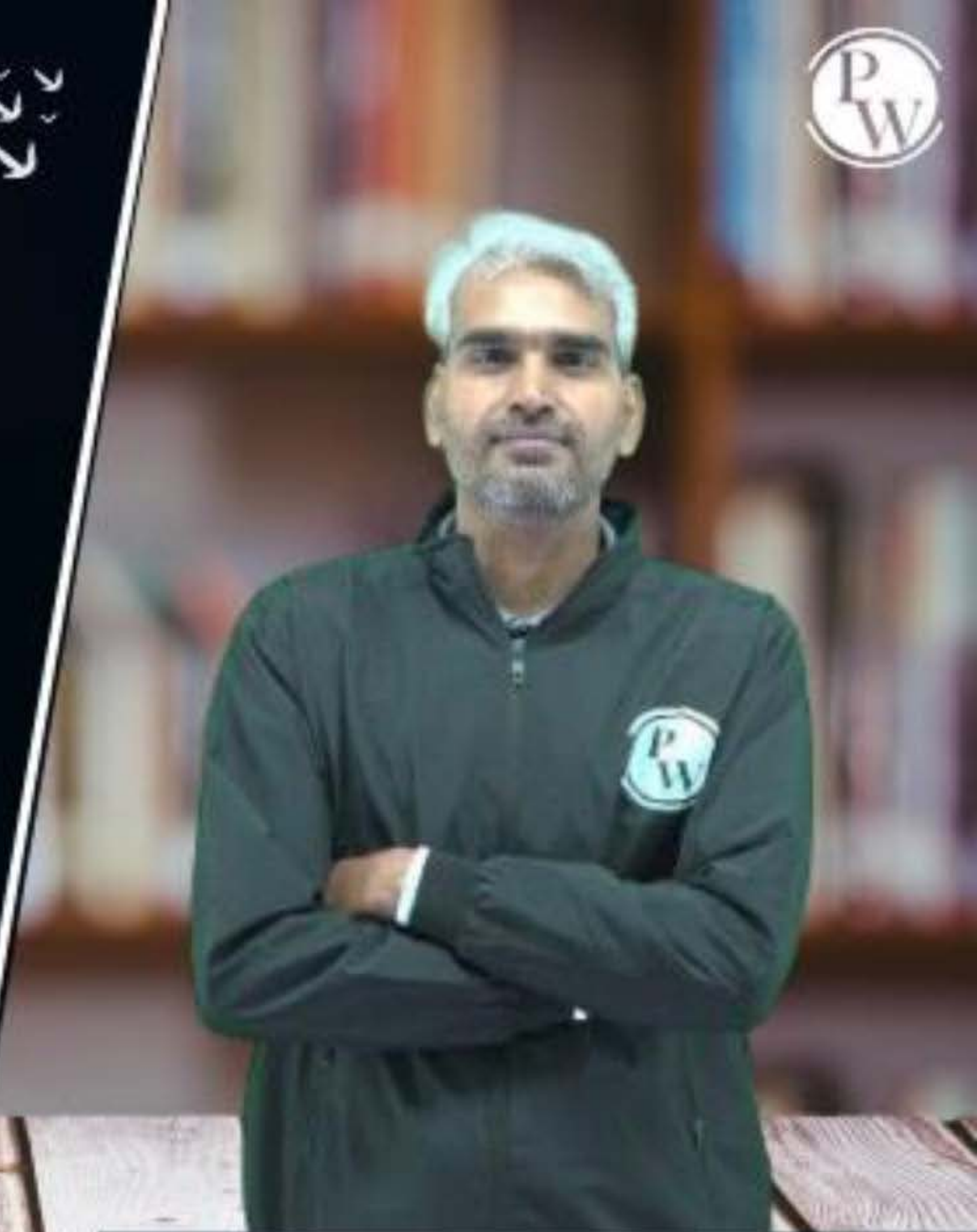
Computer Science & IT

Database Management System

File organization and indexing

Lecture No. 07



By- Vishal Sir

Recap of Previous Lecture

Topic Insertion into B tree

Topic Deletion from B tree

Slide

Topics to be Covered

Topic — Insertion into B+ tree

Topic — Deletion from B+ tree

Slide

Analysis w.r.t. B tree of order = P

Height/ level	Minimum number of nodes at	minimum number of Child pointer at	Minimum number of Keys/RP at	Maximum number of nodes at	Maximum number of Child pointer at	Maximum number of Keys/RP at
(root) $h = 0$ / $1 = l$	1	2	$(2-1) = 1$	1	P	$(P-1)$
$h = 1$ / $l = 2$	2	$2 * \lceil \frac{P}{2} \rceil$	$2 * (\lceil \frac{P}{2} \rceil - 1)$	P	$P * P = P^2$	$P * (P-1)$
$h = 2$ / $l = 3$	$2 * \lceil \frac{P}{2} \rceil$	$(2 * \lceil \frac{P}{2} \rceil) * \lceil \frac{P}{2} \rceil$ $= 2 * \lceil \frac{P}{2} \rceil^2$	$(2 * \lceil \frac{P}{2} \rceil)(\lceil \frac{P}{2} \rceil - 1)$	P^2	$P^2 * P = P^3$	$P^2 * (P-1)$
$h = 3$ / $l = 4$	$2 * \lceil \frac{P}{2} \rceil^2$	$2 * \lceil \frac{P}{2} \rceil^2 * \lceil \frac{P}{2} \rceil$ $= 2 * \lceil \frac{P}{2} \rceil^3$	$\underline{2 * \lceil \frac{P}{2} \rceil^2} * (\lceil \frac{P}{2} \rceil - 1)$	P^3	P^4	$P^3 (P-1)$
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Height $= h$ / $l = h+1$	$2 * \lceil \frac{P}{2} \rceil^{h-1}$	$2 * \lceil \frac{P}{2} \rceil^{h}$	$\underline{2 * \lceil \frac{P}{2} \rceil^{h-1}} * (\lceil \frac{P}{2} \rceil - 1)$	P^h	P^{h+1}	$P^h (P-1)$

Correction

Topic : Insertion in B+ tree

✓Insertion will be performed at leaf node, therefore search for appropriate leaf node

✓1. If leaf node is not full then insert the key into the leaf node in ascending order. {Nothing else needs to be done}

