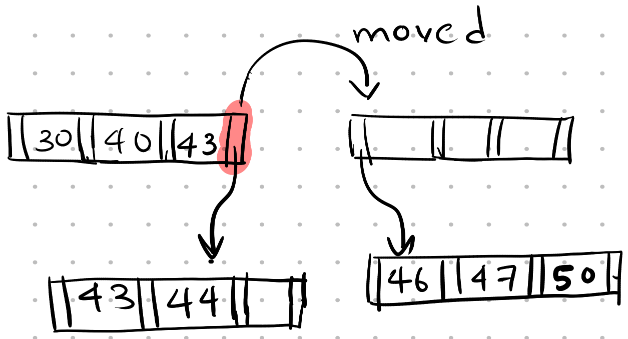
here the index entry 50 is merged with its sibiling node,

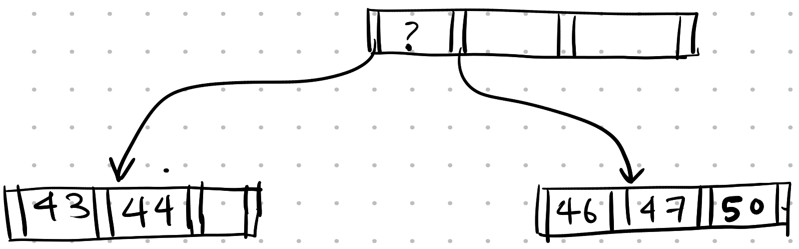
search key in the parent is also deleted, which leaves the parent with only one pointer, n=1

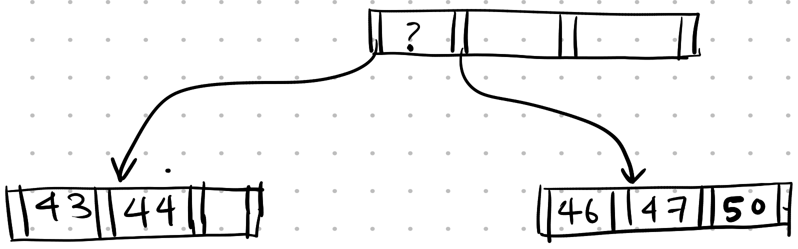
∴ It must be merged or redistributed

here merging is not possible with the sibiling node as it is already full we go with the Redistribution of Pointers

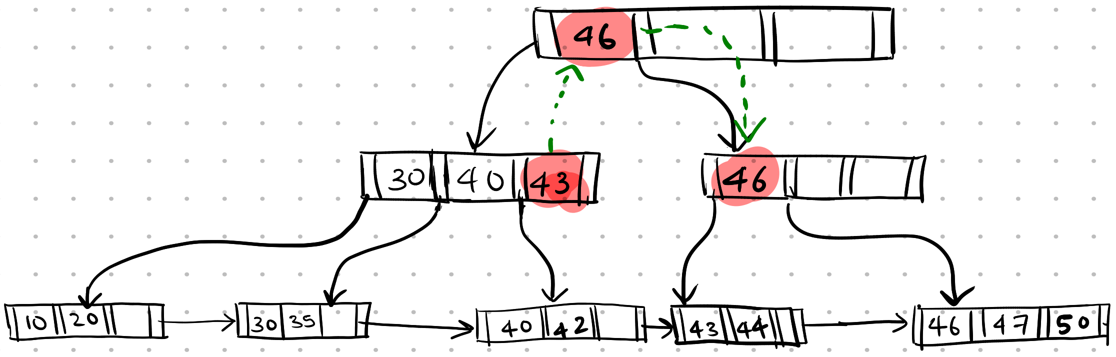
right most pointer is moved to the right node



