B+ Trees

07 November 2020 17:07

B+ Trees

- Extension of B Trees
- Uses multi level Indexing
- In B tree dense index is used and in B+ tree Sparse index is used.
- As the index file grows we create index of index called multi level indexing.

- Properties of B+ tree are same as B tree

(A) (i) Maximum child pointers are m and m-1 keys for all nodes

(ii) Minimum child pointers should be $\lceil m/2 \rceil$ except root node which can have atleast 2 child pointers.

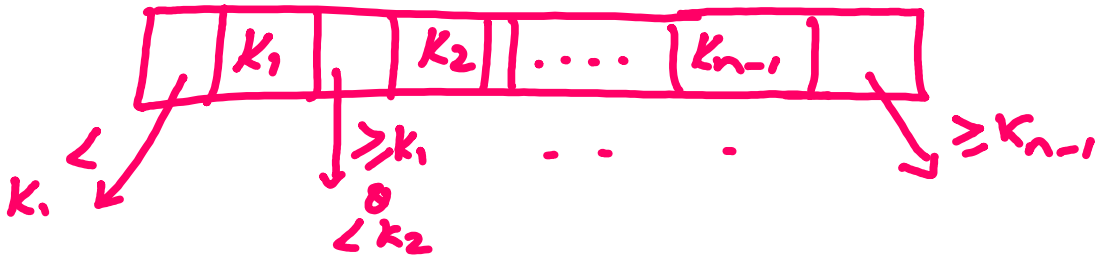
(iii) All leaf nodes are at

same level.

The differences with respect to B tree is as follows.

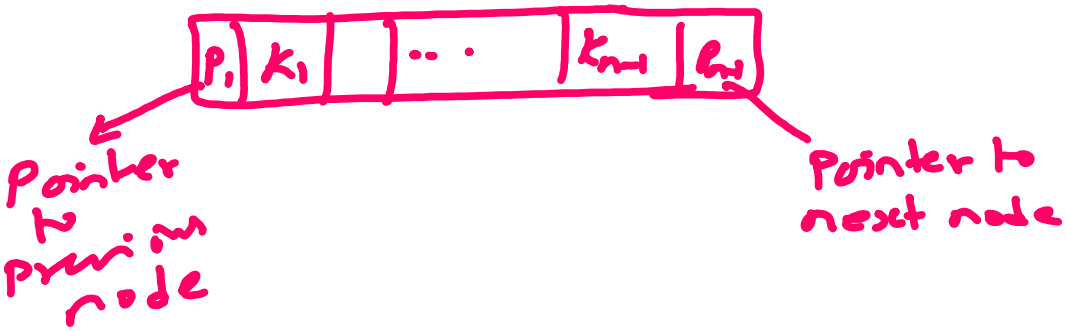
1) Non leaf nodes – Index Pages
   Leaf nodes – Data Page

2) Node structure.



(ie) We have $\geq$ keys in Right side of node. (In B tree > only)

3) Leaf node.



(ie) All the nodes in leaf levels are connected.

Hence 'Range Search' is possible.

Insertion: Example

Insert the following key into a B+ tree in sequence.

6, 16, 26, 36, 46.

Given the order m=3.

Any

Given $m = 3$

$\therefore$ 6 = min Child Pointer

$= \lceil 3/2 \rceil = 2$

$\therefore$ Max keys $= m-1 = 3-1 = 2$

Min keys $= t-1 = 2-1 = 1$

Step 1: Insert 6

Step 2: Insert 16

Step 3: Insert 26

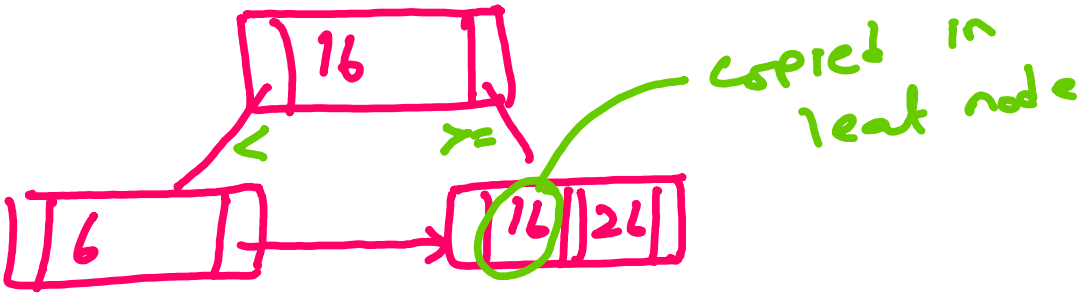
Node is full.

