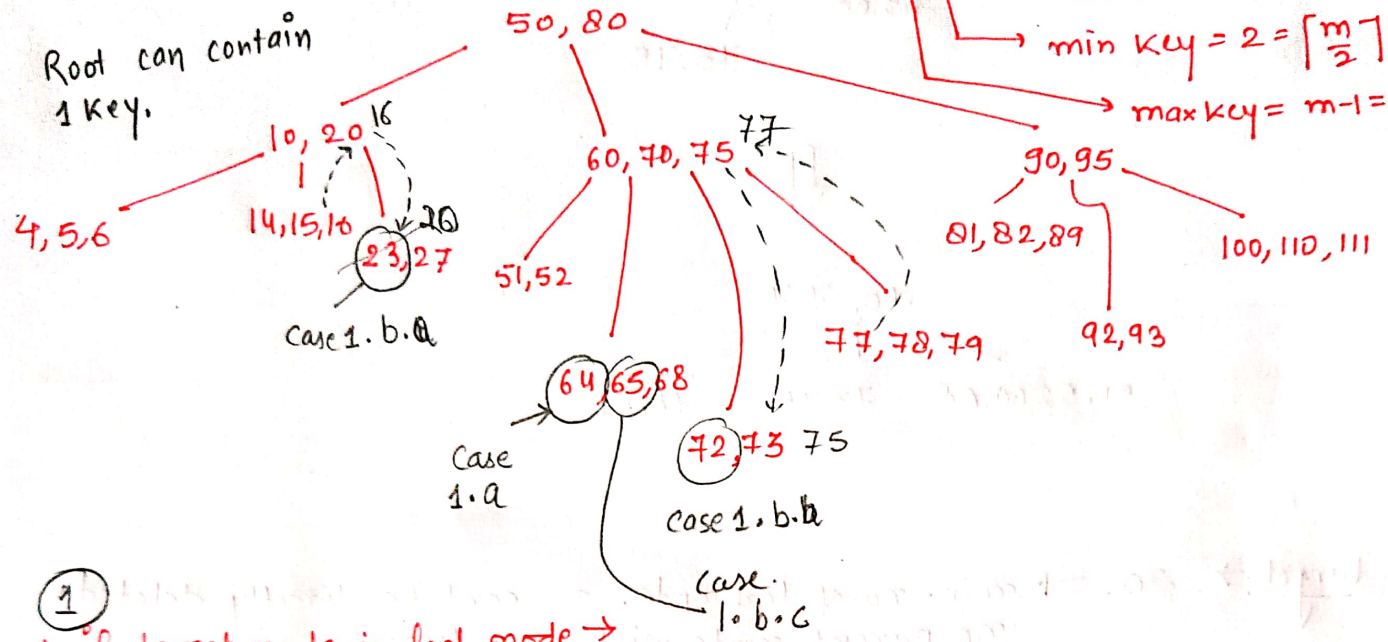
B TREE. of order 5

① if target node is in leaf node.
② if target key is in internal node.

- order m = 5
  - → min children = $\lceil \frac{m}{2} \rceil = 3$
  - → max children = 5
  - → min key = $2 = \lceil \frac{m}{2} \rceil - 1$
  - → max key = $m-1 = 4$.



(1) → if target node is leaf node →

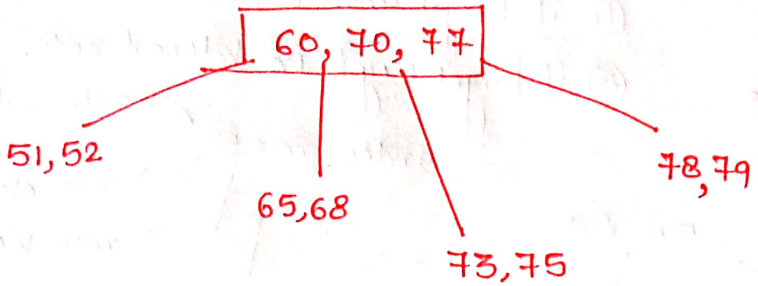
(a) that leaf node contains more than min. no. of keys
↳ Ex 64