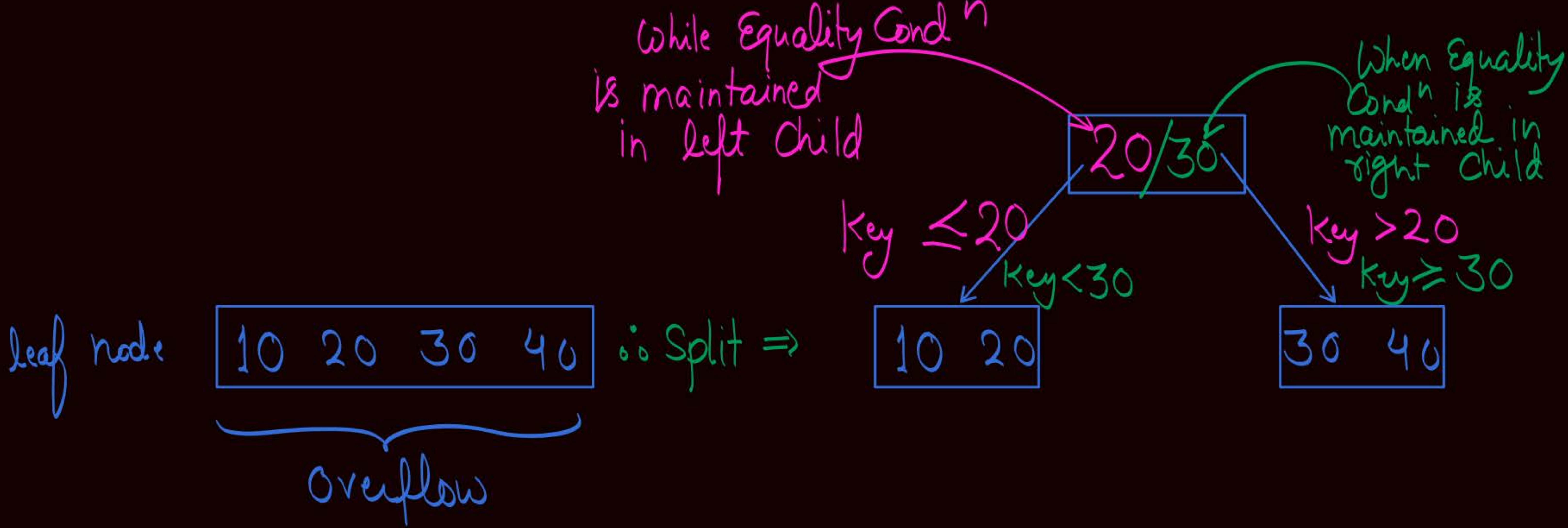
✓2. If leaf node is full,

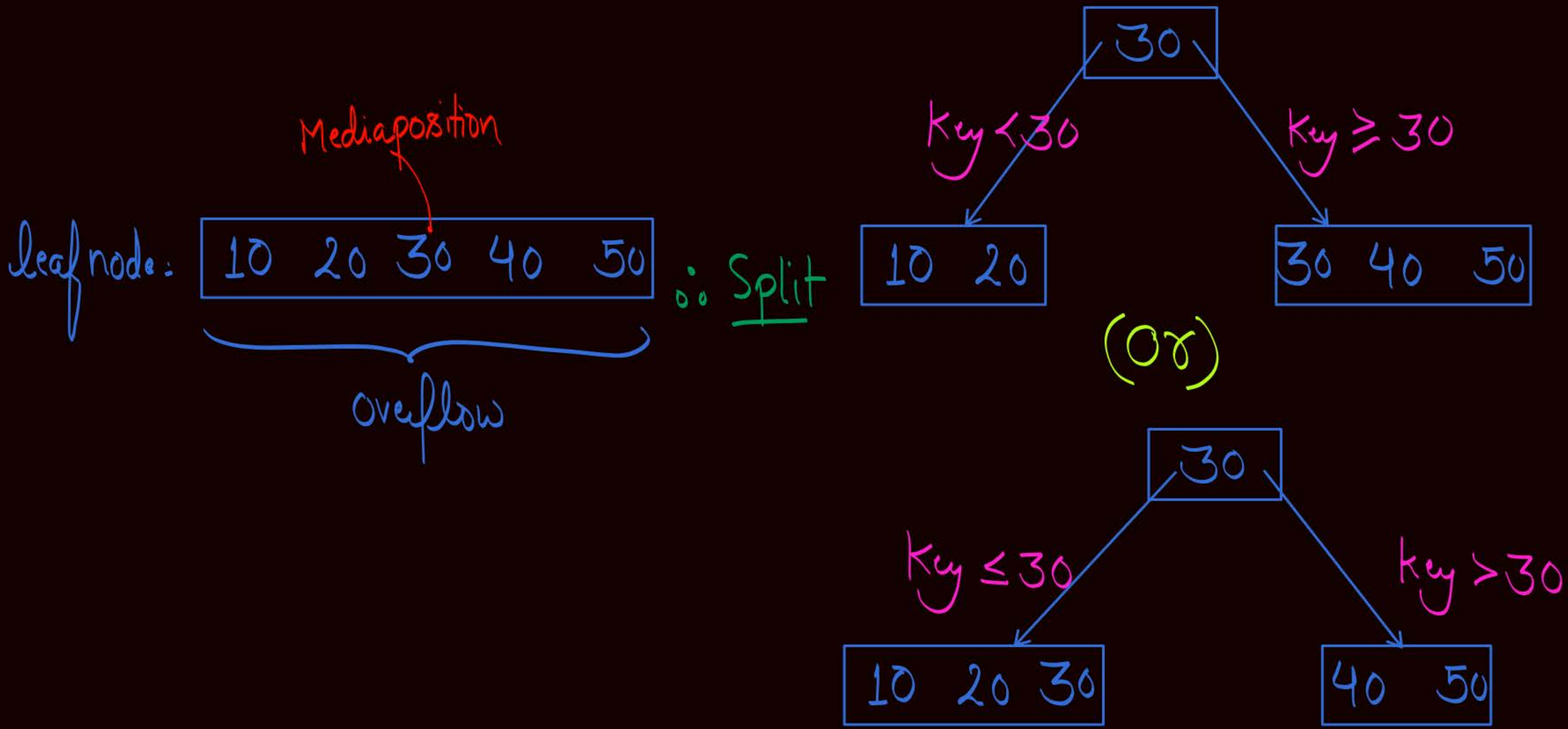
   a) Insert the key in the ascending order, and there will be overflow

   b) Split the node into two parts, and copy the median position key into the parent node, Because of this if there is overflow in the parent node then split the parent node as well.