Virtual insertion [6 (16) 26]

Median is 16.

Split and move median [6] [26]
to parent.

Parent is not available,
hence create new root.



Step 4: Insert 36

Search for 36.

$36 \geq 16$. Hence Traverse right side of 16. In leaf node [16 26] we have to insert 36.

But [16 26] is full.

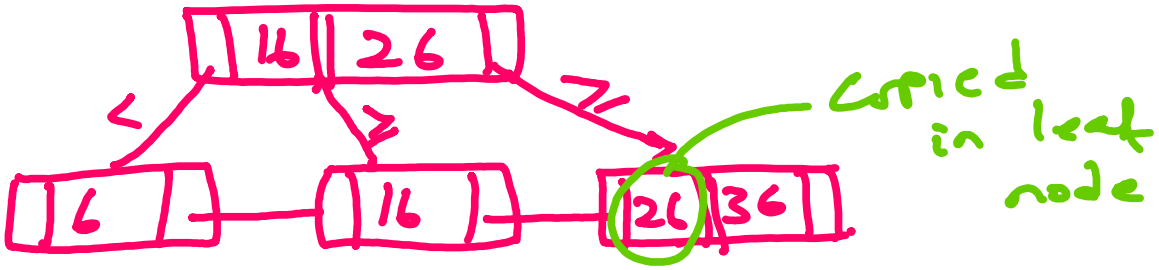
Do virtual insertion, find median, split, move median to parent.

[16 (26) 36]

26

[16] [36]

Now,



Step 5: Insert 46

46 to be inserted in the node [26 36].

Node is full. Do virtual insertion

[26 36 46]

Find median, split, move median to parent.

36

[26] [46]

Insertion 36 in Parent node [16 26]

Parent node is also full.

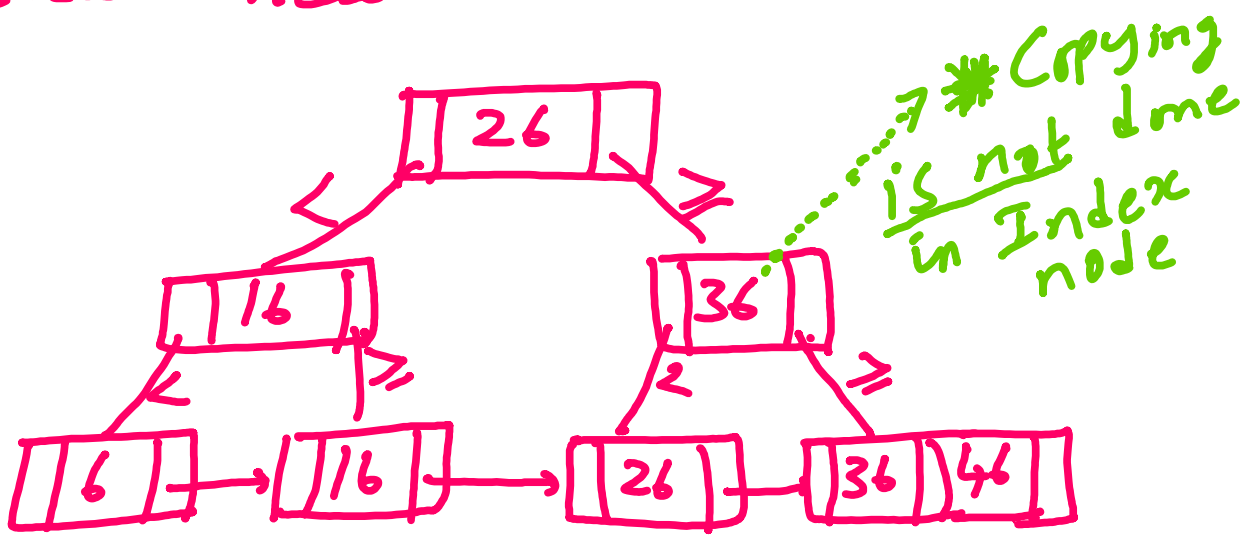
Again Virtual insertion.