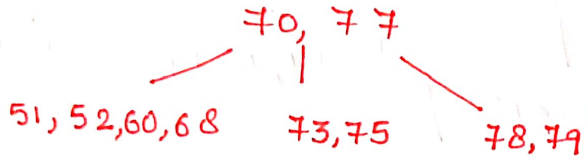
(b) that leaf node contains min. no. of keys. → so have to borrow nodes from immediate left or immediate right node.
↳ Ex 23
- a → borrow from left sibling if more than min.
- b → " " right sibling " "
- c → neither left sibling nor right sibling have more than only min. key.

Target key → 64, 23
↳ can't delete directly because it will violate B-Tree property.

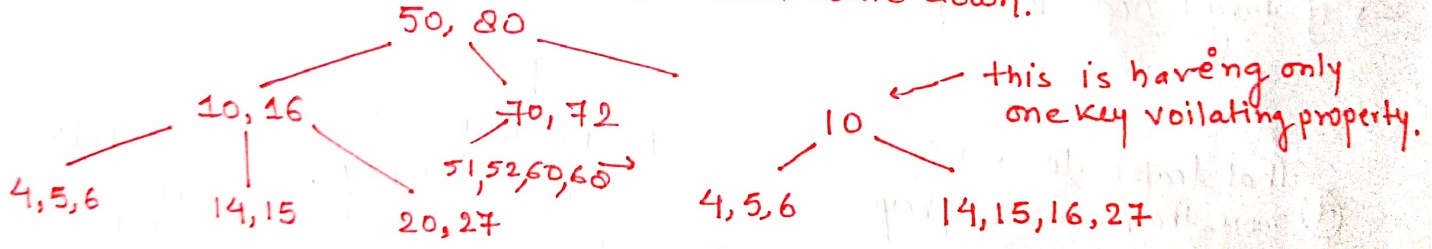
72 → case 1.b.b

65 → can't be directly deleted because nodes have min. no. of keys. → both sibling have min no. of keys. → than we will merge 65 वाली node either with right sibling or left sibling and parent will also be merged.

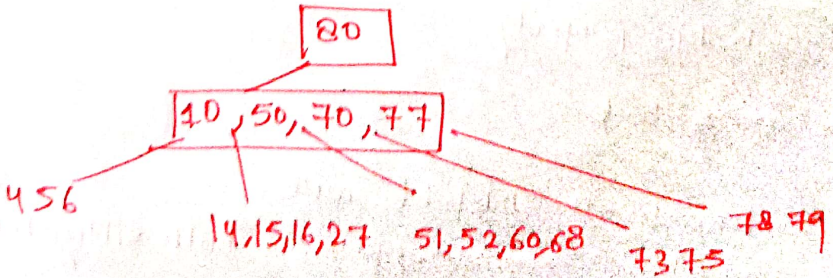




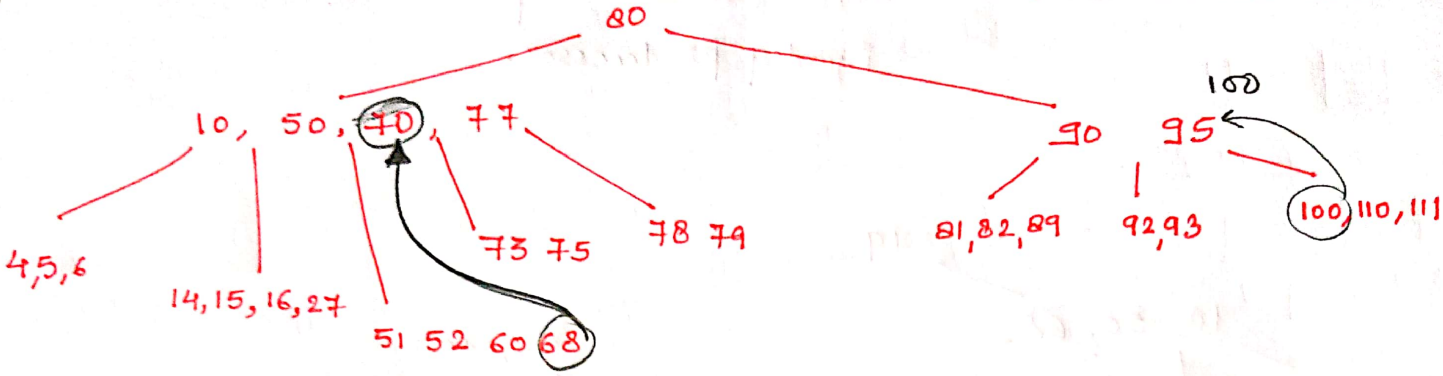
target → 20. → min. no. of leaf node. so cant be directly deleted. no parent node will come down.



