Construct a B+ tree for the set of key values.

(2, 3, 5, 7, 11, 17, 19, 23, 29, 31)

Assume that the tree is initially empty and values are added in ascending order. Construct B+ tree for the case where the number of pointers that will fit in one node is four. Also show the form of the tree after deletion of 23. [CPU: 2015 Spring].

Soln:

Construction of B+ tree for order n=4:

As we know,

→ Internal nodes (non-leaf nodes) must hold at least $\lceil n/2 \rceil$ and at most n children. (Pointers to children)

Here,

for order n = 4

minimum Pointers in internal nodes $= (4/2) = 2$

maximum Pointers in internal nodes = 4

Similarly,

→ Each leaf node must hold between $\lceil (n-1)/2 \rceil$ and (n-1) value.

Now,

Minimum number of search key values

$= \frac{(4-1)}{2} = 3/2$

$=$ ceiling (1.5)

$= 2$

(Node gets merged if count goes below in particular node)

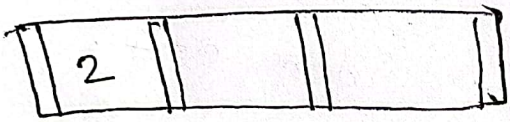
∴ maximum number of search key values = 3

Here, Nodes get split when number of search key values reaches 4 in particular node.

→ If root node is not a leaf it has at least 2 children.

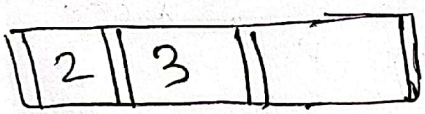
Step 1

Insert key value 2



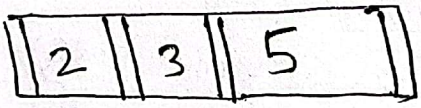
Step 2

Insert key value 3



Step 3

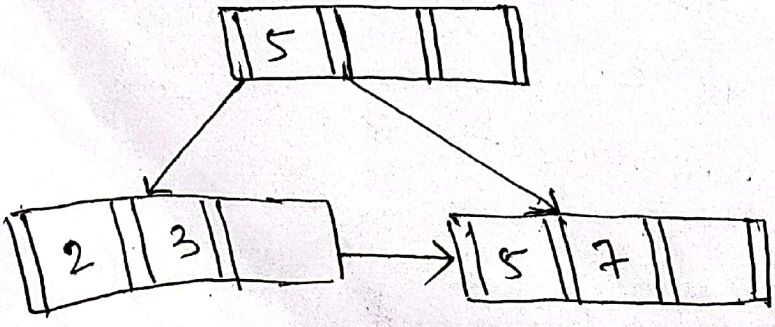
Insert key value 5



Step 4

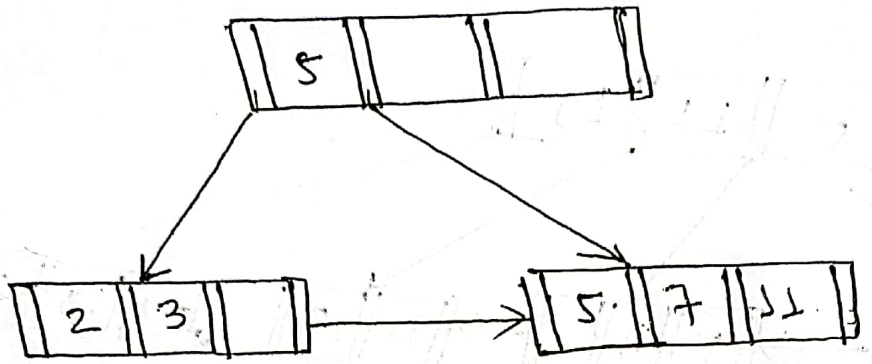
Insert key value 7,.