[16 (26) 36]

[16] 26 [36]

Find median, split, move median to Parent. No Parent, Hence

Create new root.



* As we note, when we split and more, median is copied only in leaf node and not in index node.

Example 2:

Insert 1, 3, 5, 7, 9, 2, 4, 6, 8, 10 of order $m = 4$ in B+ tree

Ans:

$m = 4, \quad t = \lceil 4/2 \rceil = 2$

Max Child Pointer $= m = 4$

Max Key $= m - 1 = 3$

Min Child pointer = $t = 2$
Min key = $t-1 = 1$

Insert 1, 3, 5

Insert 7

For split we follow the rule as, Virtual Insert 7

[1 3 5 7]

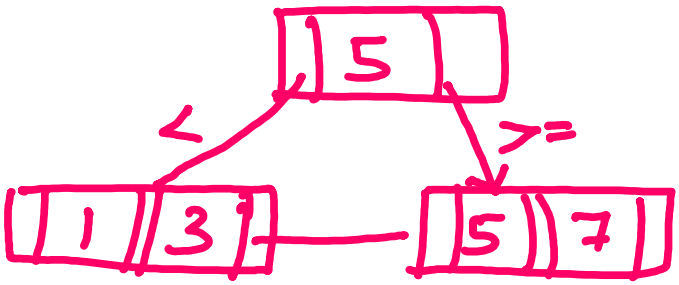
Middle element is $n/2 + 1$

(i.e) $4/2 + 1 = 3^{rd}$ element

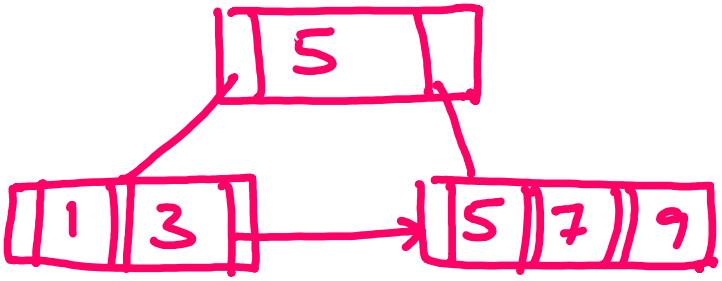
[* In B tree, we follow split first then insert in case of even number of elements]

Split, move median to Parent, Copy median to Right side.
[Copying is done because we are Splitting leaf node]

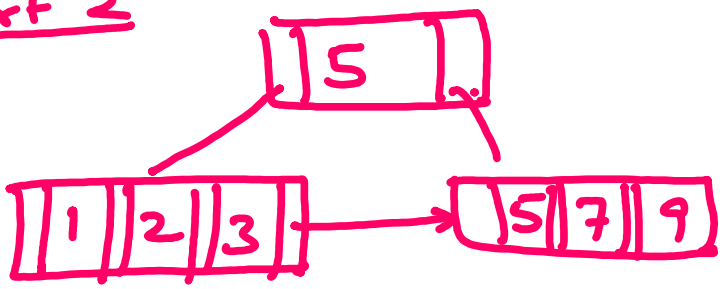
Result is,



Insert 9



Insert 2



Insert 4

Correct node of insertion is

[1 2 3]

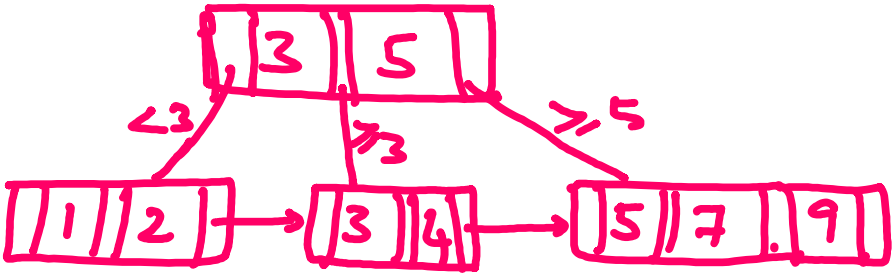
Node is full.