   ← Each key must be present at leaf level

Note: While splitting the internal node the median position key will be Promoted (not Copied) to the parent node of that node


While Equality Cond^n
is maintained
in left Child
When Equality
Cond^n is
maintained in
right Child
20/30
Key ≤ 20
Key < 30
Key > 20
Key ≥ 30
leaf node
10 20 30 40
∴ Split ⇒
10 20
30 40
Overflow


Mediaposition
leaf node:
10 20 30 40 50
overflow
∴ Split
30
Key < 30
Key ≥ 30
10 20
30 40 50
(Or)
30
Key ≤ 30
key > 30
10 20 30
40 50

#Q. Let,

$\overbrace{\text{Order of internal node}}^{q}$ of B+ tree = 3, and

$\underbrace{\text{Order of leaf node}}_{m}$ of B+ tree = 2

Insert 5, 15, 25, 35, 45 into the B+ tree in the same order

- ✲ Min no. of Child pointer an internal node (non-root) must have $= \left\lceil \frac{q}{2} \right\rceil = \left\lceil \frac{3}{2} \right\rceil = 2$

  Min no. of Keys an internal node (non-root) must have $= \left\lceil \frac{q}{2} \right\rceil - 1 = \left\lceil \frac{3}{2} \right\rceil - 1 = 1$

- Min no. of keys in root = 1

  Min no. of Child pointer in root = 2

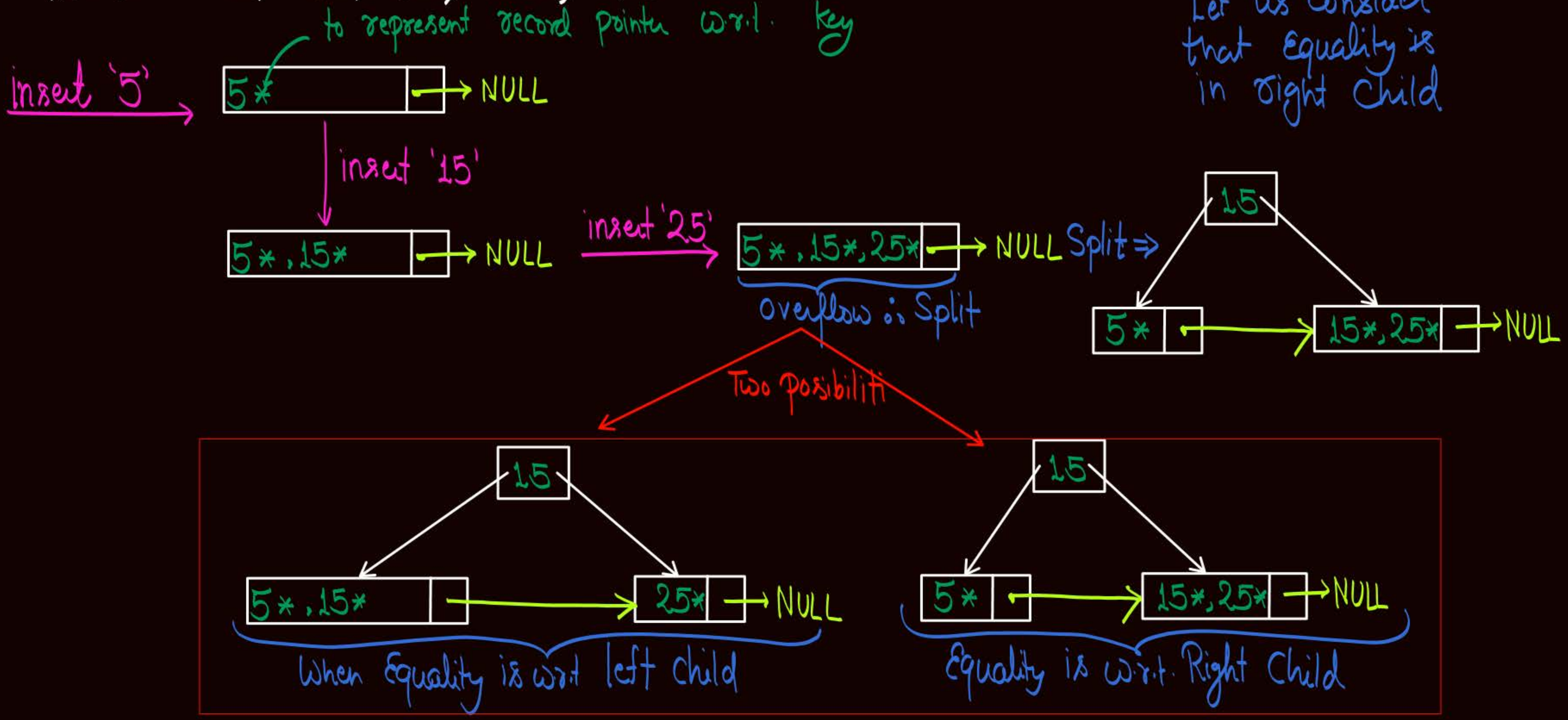
- Max no. of keys a non-leaf node can have $= (q-1) = (3-1) = 2$