Now, Node becomes full, Splitting the nodes



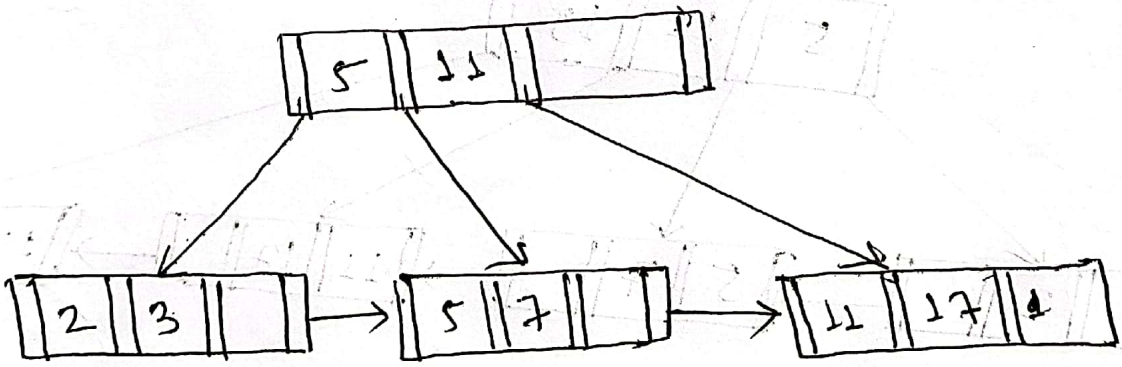
Step-5

Insert key value 11



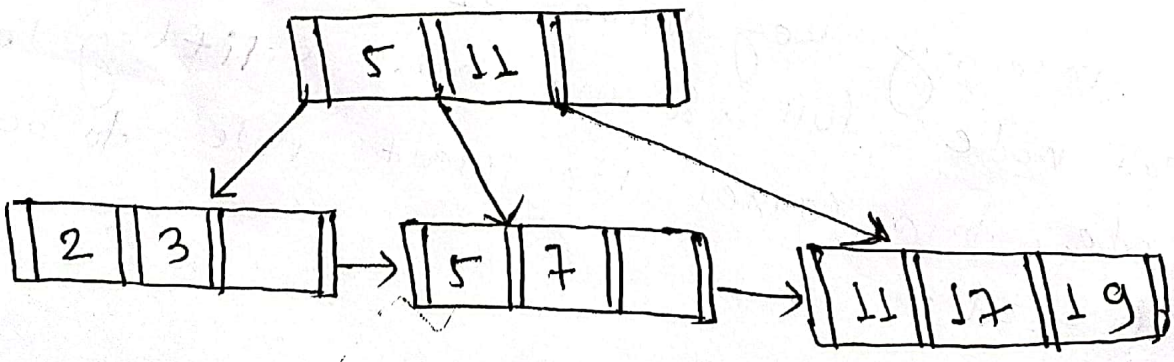
Step-6

Insert key value 17, Here second leaf node becomes full, Now splitting leaf node



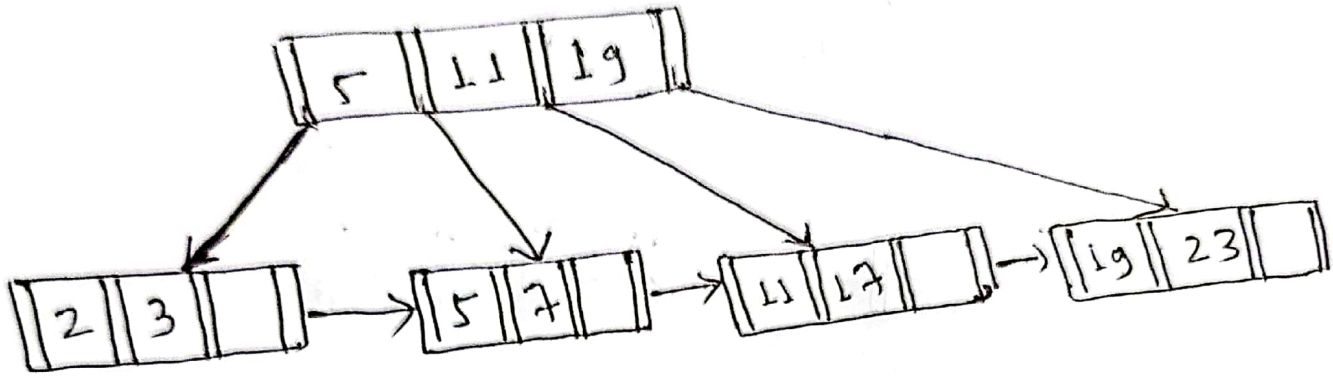
Step-7

Insert key value 19



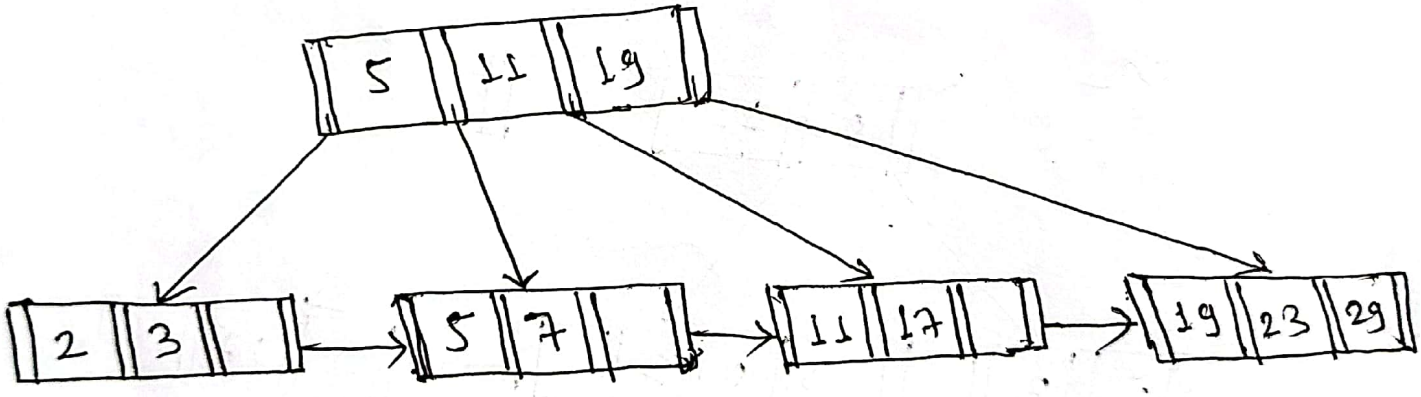
Step-8

Insert key value 23, Here Third leaf node is full, Now splitting leaf node



Step-9

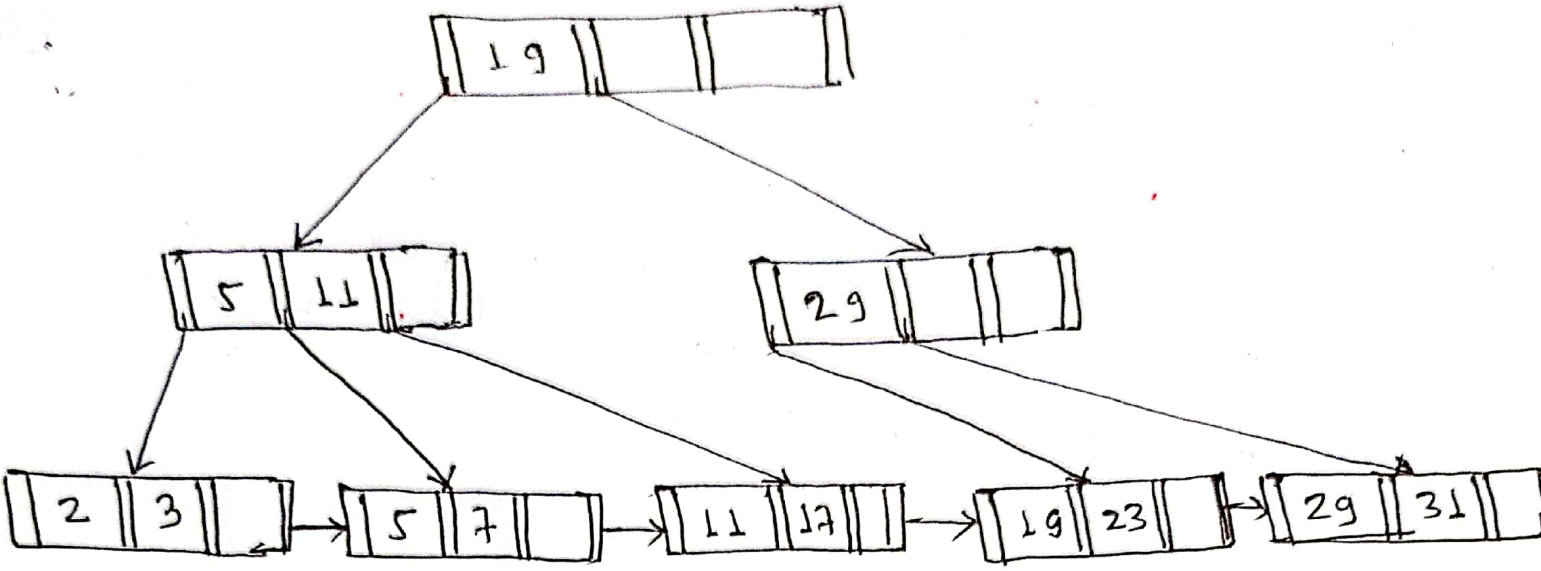
Insert key value 29



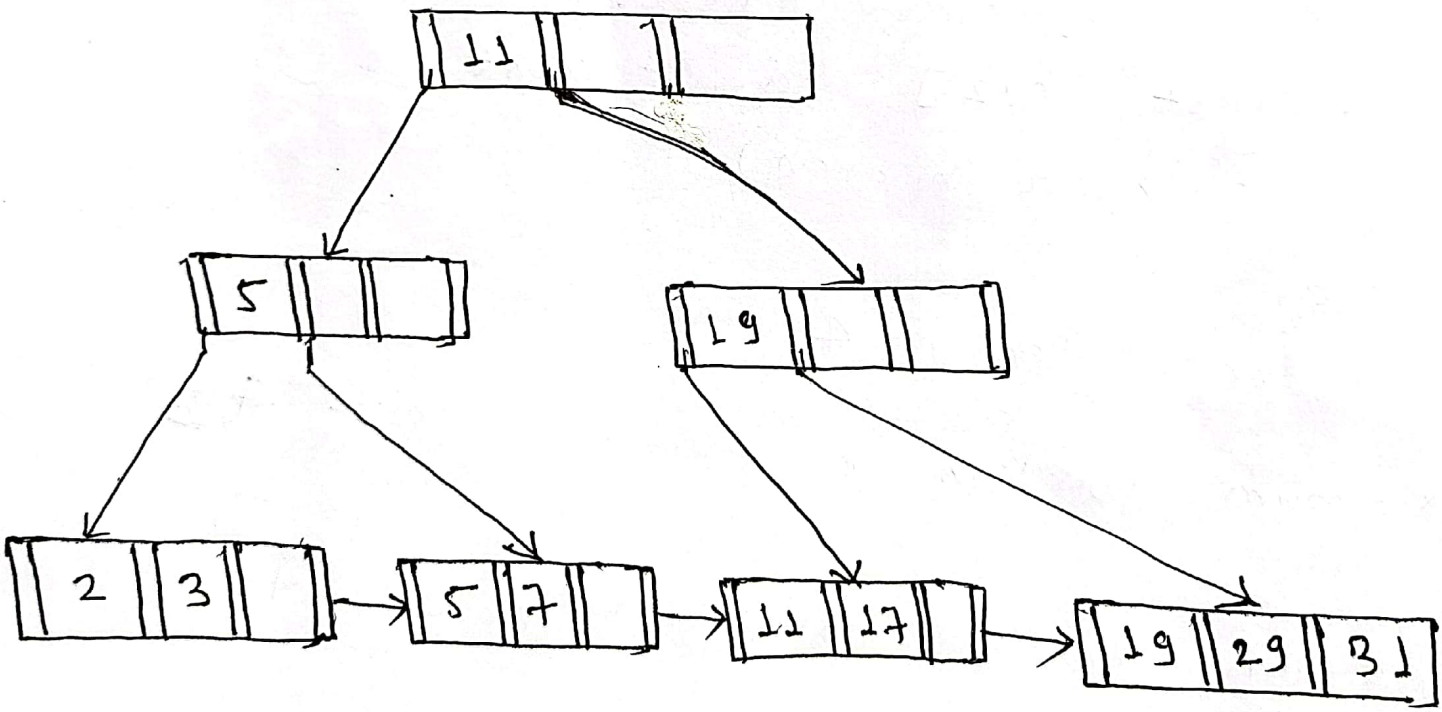
Step-10

Insert key value 31.

Here inserting key value 31 causes rightmost leaf node full, so we must splitting leaf node, this causes its parent node to be splitted.



After deletion 23



Create B+ tree for order 4 with following data.
(4, 9, 16, 25, 1, 20, 13, 15, 10, 11, 12)
of order 4. Assume that tree is initially empty and values are added in ascending order. Also show formation of tree after deletion of 16. CPU: 2019 spring]

→ Soln:

Construction of B+ tree for order n=4.

As we know,

→ Internal nodes (non-leaf nodes) must hold at least $\lceil n/2 \rceil$ and at most n children. (Pointers to children)

Here,
for order n=4

Minimum pointers in internal nodes = $(4/2)$
= 2

Maximum pointers in internal nodes = 4

Similarly,

→ Each leaf node must hold between $\lceil (n-1)/2 \rceil$ and (n-1) value

Now, Minimum number of search key values = $\left(\frac{4-1}{2}\right)$
= ceiling (1.5)
= 2

Node gets merged if count goes below in particular node.