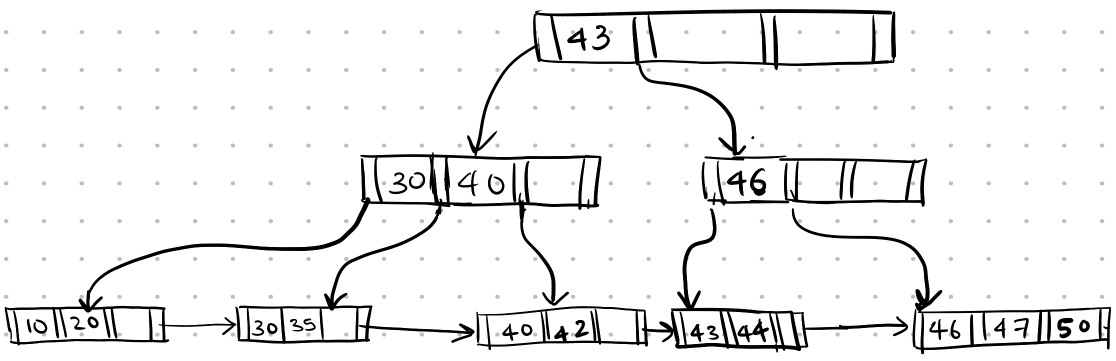




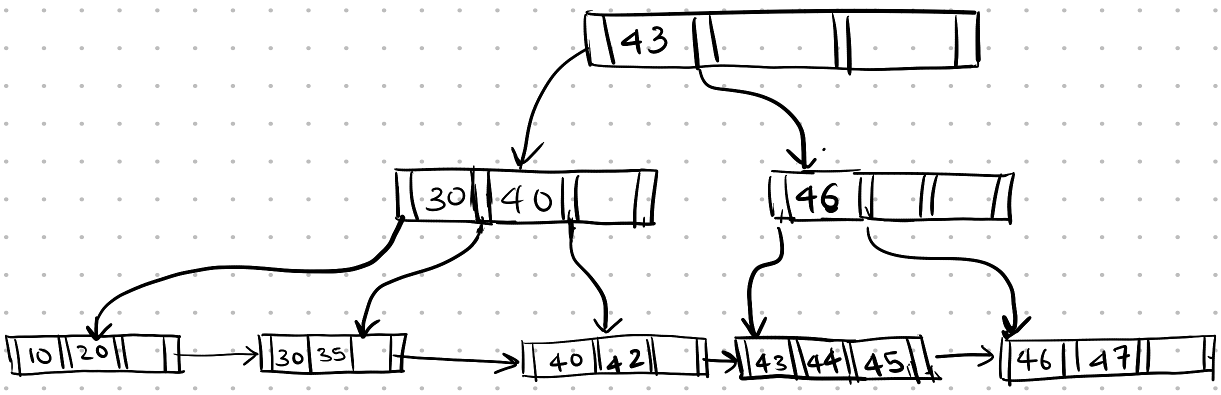
here the value seperating these two pointers is not present in both the nodes, but the search key value present in the "parent node" correctly seperates them



As the pointer are now redistributed, the parent key no-longer correctly seperates the sibiling nodes, so the correctly seperating key "43" is moved up