Virtual insert

[1 2 3 4]

Median = $4/2 + 1 = 3^{rd}$ element

Split, Push median to parent.
and copy median to Right Side.

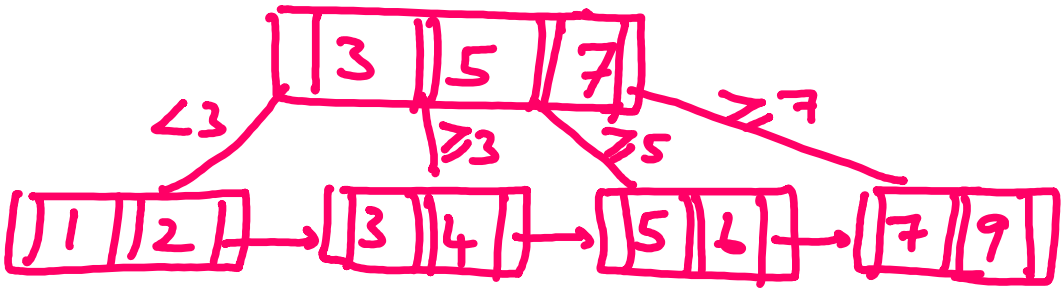


Insert 6

Correct node of insertion is

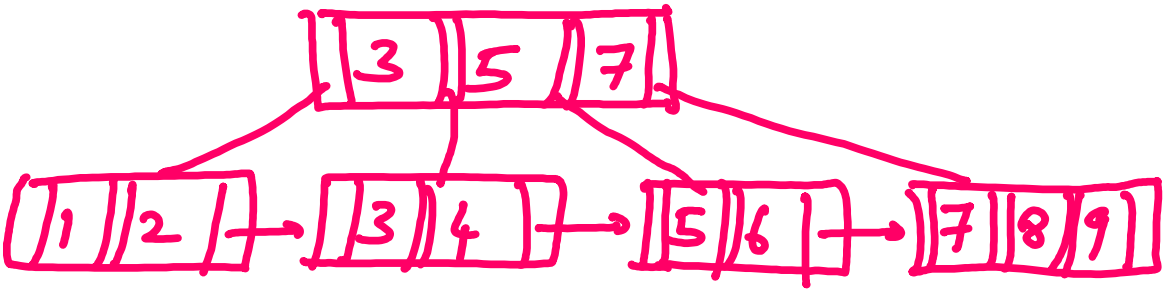
[5 7 9]

Node is full, find median
Push to parent, Copy to Right Side



Insert 8

Result in



Insert 10

Correct node of insertion is [7 8 9]

Find median split, move to parent & copy to right child.

9

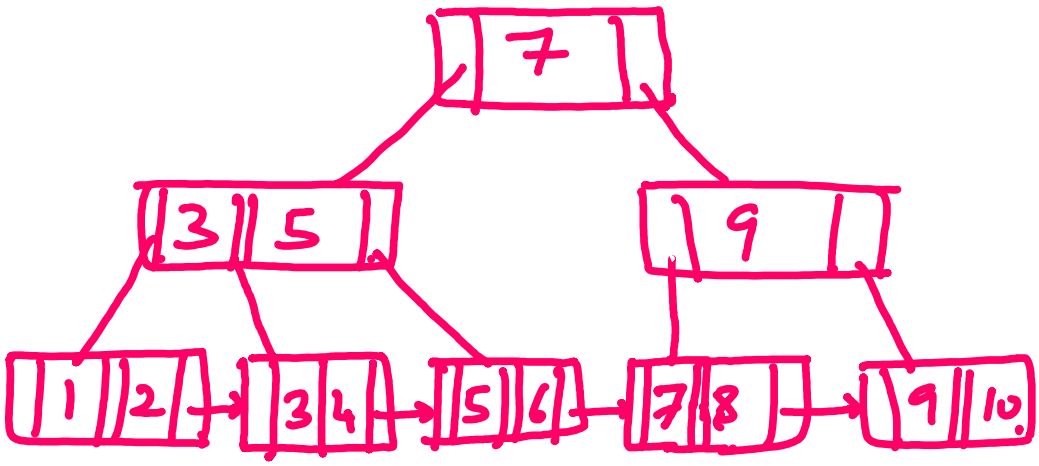
[7 8] [9 10]