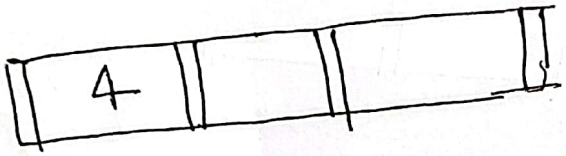
∴ maximum number of search key values = 3

(Node gets split when number of search key values reaches 4 in particular node)

→ If the root node is not a leaf, it has at least ^ 2 children.

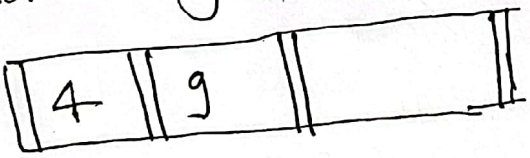
Step 1:

Insert key value 4



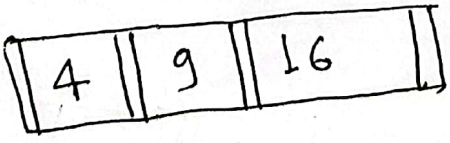
Step 2

Insert key value 9



Step 3

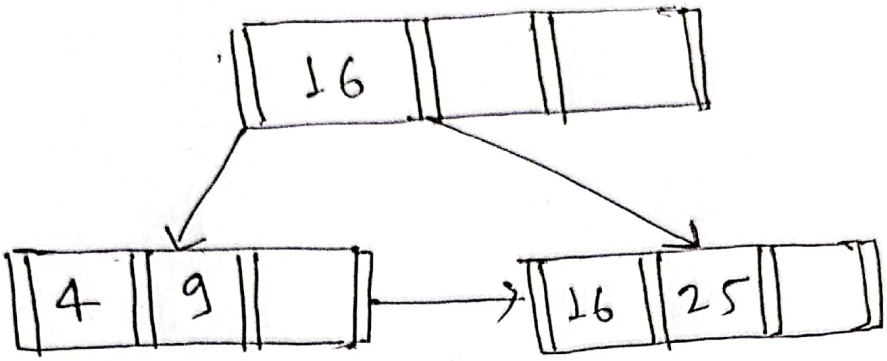
Insert key value 16



Step 4

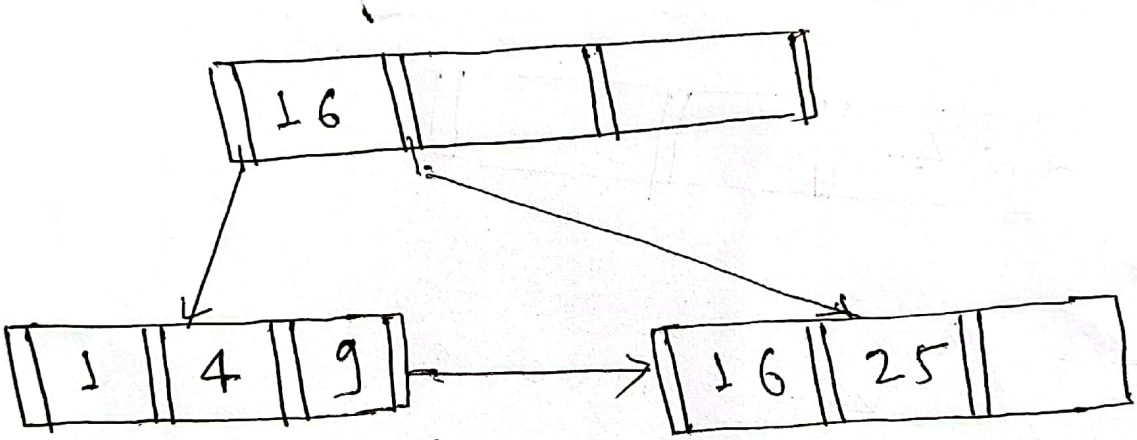
Insert key value 25

Now, Node becomes full, so splitting the nodes we get



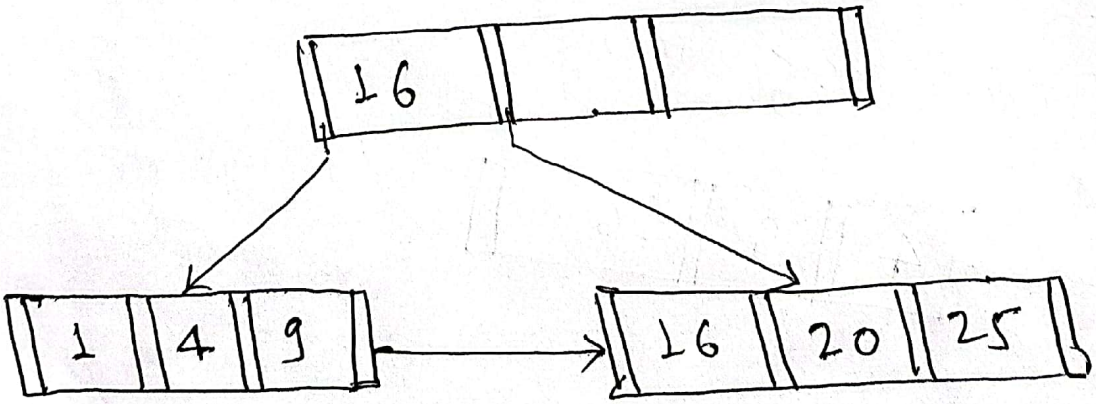
Step - 5

Insert key value 1



Step-6

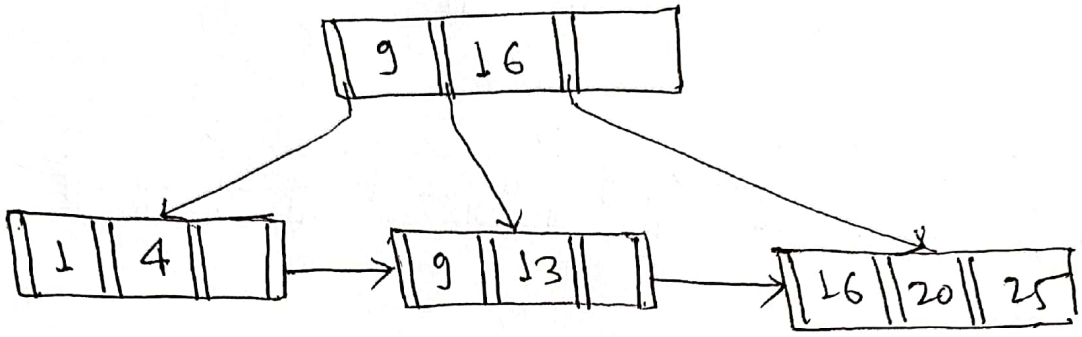
Insert key value 20



Step-7

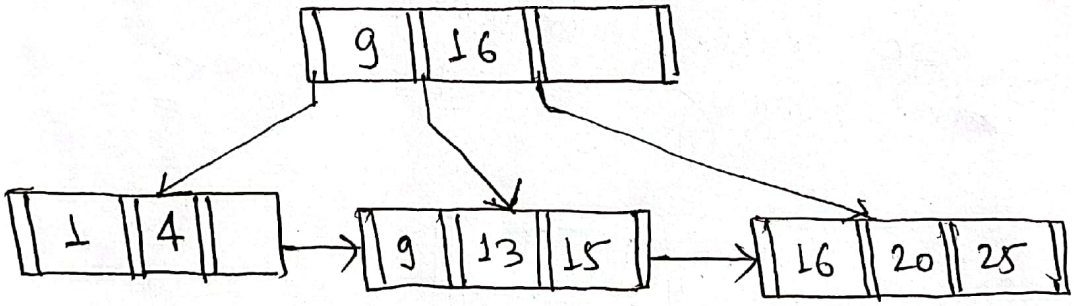
Insert key value 13,

Now, first leaf node is full, now splitting nodes



Step-8