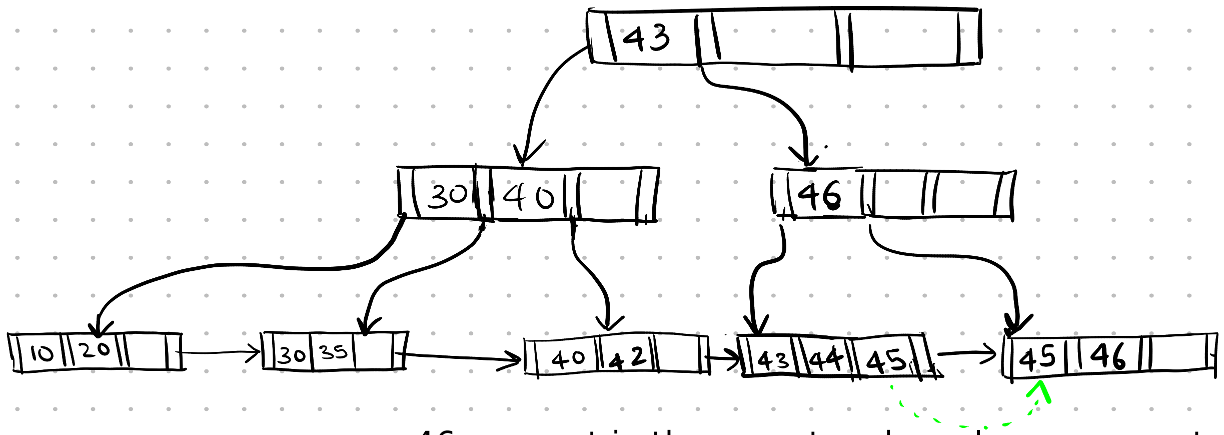


now, assume the below b-tree

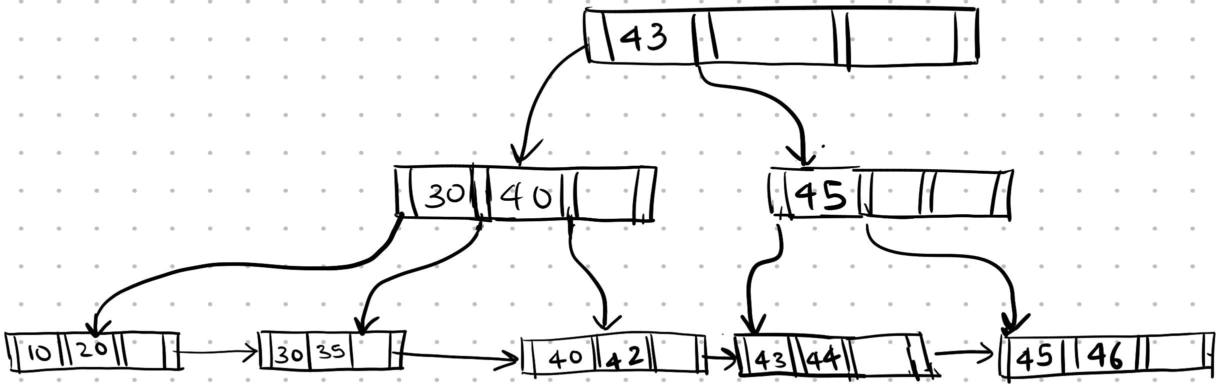


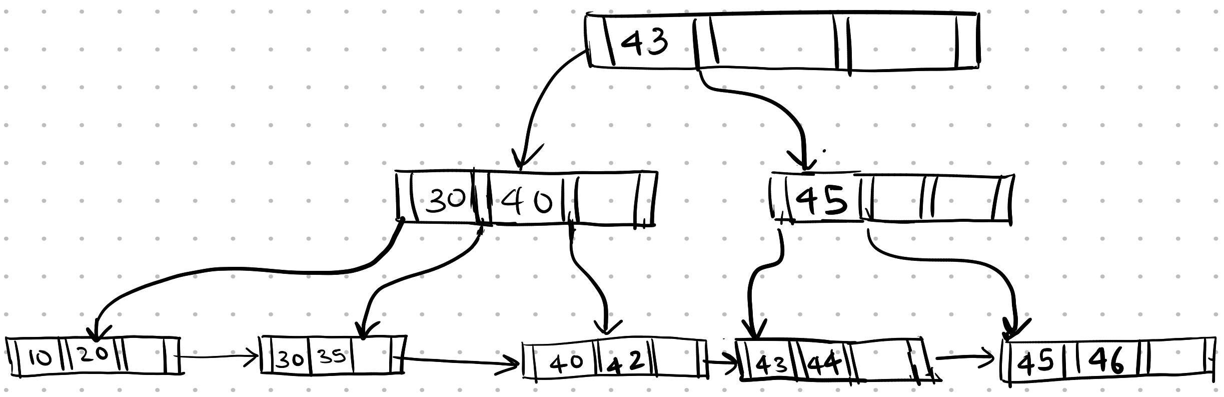
Delete 47

deletion of 47 from the leaf node makes it underfull, now the leaf node
cannot be merged with the silibing node as, it if full so we redistrubute
the nodes, the left most node that is 45 is brought to the right node

