B^+ Tree

- Quick Access
- Indexing (database indexes)
- multi-level
- Occupy less space than B Tree

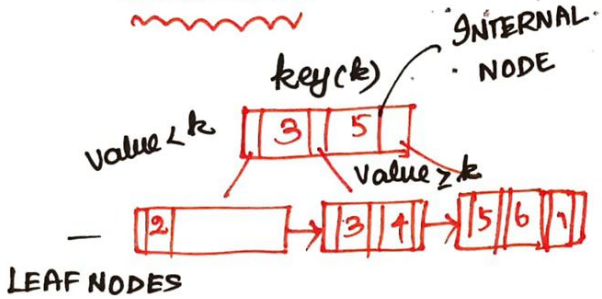
(i) Leaf nodes are actual data points

(ii) the leaf nodes are linked

Checklist / Rules

Order $m = 3$

STRUCTURE



(a) Maximum number of keys per node

$m-1 = 3-1 = 2$

$m=2$ ✓

(b) minimum number of keys per node

$\left[\frac{m}{2}\right] - 1 = 1$

$\left[\frac{2}{2}\right] - 1 = 1$ ✓

(a) left hand side

Value < key ✓

(b) Right hand side

Value ≥ key ✓

(c) Maximum number of pointers / childrens per node $m=3$

(d) Minimum number of pointers / childrens Per node $\frac{m}{2} = 2$ $m=2$

Example : 2, 4, 7, 10, 17, 21, 28

→ Assume for the Example, only (3) Values allowed

Insert 2 — [2]

Insert 4 — [2|4]

Insert 7 — [2|4|7] — Three filled, then split because fourth element cannot be inserted

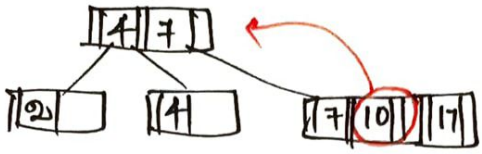
Insert 10 — [2|4|7] will be splitted as :

Pick middle element

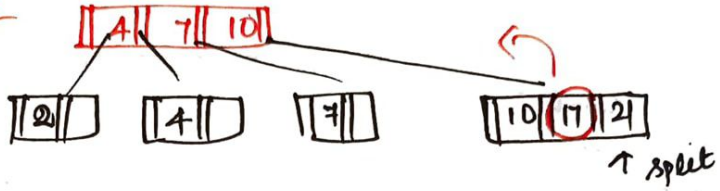
Insert 17

Insert 21 : Since already, values of 3 filled, split by finding the middle element

From the previous state,

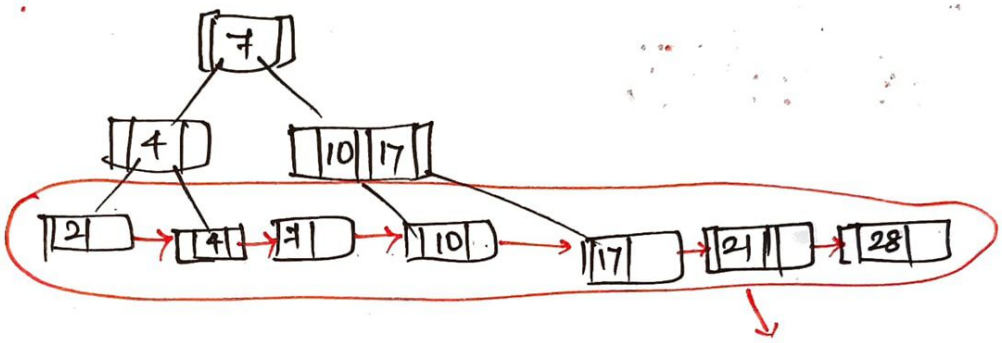


already filled Pick middle



↑ split

Insert 28 :



Elements are inserted

B^+ Tree insertion Example order = 3

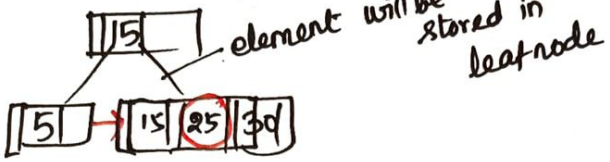
5, 15, 25, 30, 45, 60, 18, 28,

Insert 5 :

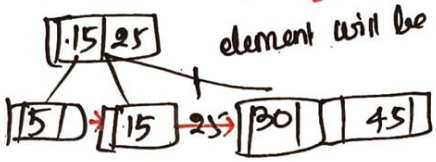
Insert 15 :