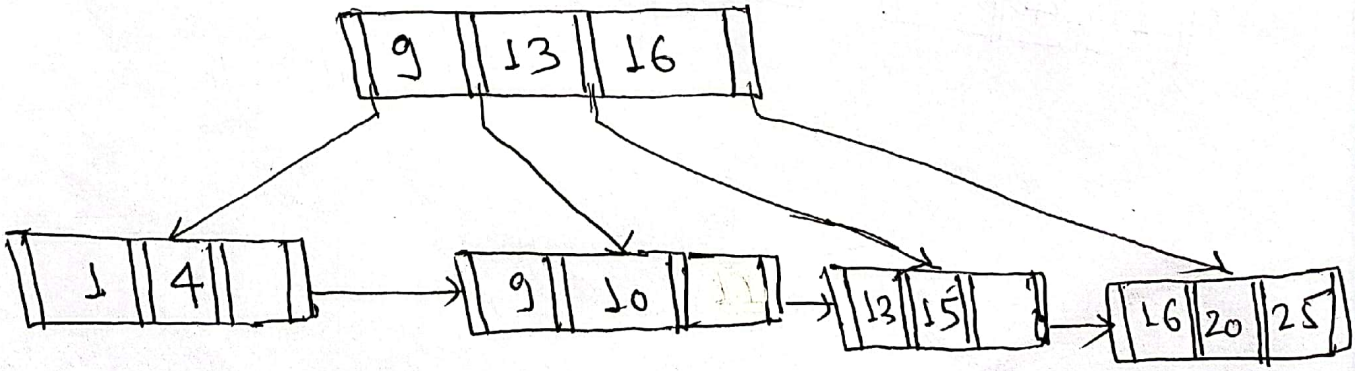
Insert key value 15



Step-9

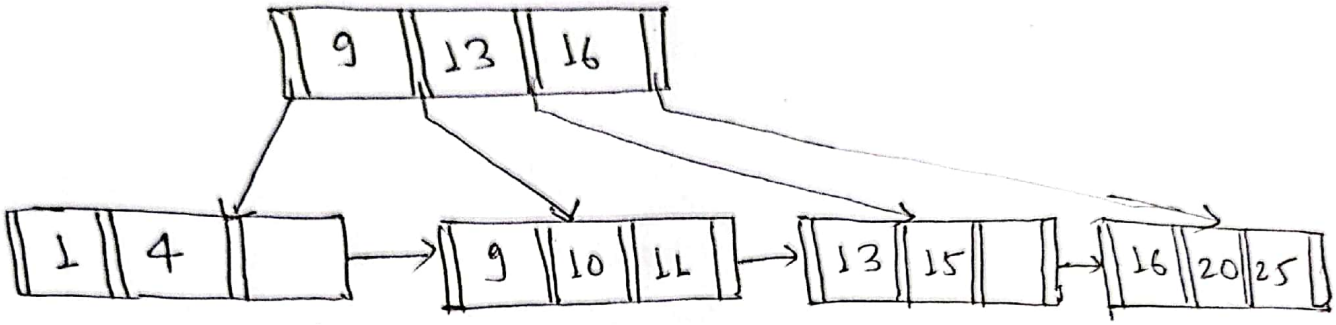
Insert key value 10

Now, second leaf node is full, Now splitting nodes.



Step - 10

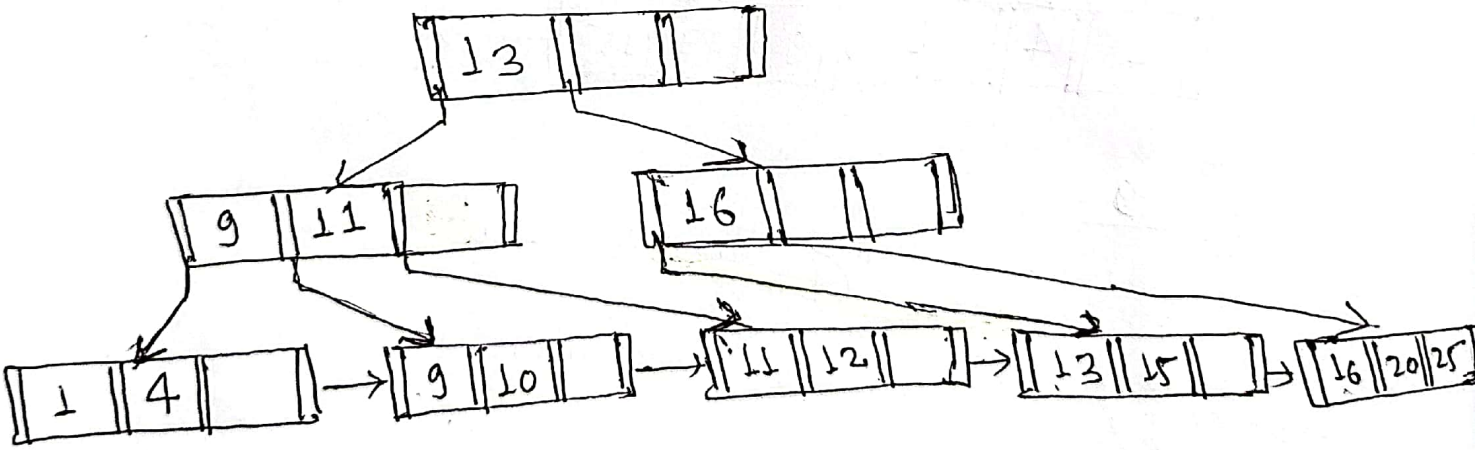
Insert key value 11



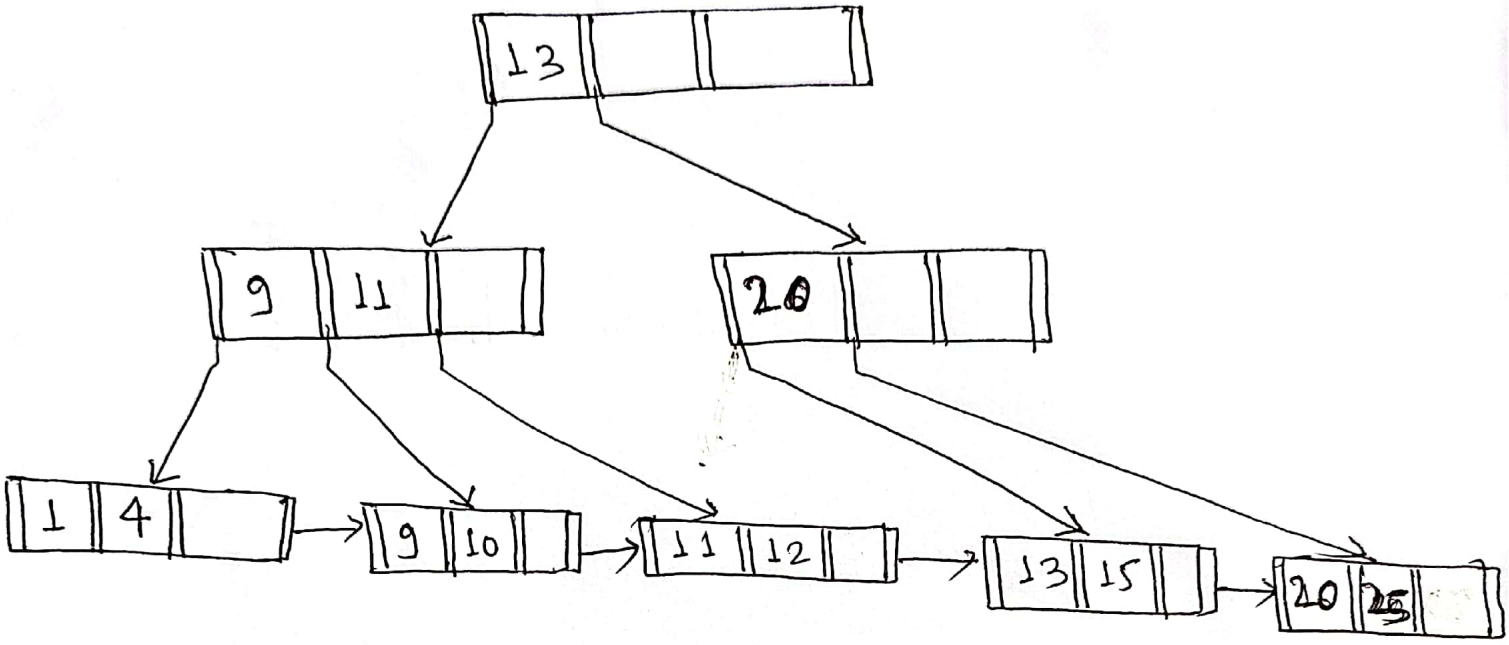
Step - 11

Insert key value 12

This causes overflow in 2nd leaf node. Splitting the leaf node causes overflow in it's parent node as well. finally B+ tree becomes.



After deletion of 16



Construct a B+ tree for the following set of key values. (1, 3, 6, 7, 11, 17, 19, 23, 30, 32). Assume that the tree is initially empty and values are added in ascending order. Construct B+ tree for the case where the numbers of pointers that will fit in one node is four. Also show the form of the tree after insertion of 9. [PU: 2018 fall]

Soln:

Construction of B+ tree for order $n = 4$.