So now 10 will ask its sibling. → sibling have only minimum number of keys. Now parent node will merge.



INTERNAL NODE



delete 70. → go choose from Inorder Successor or Inorder Predecessor. (maximum element from left Sub-Tree.

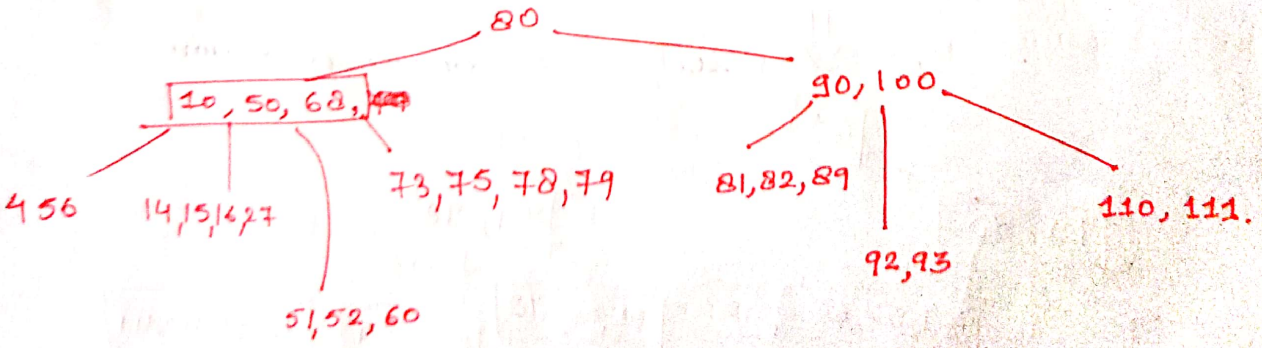
So 70 replaced by 68.

delete 95 → Inorder Successor.

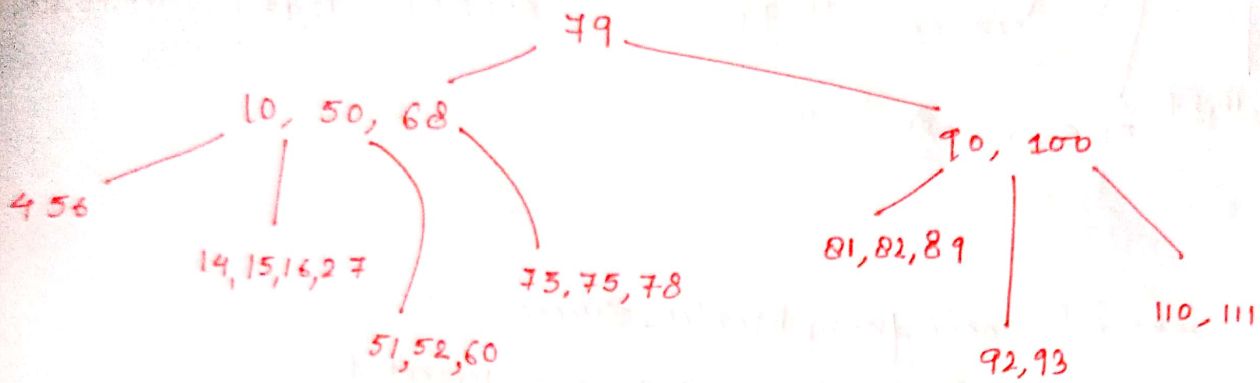
95 replaced by 100.