Moving 9 to parent, Parent is also full, again split & move. (No copying because of non leaf)

7

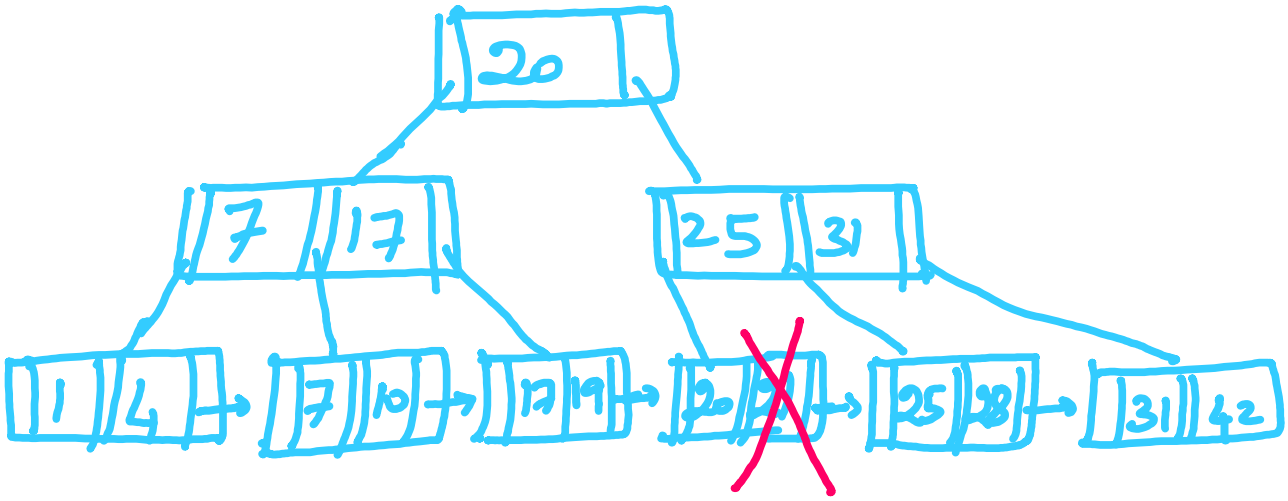
[3 5] [9]

∴ The result is,



Deletion : Example

Delete 21, 31, 20, 10, 7, 25, 42 sequentially from the following B+ tree of order m=4.



Delete 21

Given m = 4

min child = $t = \lceil 4/2 \rceil = 2$

min key = $t - 1 = 1$

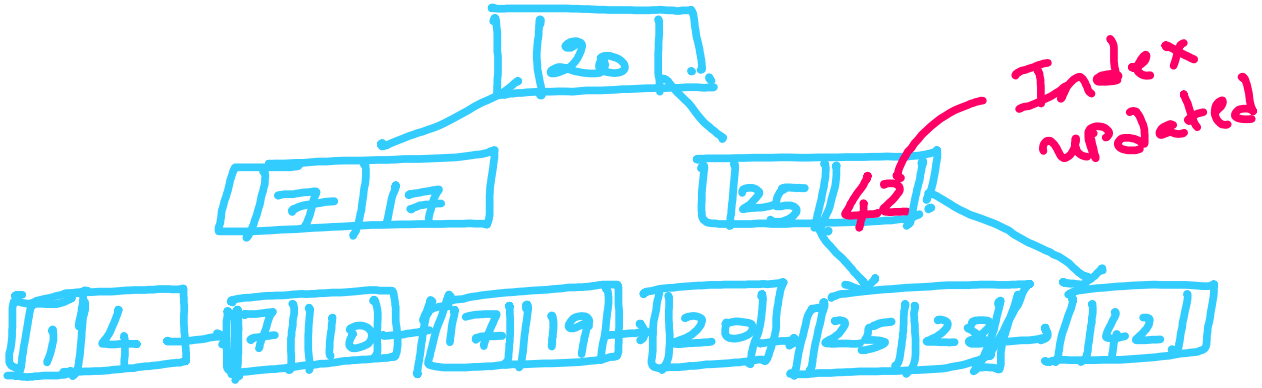
Deletion of 21, we have minimum key available, and key is not available in index page and hence simple deletion.