As we know,

➔ Internal nodes (non-leaf nodes) must hold at least $\lceil n/2 \rceil$ and at most n children. (pointers to children)

So, for order $n = 4$

Minimum pointers in internal nodes $= (4/2) = 2$

Maximum pointers in internal nodes $= 4$

Similarly,

➔ Each leaf node must hold between $\lceil (n-1)/2 \rceil$ and $(n-1)$ value.

Now,

Minimum number of search key values

$$= \frac{(4-1)}{2} = 3/2$$

$$= \text{ceiling}(1.5)$$

$$= 2$$

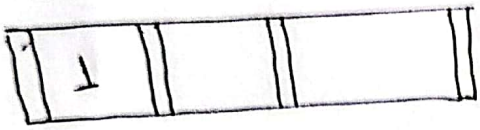
Node gets merged if count goes below in particular node

∴ Maximum number of search key values = 3 :

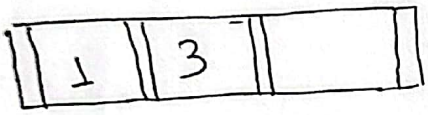
Here, Node yet split when number of search key values reaches 4 in particular node.

⇒ If the root node is not a leaf, it has at least 2 children

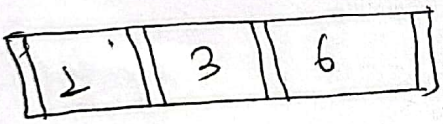
Step 1 Insert key value 1



Step 2 Insert key value 3

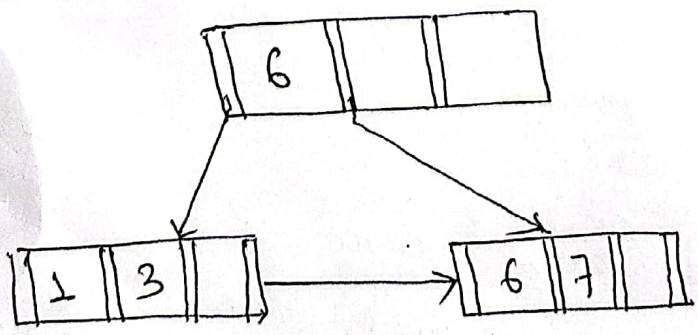


Step 3 Insert key value 6

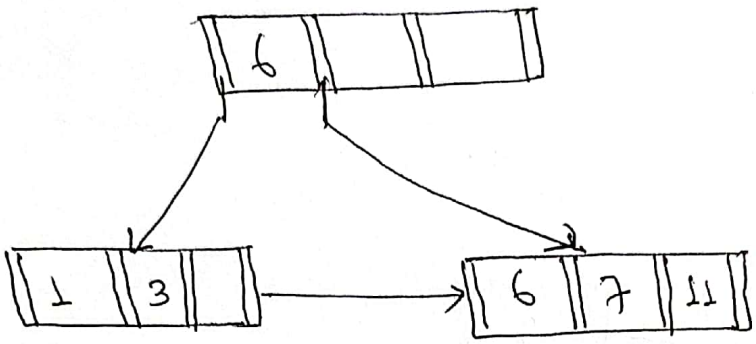


Step 4 Insert key value 7

Now, Node becomes full. Splitting the nodes

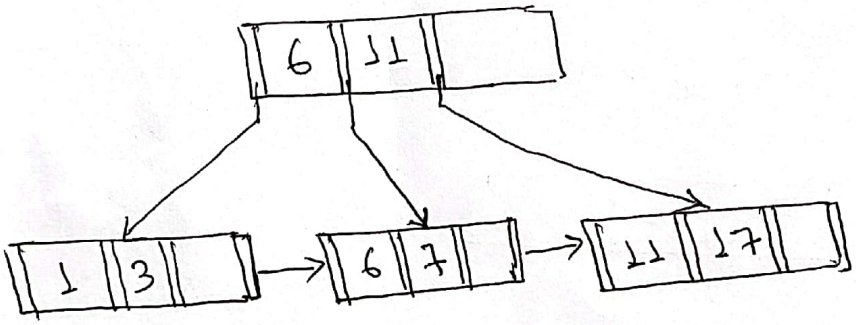


Step-5 Insert key value 11



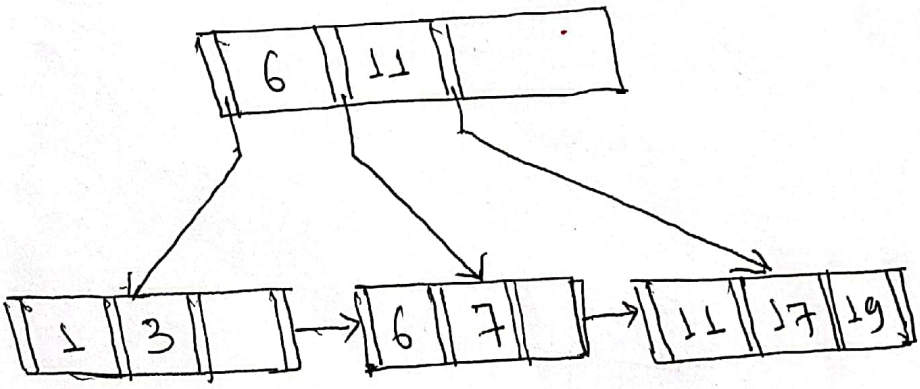
Step-6

Insert key value 17, Here, second leaf node becomes full, Now splitting leaf node.



Step-7

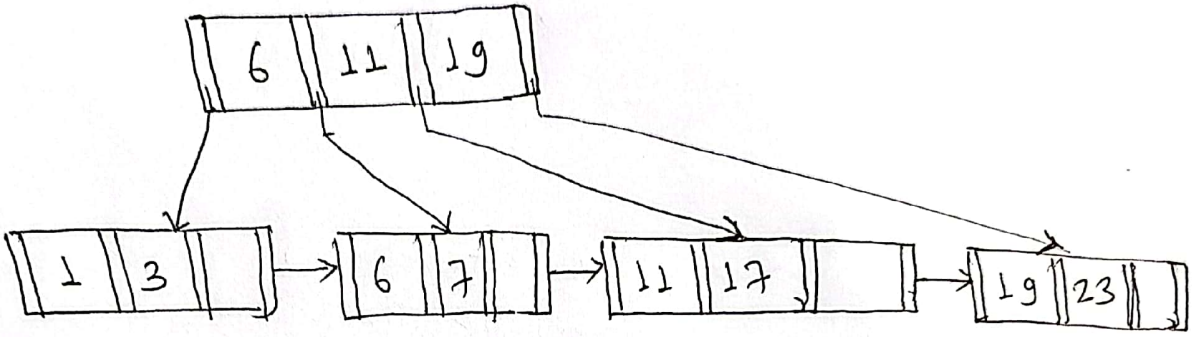
Insert key value 19



Step - 8

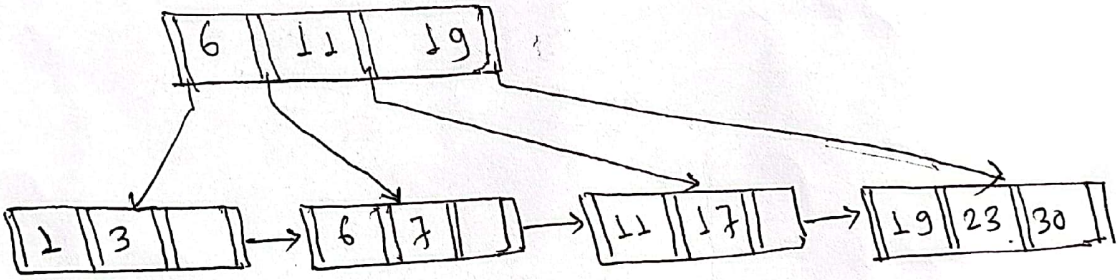
Insert key value 23

Here Third leaf node becomes full. Now splitting the nodes we get.