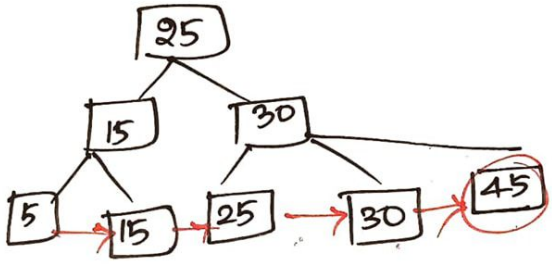
Insert 25 :

Insert 30 : not possible, therefore split by finding the middle element



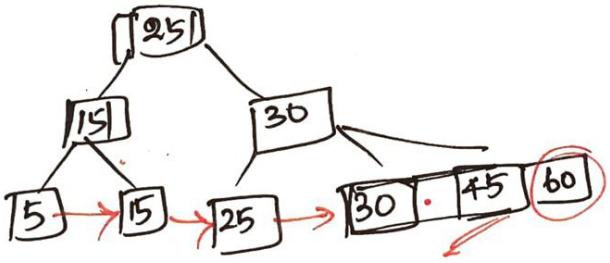
Insert 45 : not possible, split by finding the middle element





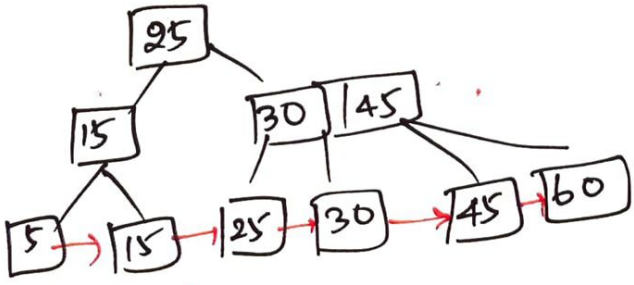
added

Insert 60 :

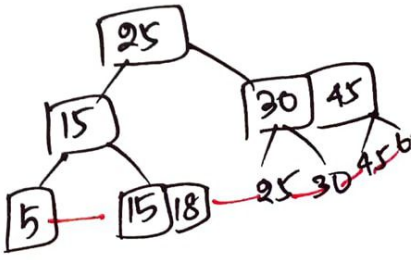


added

split



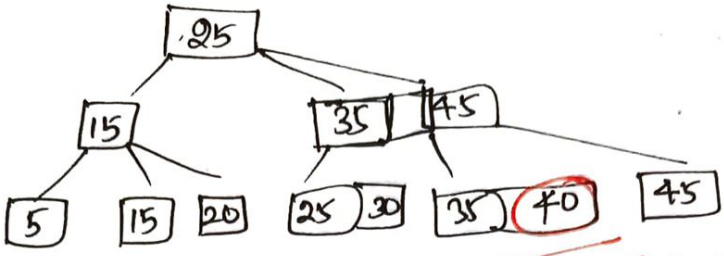
Insert 18



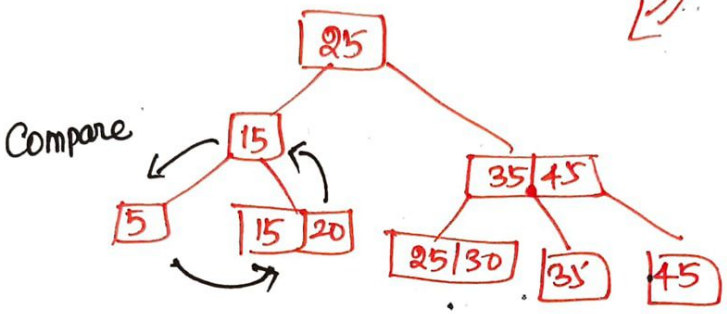
Proceed in the same way

B+ Tree Deletion

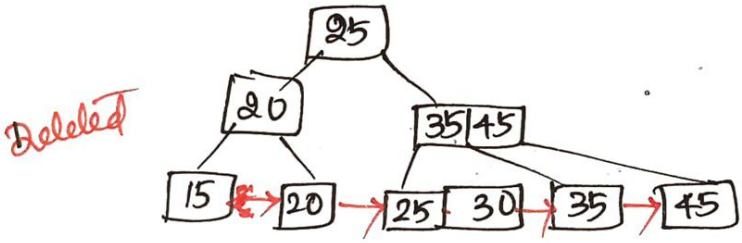
40 5 45 35 25



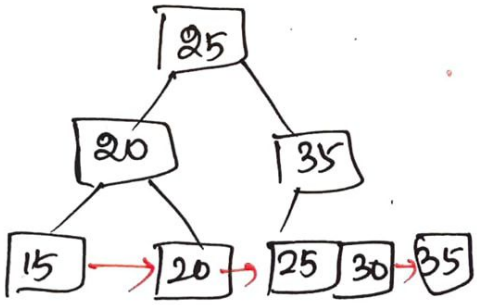
Delete 40 — mapped and deleted



Delete 5



Delete 45



Delete 35