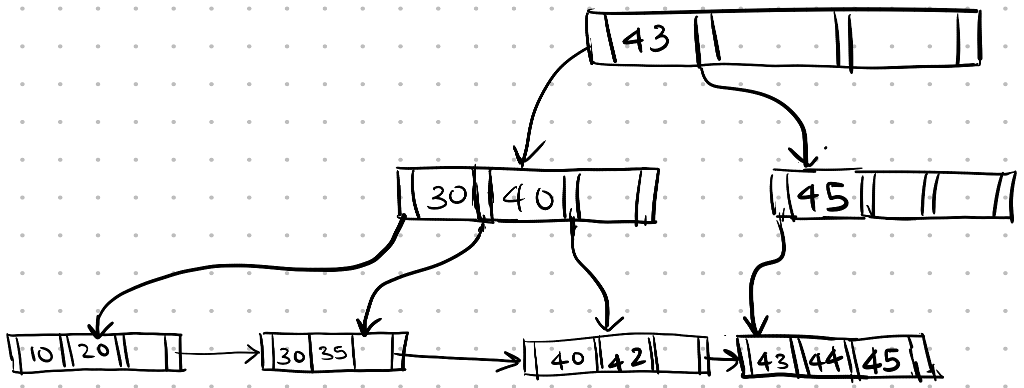


now 46, present in the parent node no longer seperates,
the two child nodes, so we correct it by changing it to
46

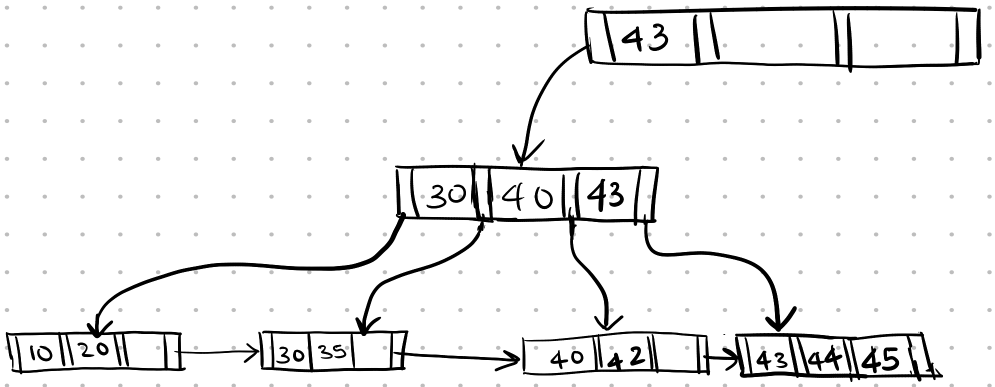


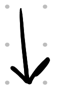


if we delete 46, the index entry 45 can be merged with its' sibiling node

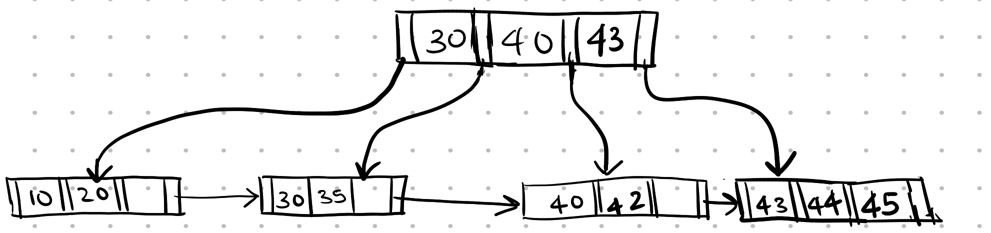


now the parent node is underflowing, it can now also be merged with it's sibiling, the search key value seperating them is the value present in their parent's node 43





now, the root node contains only one pointer it can have atlease two pointers, so the root node is also remove


30
40
43
10
20
30
35
40
42
43
44
45