  Max. no. of Child pointer a non-leaf node can have $= q = 3$

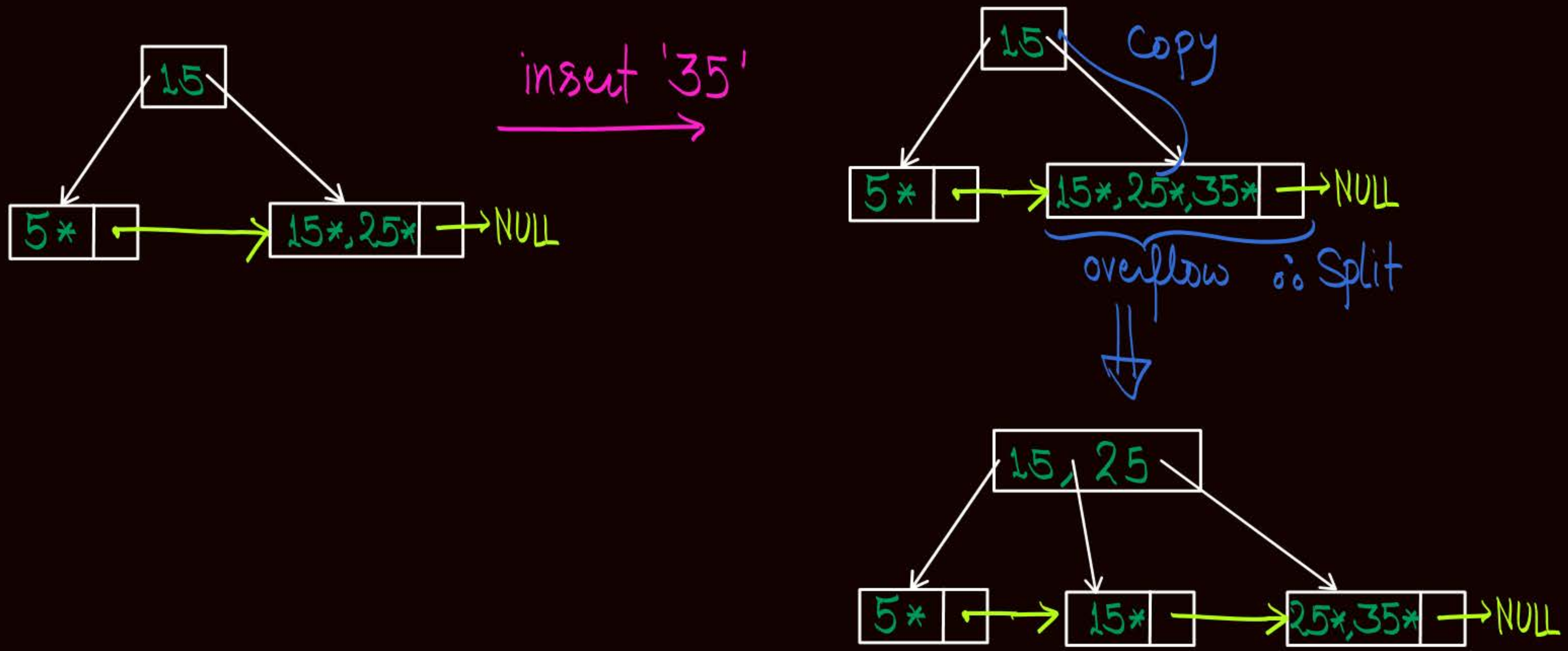
  Max no. of Keys a leaf node can have $= m = 2$