Internal node
- Inorder Pre
- Inorder Succ

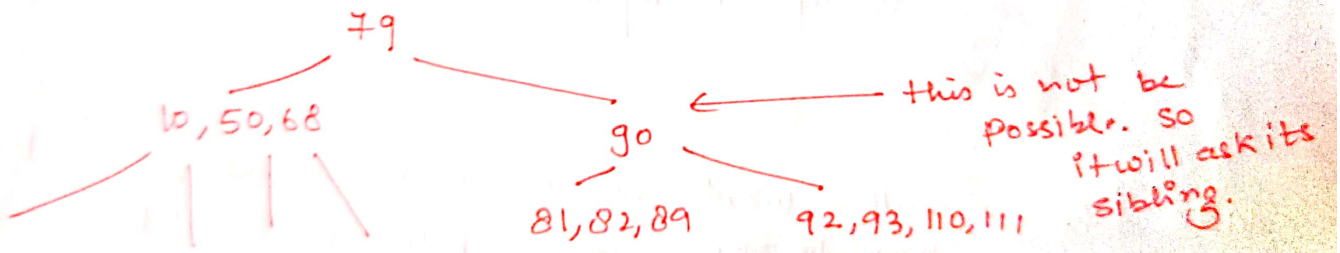
delete 77. → neither Inorder Successor or Predecessor ~~have~~ do any so merge both sibling & delete 77.

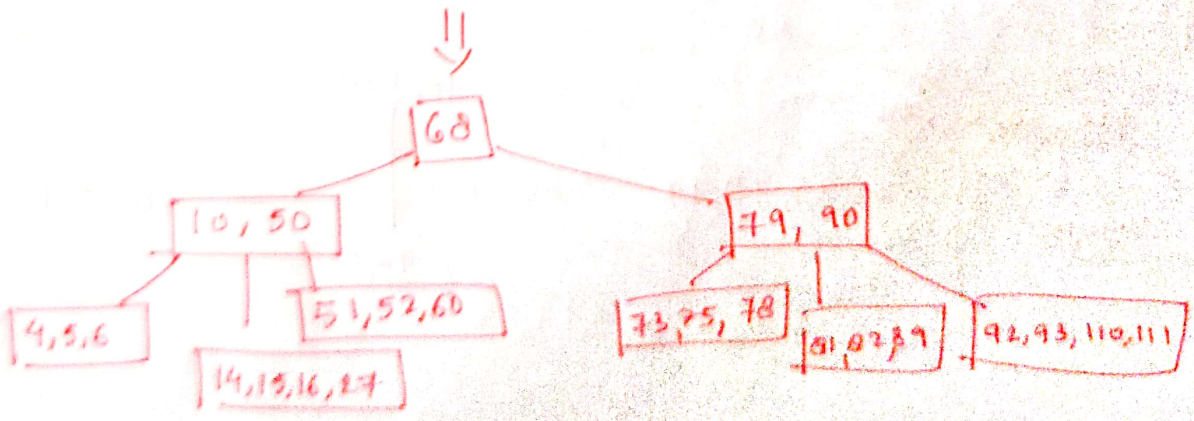


Now delete 80 → replaced it with largest value of inorder predecessor.



Now delete 100. → cant be replaced with ~~any order~~ inorder successor or predecessor..