[shown in figure]

Delete 31

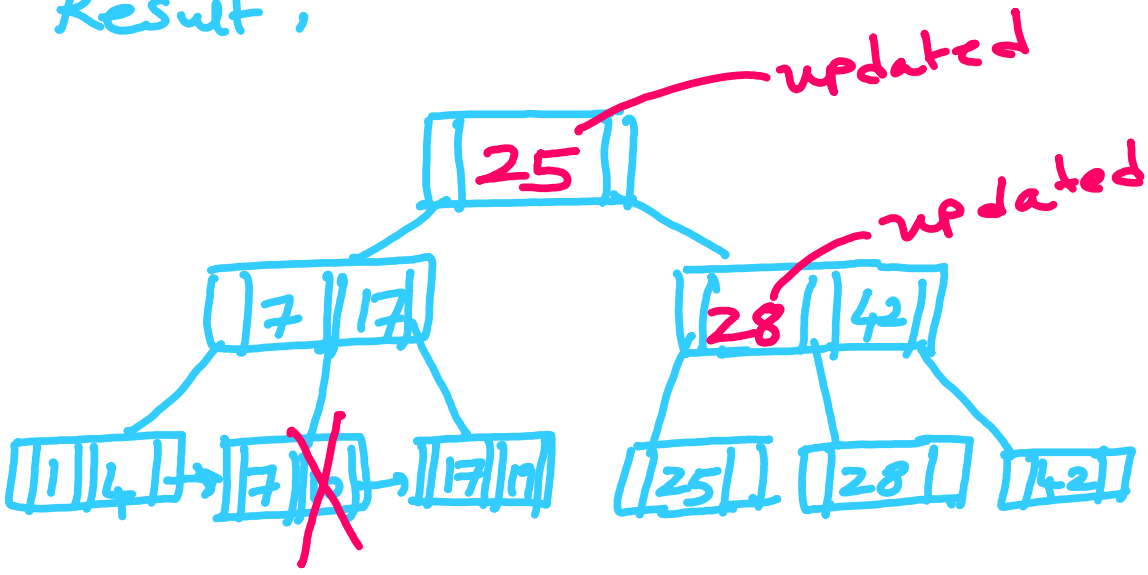
Min key available

Min key violation
Key available in index Page
Replace the index with next key.