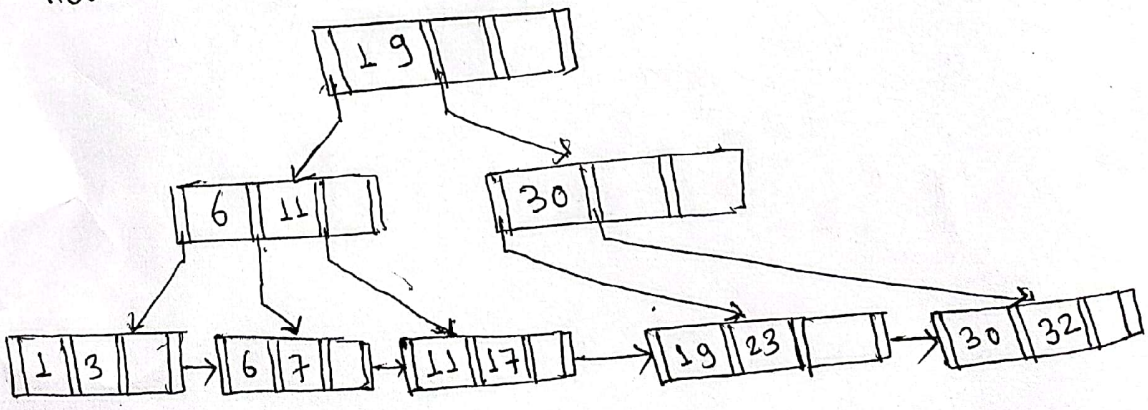
Step - 9

Insert key value 30



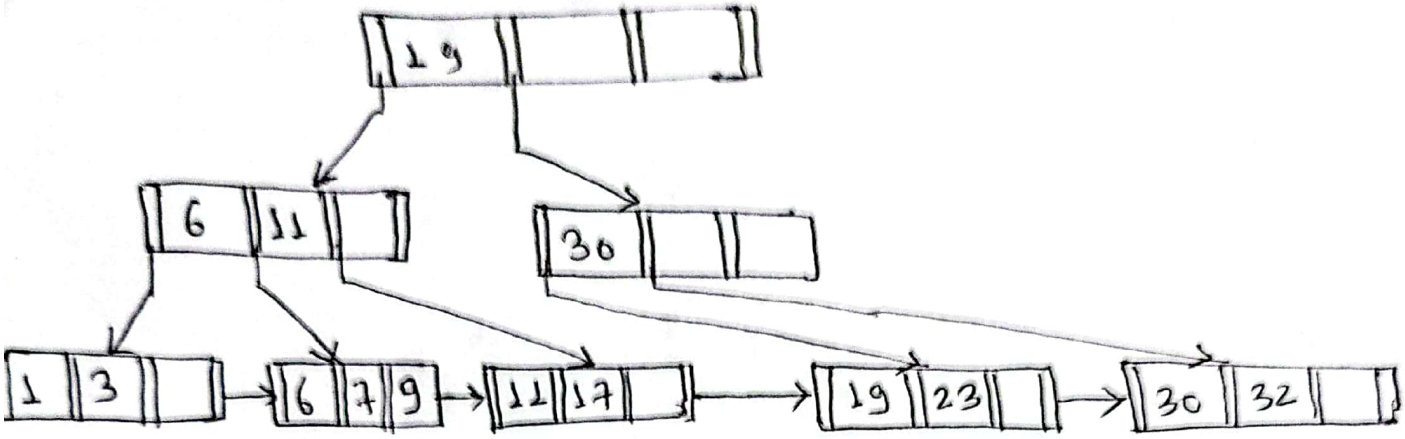
Step - 10

Insert key value 32, This causes rightmost leaf node full and splitting the nodes causes it's parent node also full, finally B+ becomes.



∴ After inserting 9

∴ B+ tree Becomes



Construct a B+ tree for the following set of key values (1, 4, 7, 10, 17, 21, 31, 25, 19, 20, 28). Assume that tree is initially empty and value are added in ascending order. Construct B+ tree for the case where the number of pointers that will be fit in one node is Four.

Soln:

Construction of B+ tree for order $n=4$,

As we know,

➡ Internal nodes (non-leaf nodes) must hold at least $\lceil n/2 \rceil$ and at most n children.
(Pointers to children)

Here, for order $n=4$

Minimum pointers in internal nodes $= (4/2) = 2$

Maximum pointers in internal nodes $= 4$

Similarly,

➡ Each leaf node must hold between $\lceil (n-1)/2 \rceil$ and $(n-1)$ value.

Now,

Minimum number of search key values

$= \left(\frac{4-1}{2}\right) = 3/2$

$=$ ceiling (1.5)

$= 2$

∴ Maximum number of search key values $= 3$

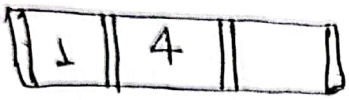
➡ If the root is not a leaf, it has at least 2 children.

Step 1

Insert key value 1

Step 2

Insert key value 4



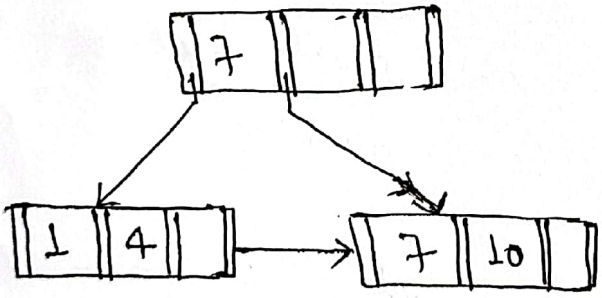
Step 3

Insert key value 7

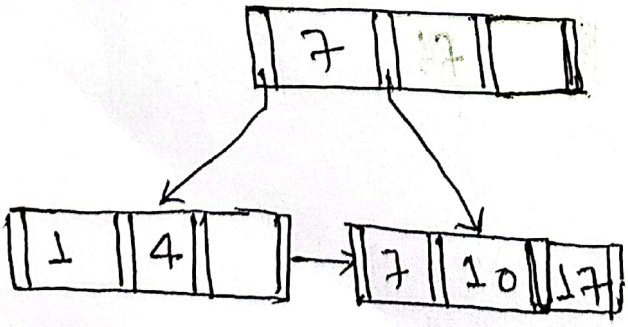
Step 4

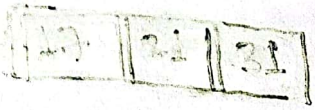
Insert key value 10

Now, node becomes full, splitting the nodes



Step 5 Insert key value 17

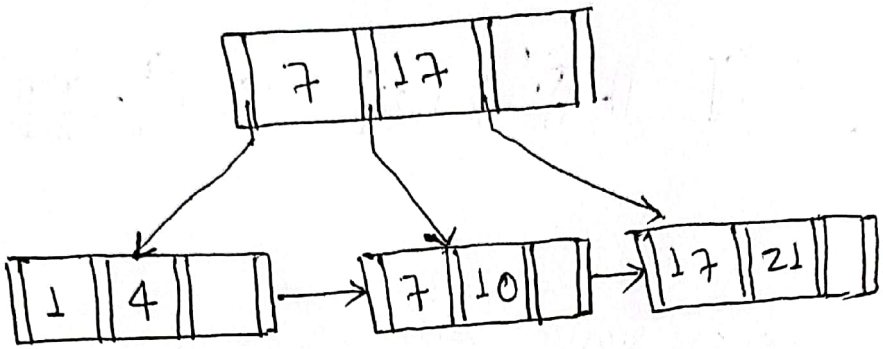




Step-6

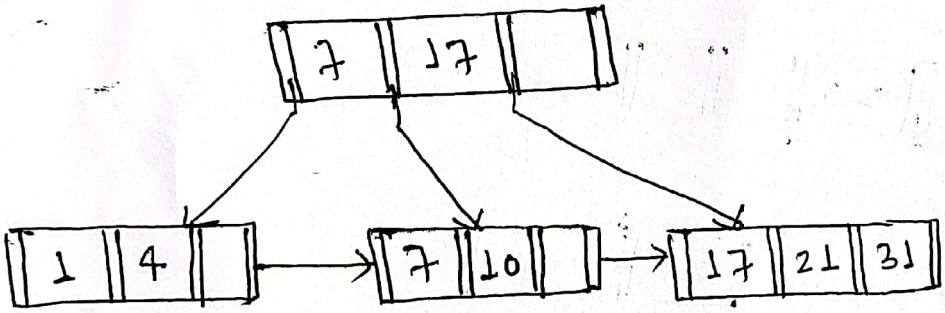
Insert key value 21

Now, Secord leaf node becomes full. Splitting the nodes we get.



