B trees

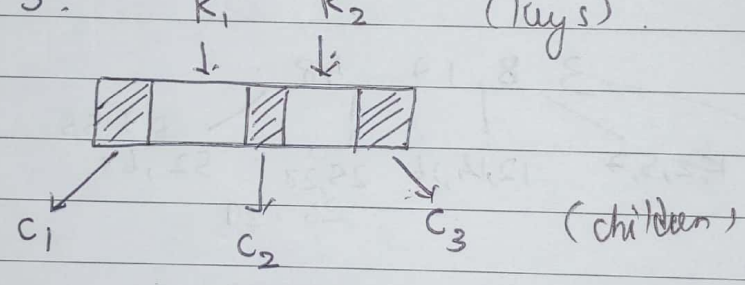
Given factor 'p', no of children of each node ranges from $\lceil \frac{p}{2} \rceil$ to p

no of keys for each node ranges from $\lceil \frac{p}{2} \rceil - 1$ to p-1.

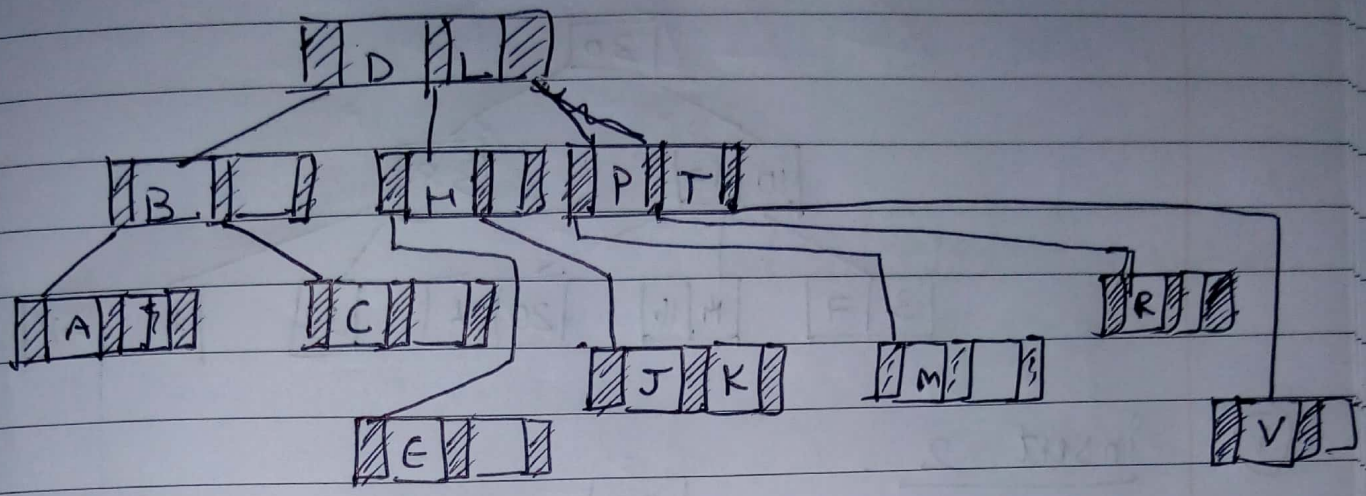
* except root and leaf node.
* All leaves must be at same level.

Structure of node.

if p = 3.



ex: p = 3



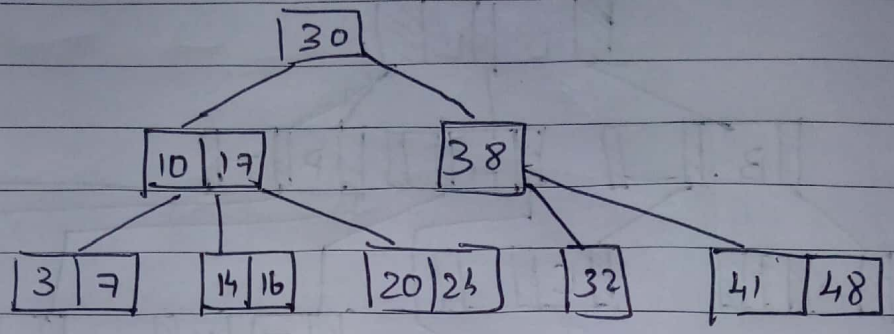
overflow — no of keys in a node becomes more than p-1
underflow — " , " , " , " less than $\lceil \frac{p}{2} \rceil$

In both these situations, rebalance the tree.