Delete 20

Min key violation
Borrow from Left / Right
Here we borrow from Right.
Result,



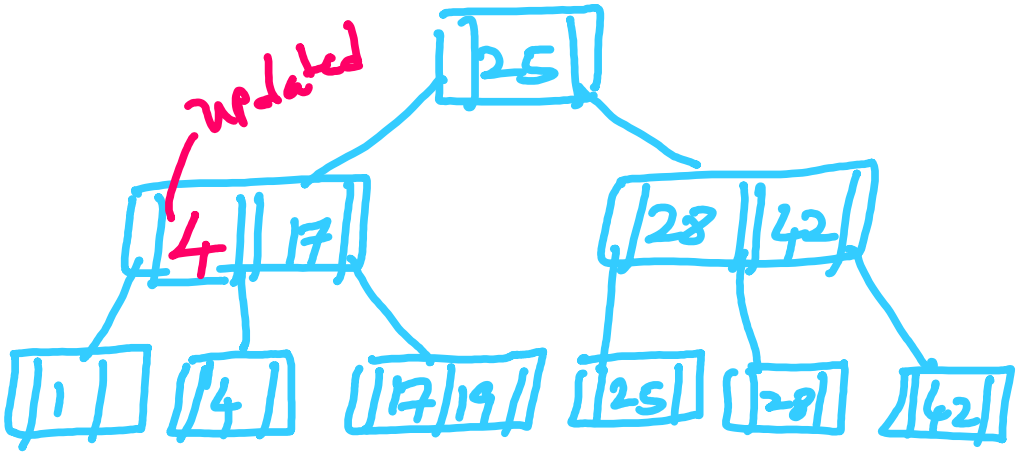
Delete 10

No min key violation
Not available in index Page

Simple deletion.
Shown in above figure

Delete 7

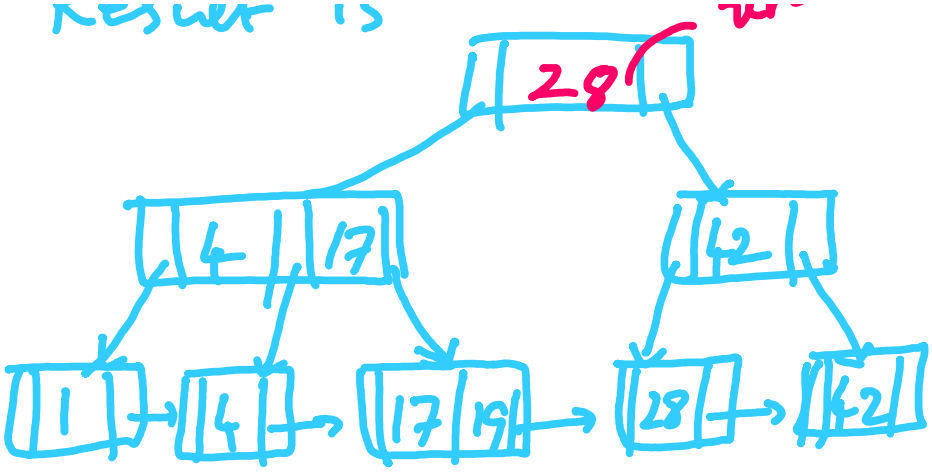
Min. Key Violation
Borrowing possible from Left Side
Result is,



Delete 25

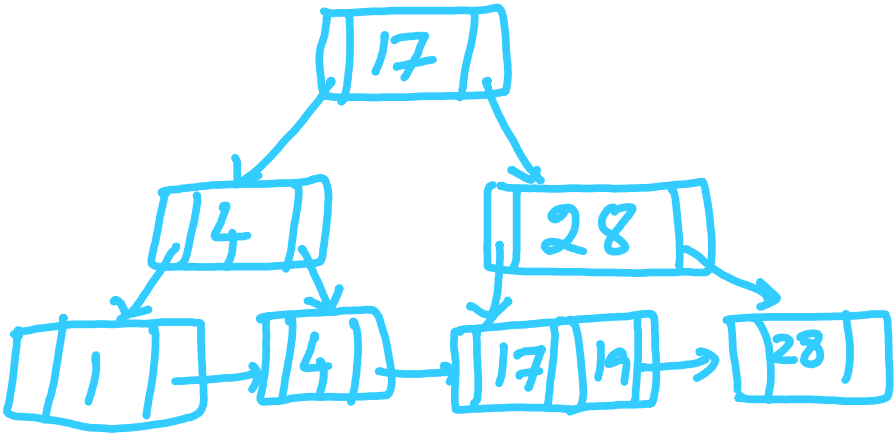
Min key violation.
Borrowing Not possible.
Merging with right Sibling and removing intervening Parent element.
[Note: For Merging Leaf nodes, Intervening Parent is not included]

Result is



Delete 42

Minimum Key Violation.
Borrowing not Possible.
Remove intervening Parent element results in min key violation. Hence Borrow from left Sibling. (Borrowing possible)
Result is,



Delete 4

Min key violation.
Borrowing not possible.
Merge with left sibling.

Since Leaf node merge, remove intervening Parent element.

(e) 4. But results again with min key violation. Borrowing not possible. Hence Merge.

Because of non leaf nodes merging, merge is done along with intervening Parent element.

Result is,