Step 7

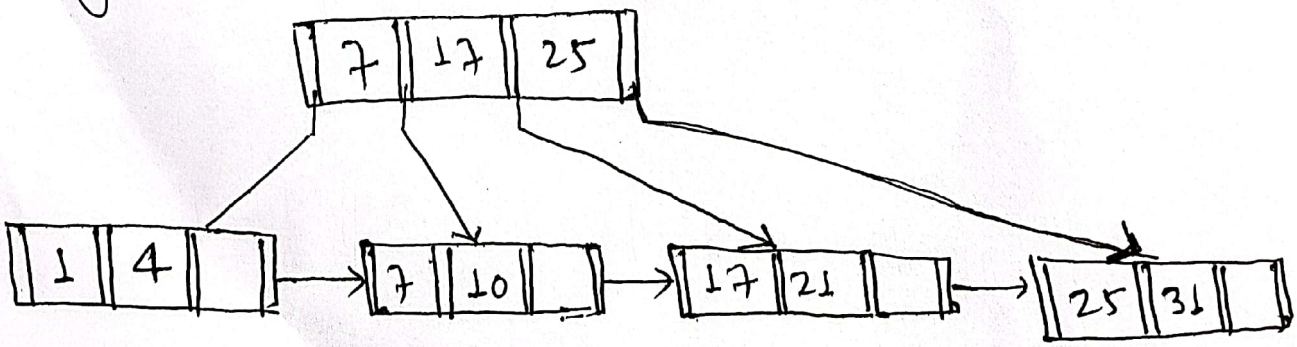
Insert key value 31



Step 8

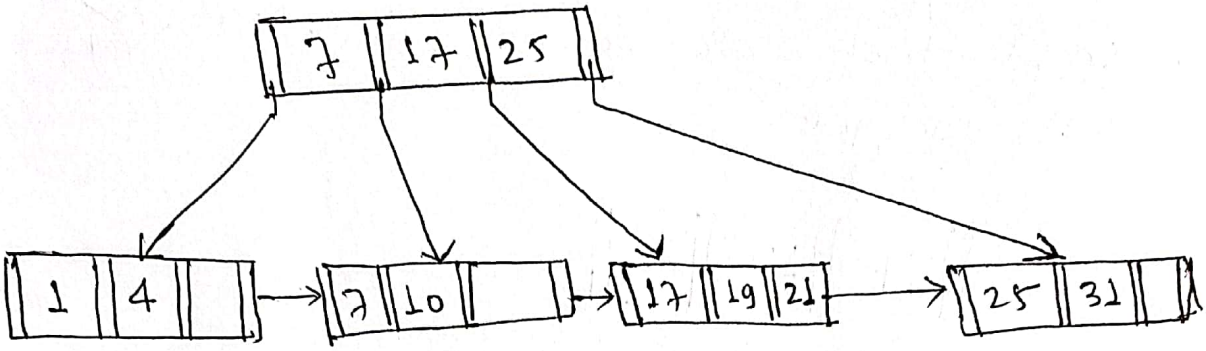
Insert key value 25

Now, Rightmost leaf node becomes full, Now Splitting the nodes we get



step-9

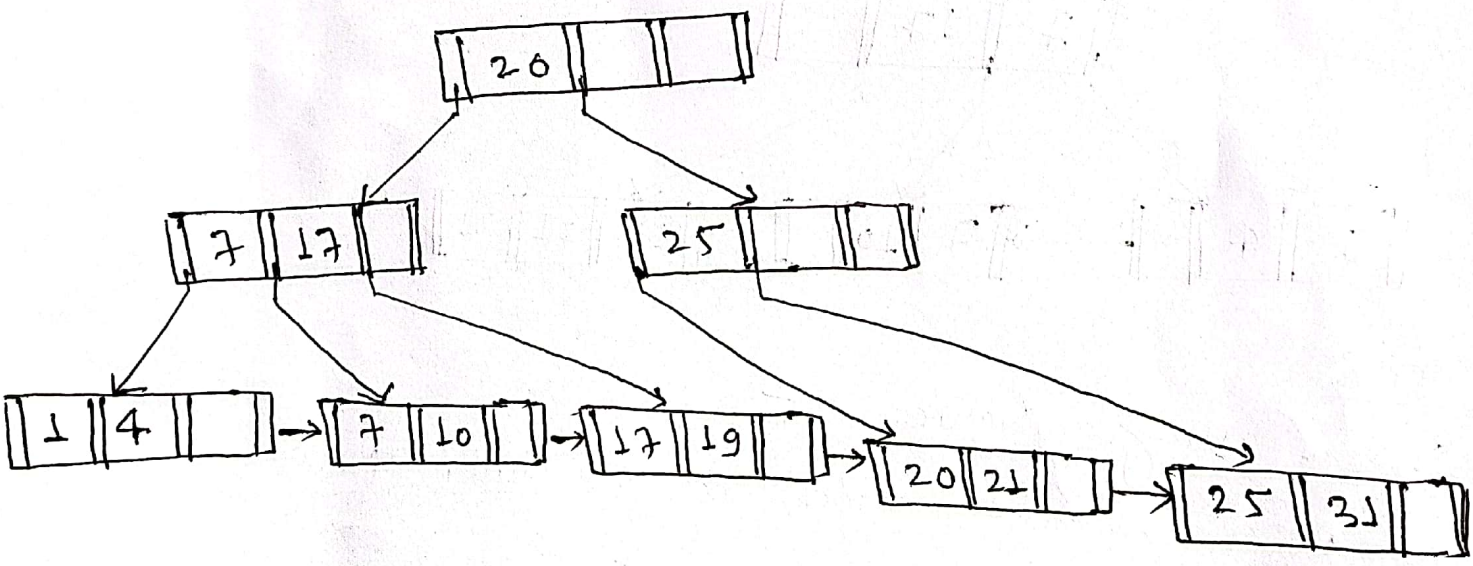
Insert key value 19



Step-10

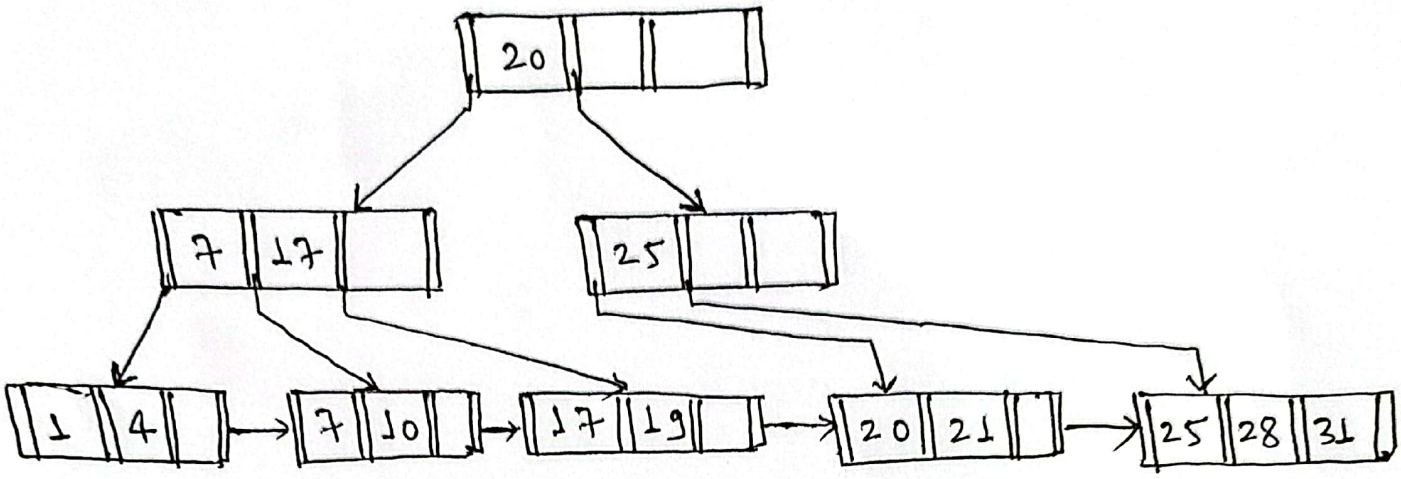
Insert key value 20.

Here, leaf node becomes full, splitting the ~~node~~ leaf nodes causes it's parent node full, Now again splitting parents node, B+ tree becomes.



Step - 11

Insert key value 28



Construct a B+ tree for the following set of key values (5, 15, 25, 30, 45, 60, 18, 28). Assume that tree is initially empty and value are added in ascending order. Construct B+ tree for the case where the number of pointers that will fit in one node is three.

Soln:

Construction of B+ tree for order $n = 3$.

As we know,

→ Internal nodes (non-leaf nodes) must hold at least $\lceil n/2 \rceil$ and at most n children. (pointers to children).

for order $n = 3$

Minimum number of pointers in internal nodes $= 3/2 = 1.5$

$= \text{ceiling}(1.5)$

$= 2$

Maximum number of pointers in internal nodes $= 3$

Similarly,

→ each leaf node must hold between $\lceil (n-1)/2 \rceil$ and $(n-1)$ value

Now, minimum number of search key values $= \frac{(3-1)}{2}$

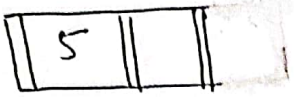
$= 1$

Maximum number of search key values $= 2$

→ If the root node is not a leaf it has at least 2 children.