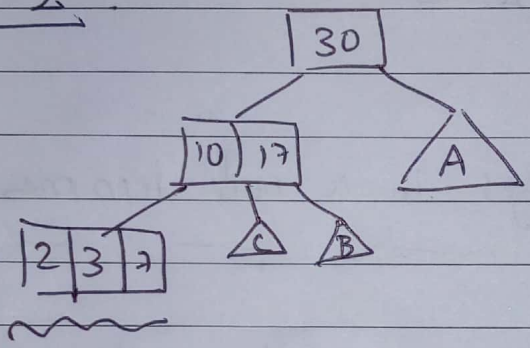
Insertion

* Find node where key to be inserted.
* Insert the key.
* If overflow occurs
  - pull the middle key upwards in the parent (split the node)
  - if parent overflows, repeat step 3 for parent
  - continue till root.

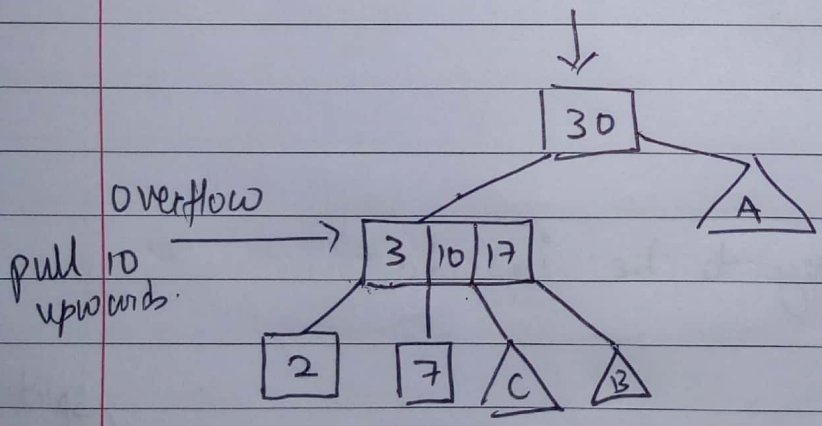
ex: p = 3 .

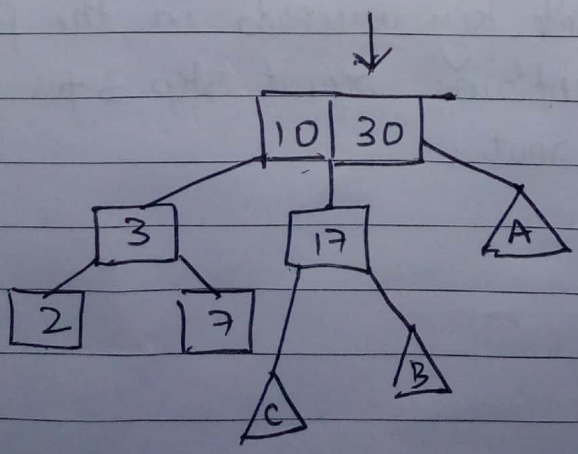


① insert 2.



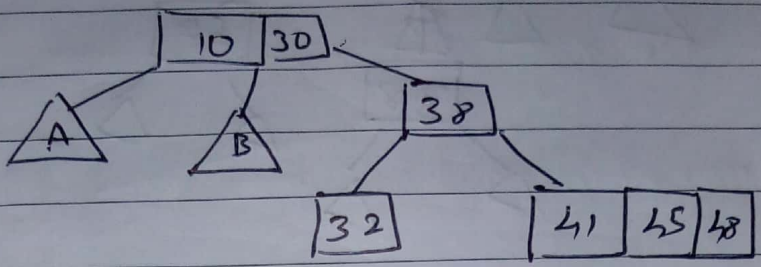
overflow → pull 3 upwards.





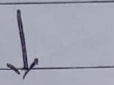
← balanced.

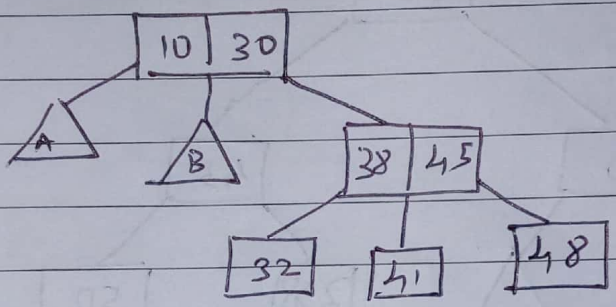
(2) insert 45.



← overflow.

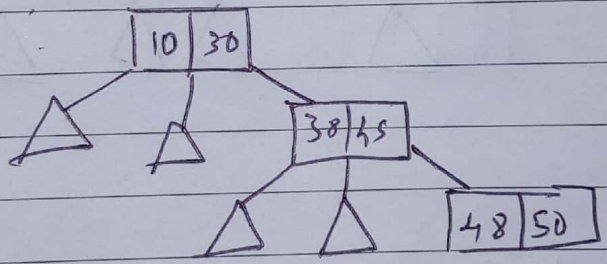
pull 45 up.