insert 5, 15, 25, 35, 45

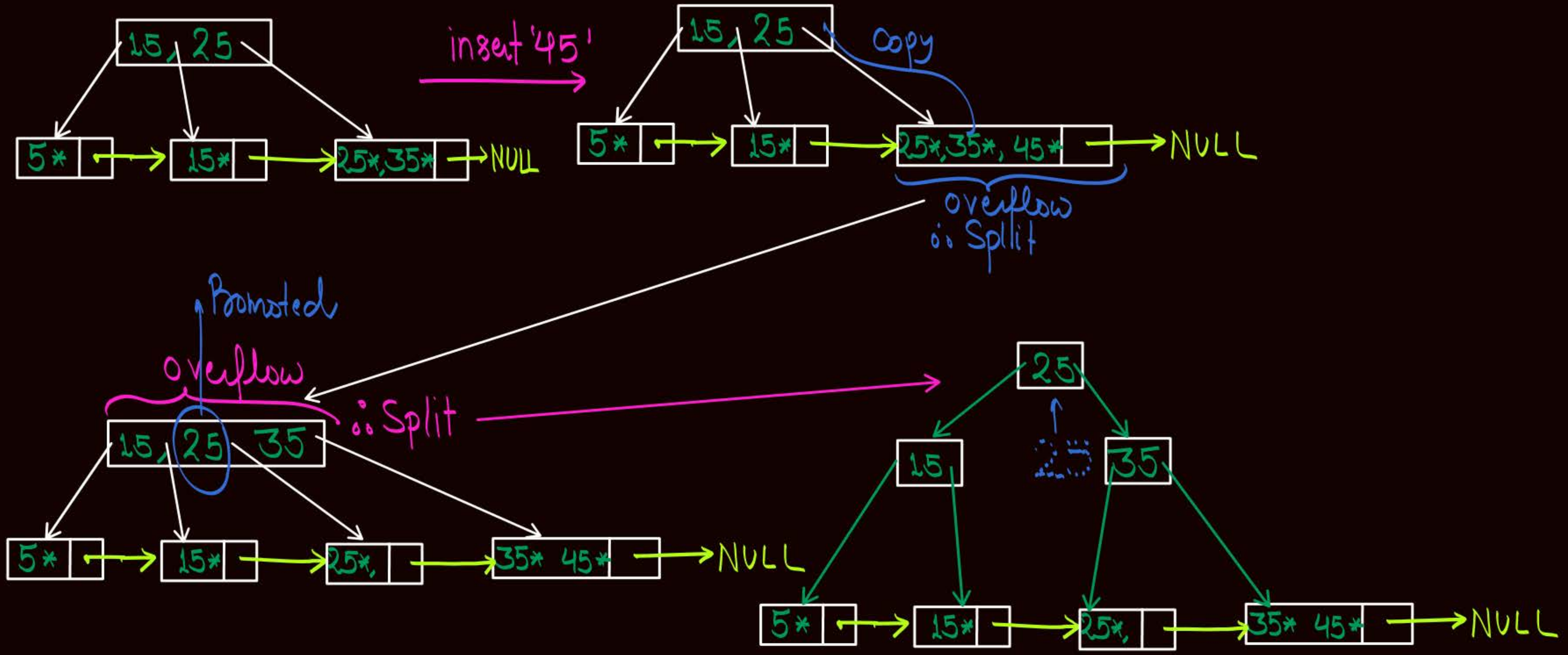
Let us consider that Equality is in right Child