Step 1

Insert key value 5



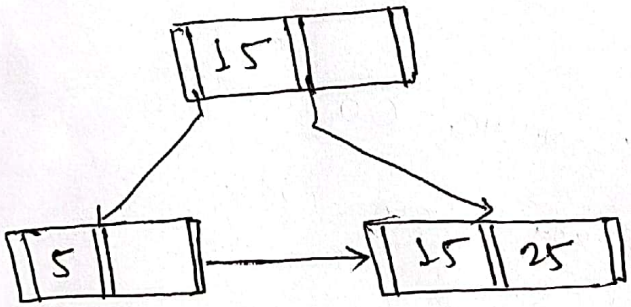
Step 2

Insert key value 15

Step 3

Insert key value 25

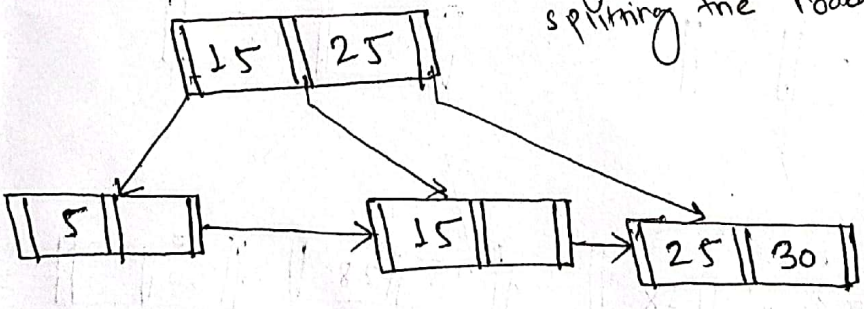
Now, node becomes full, splitting the nodes we get,



Step 4

Insert key value 30

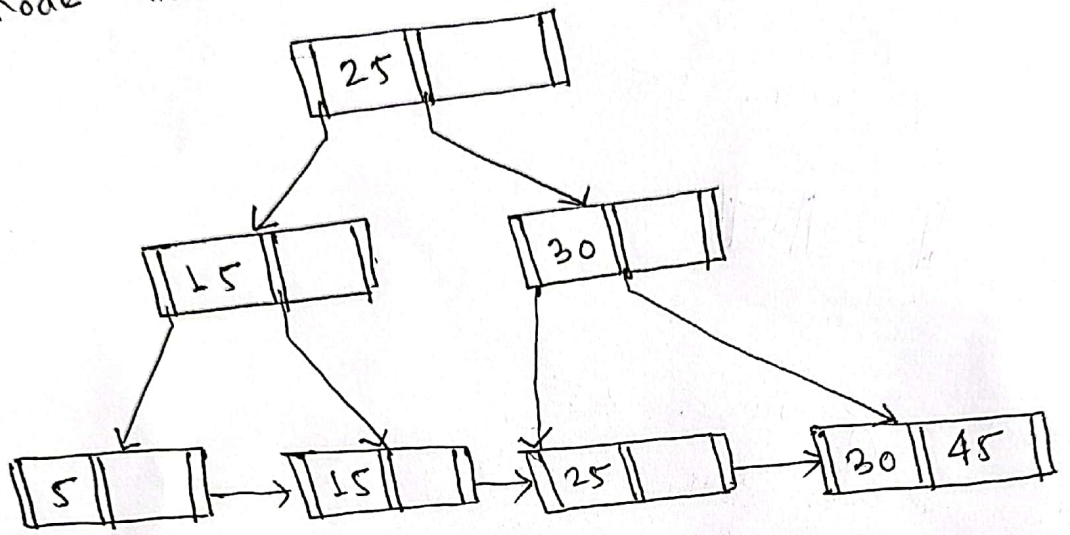
Now, rightmost node becomes full, splitting the nodes we get



Step-5

Insert key value 45

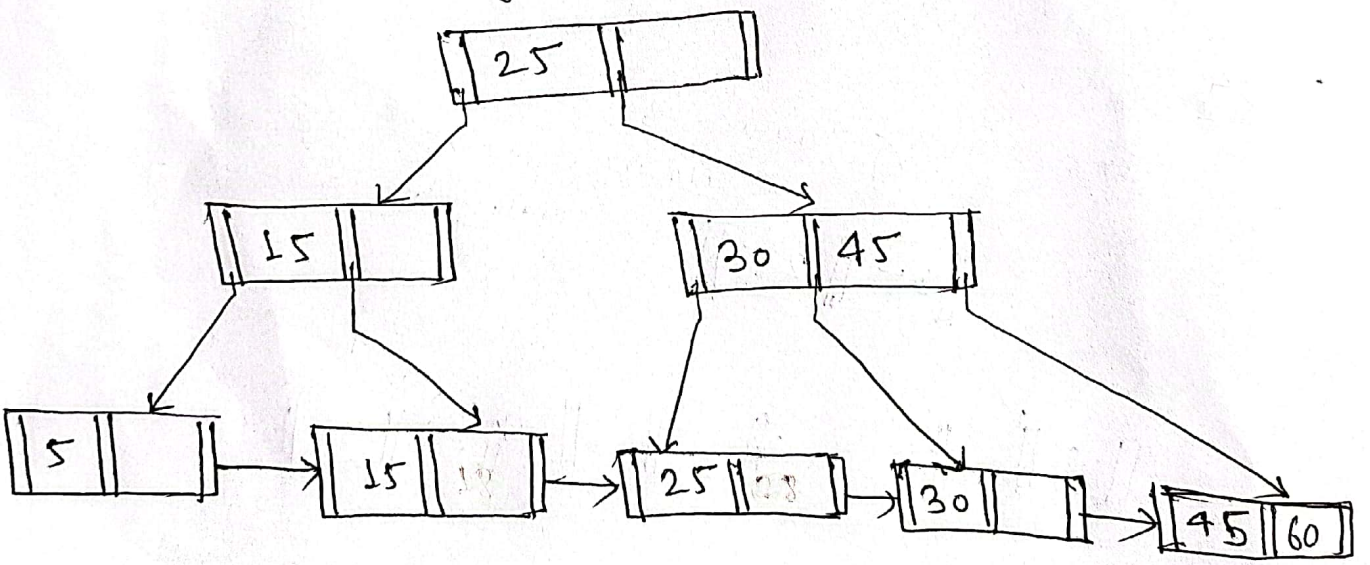
This causes rightmost leaf node full. Splitting the leaf node causes its parent node full.



Step-6

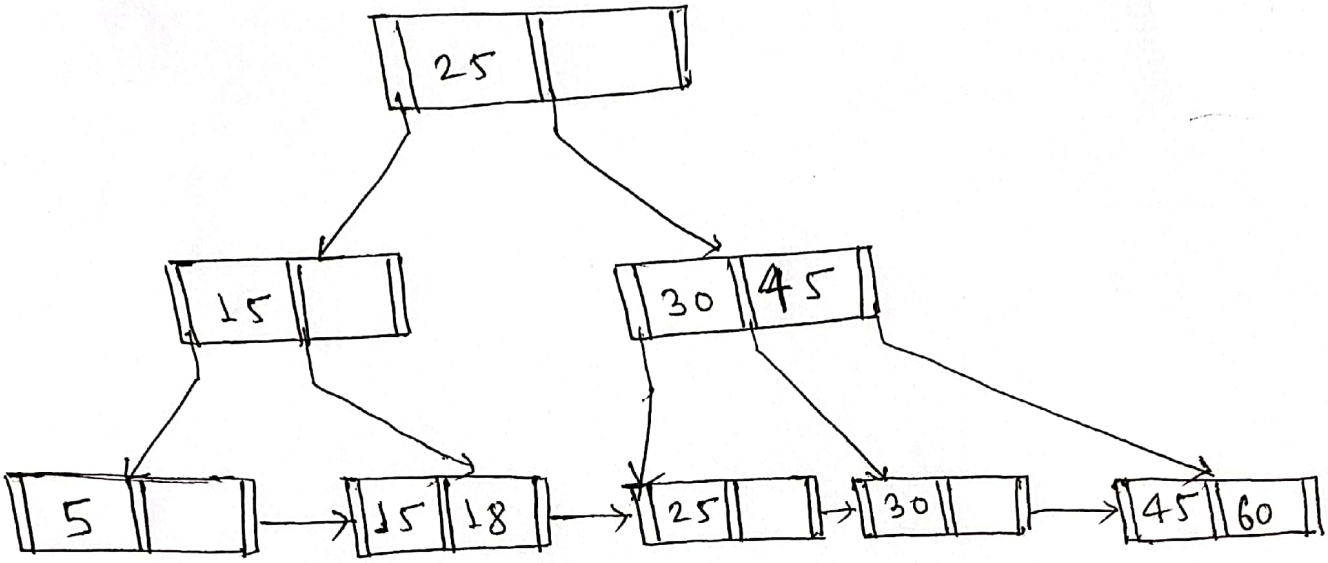
Insert key value 60

This causes rightmost leaf node full, splitting the nodes we get



Step-7

Insert key value 18



Step-8

Insert key value 28