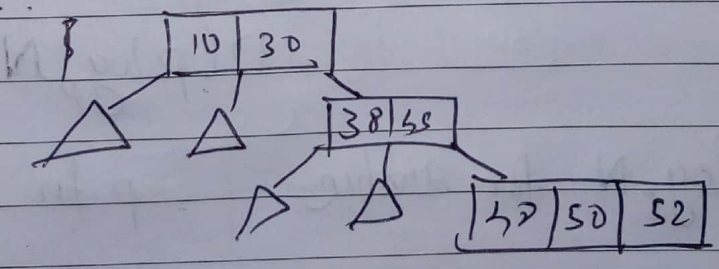


balanced.

(3) insert 50

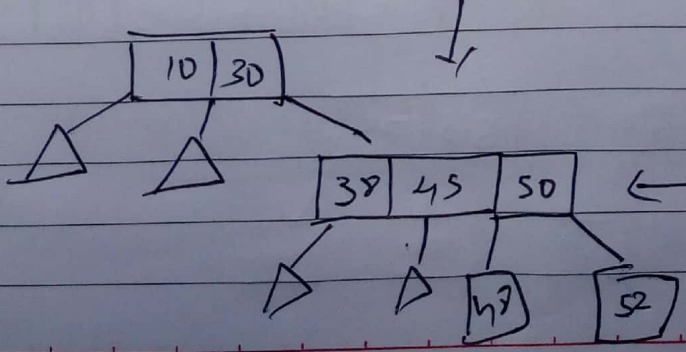


(4) insert 52.



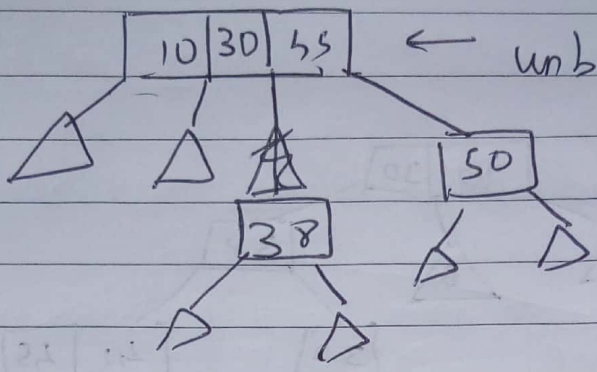
← unbalanced

pull 50.



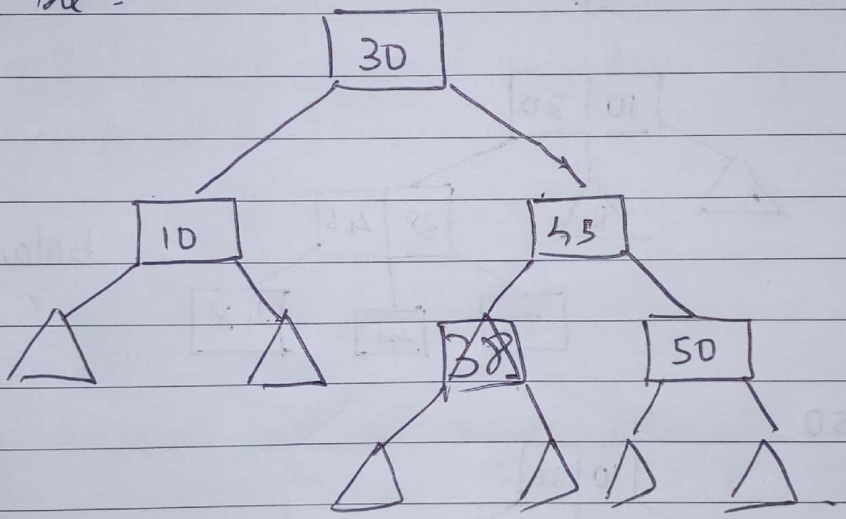
← unbalanced

pull 45 up.



← unbalanced. pull 30 up

final tree :



* Complexity of insertion : ~~O(t log~~

$O(p \log_p N)$

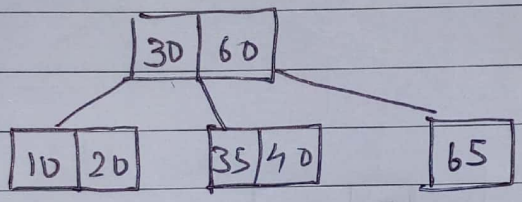
$\log_p N$ for searching p for travelling through node

Deletion

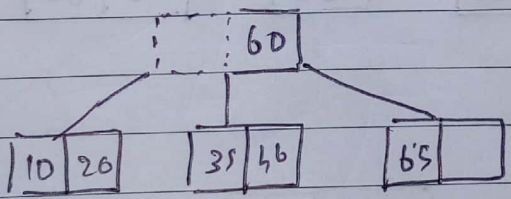
* Deletion always happens at leaf node.