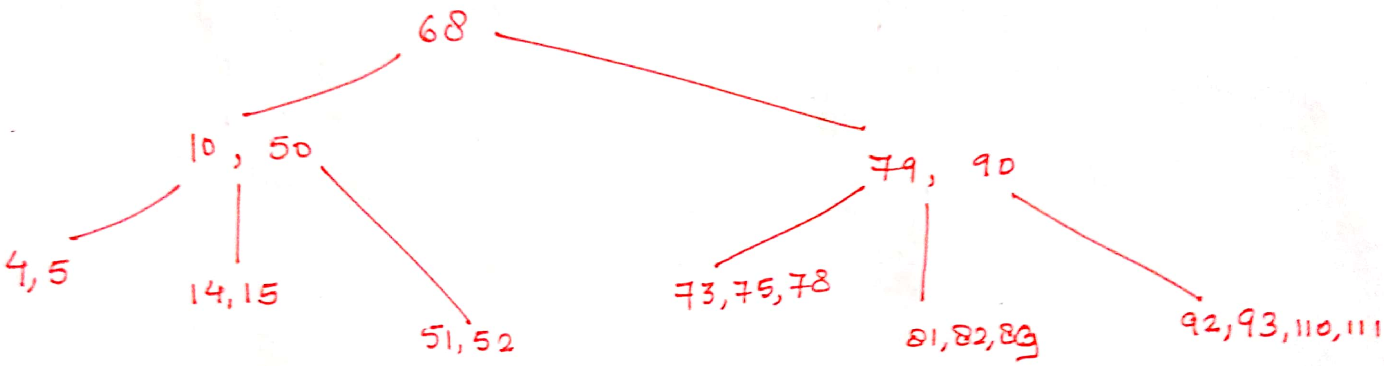


now delete 6. – directly delete.

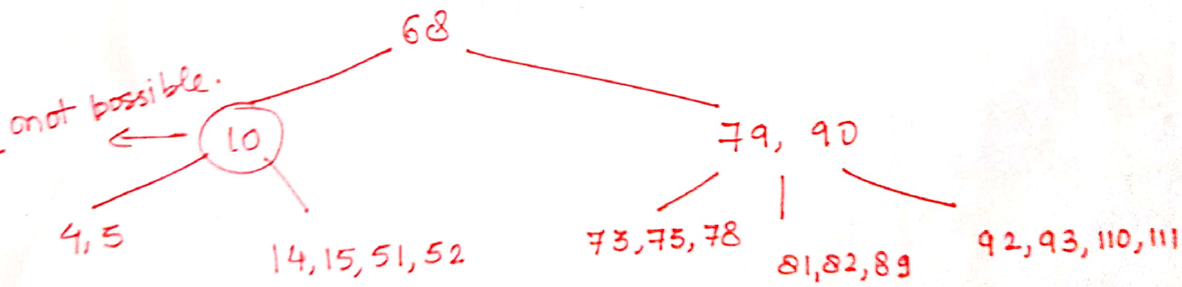
now " ~~27~~ " "

" " 60 " "

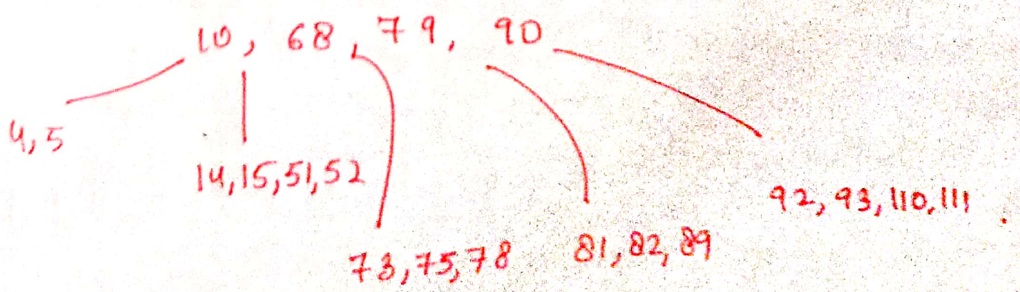
" " 16 " ".



now delete 50



so ask it's sibling
but sibling also
doesn't have key so
merge with Parent.



Scanned with OKEN Scanner