* if non-leaf ~~node~~ Key to be deleted,
  replace this key with ~~right~~ no.
  either right most element of left child
  or left most element of right child and
  propogate deletion downwards to leaf.

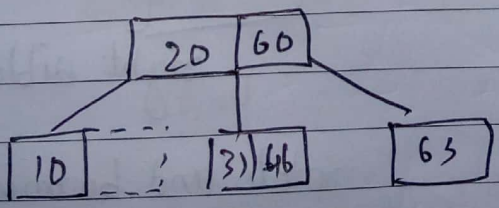
ex:



delete 30.



replace blank block with either
20 or 35 and propogate downwards.



Now this is a problem of leaf node deletion

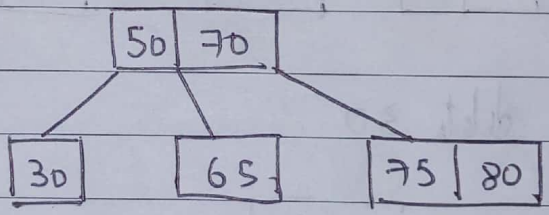
Deletion at leaf node

* delete the key (put blank box).

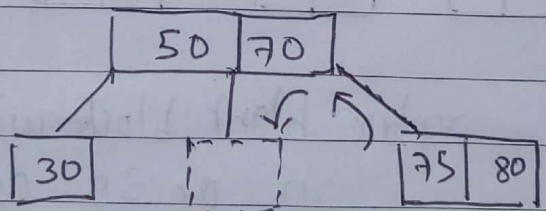
(A) * if any sibling has more than $\left\lceil \frac{p}{2} \right\rceil - 1$ keys, borrow key from sibling, perform rotation

(B) * if sibling does not have extra keys, pull parent down, apply merge

CASE A

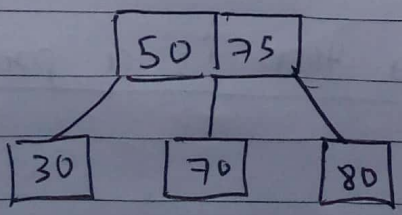


delete 65.



right sibling has extra keys.

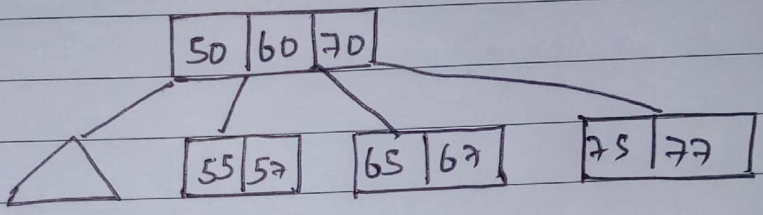
rotate and borrow.



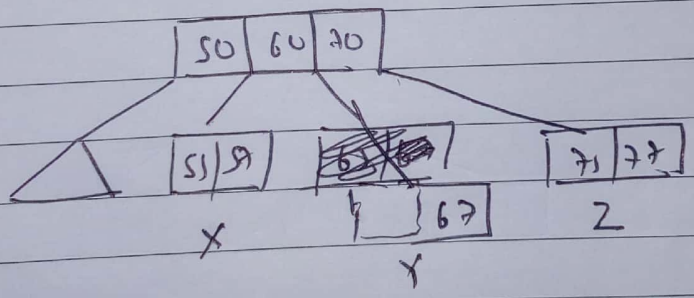
Case B

no sibling has extra keys. Bring parent down.
merge step.

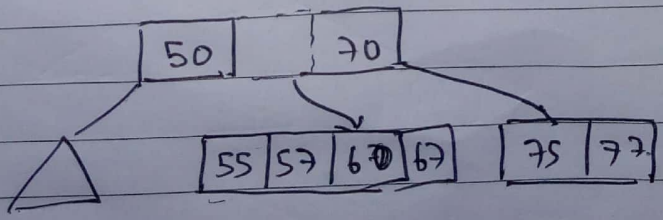
Ex: p = 6. min keys = 2.



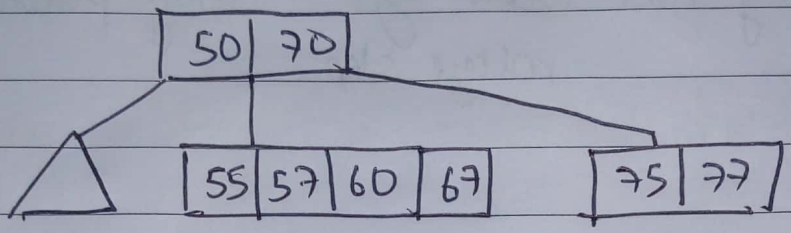
delete 65.
no sibling has more than 2 keys.



We can either merge XY or YZ
Let's merge XY with parent.
Bring 60 down.



collapse and complete merge.



valid tree