insert 5, 15, 25, 35, 45

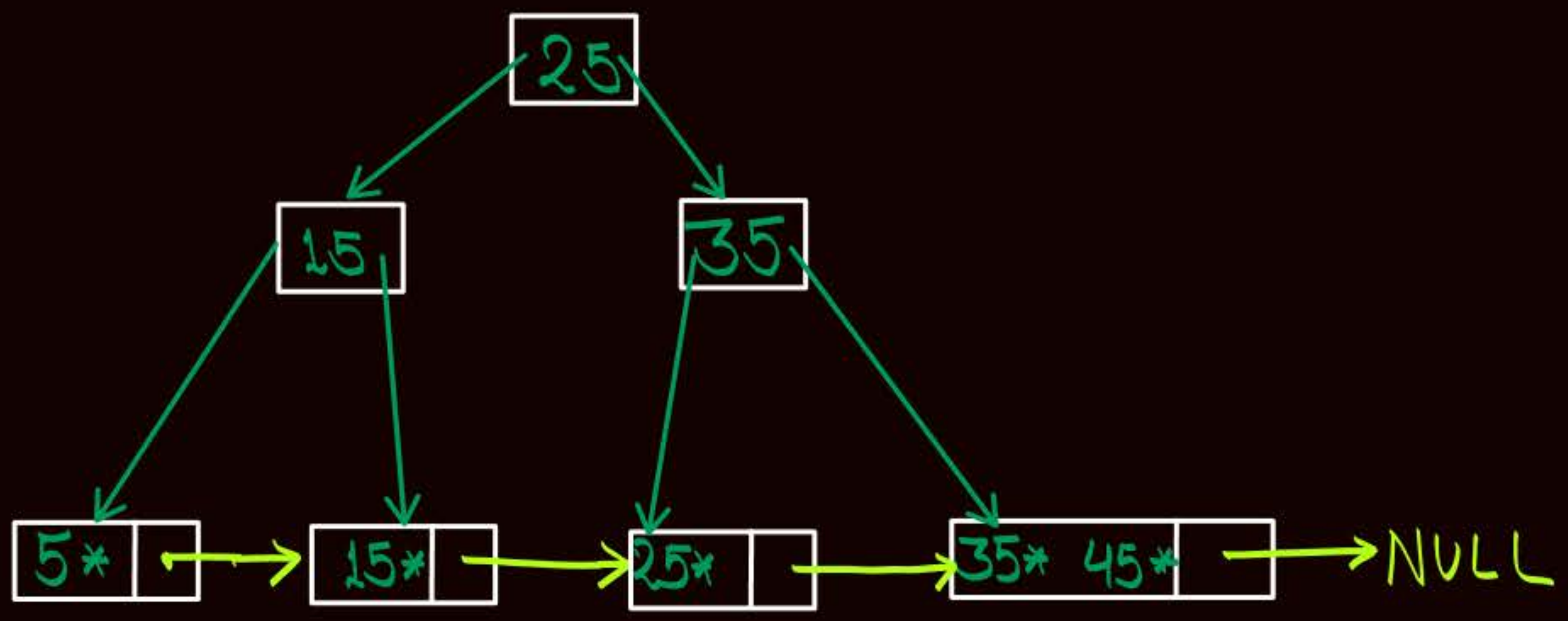


insert 5, 15, 25, 35, 45



insert 5, 15, 25, 35, 45

⇓ Final tree after all insertion



Deletion Algorithm

Topic : Deletion in B+ tree

Find the leaf node containing the key-value and delete the key from leaf node.

After deletion:-

1. If leaf node contains more than or equal to the minimum number of keys required, then no further action is required

After deletion

2. If leaf node contains less than the minimum number of keys required(i.e., underflow)

A. If any sibling node can help (left to right), then borrow (redistribute) from sibling node to rebalance the current node and update the separator key in the parent node accordingly.

B. If none of the sibling can help then merge with one of its sibling. While merging separator key from the parent node is removed., If there is underflow in the Parent node then stop.

it is w.r.t. merging of leaf node

Topic : Deletion in B+ tree

It is w.r.t. non-leaf node

3. Because of merging the parent node might also underflow, requiring recursive application of the redistribution or merging up the tree until the root is reached or a stable state is achieved.

if sibling can help

if none of its sibling can help

4. If root node becomes empty because of a sequence of merging, then it is removed and its only child becomes the new root. I.e., the height of the tree reduces.

In non-leaf node redistribute via parent

While merging non-leaf node, the separator key of parent node will be retained in merged node

Q:- Consider the following B+ tree {let, Order of internal node = 3 = q & Order of leaf node = 2 = m

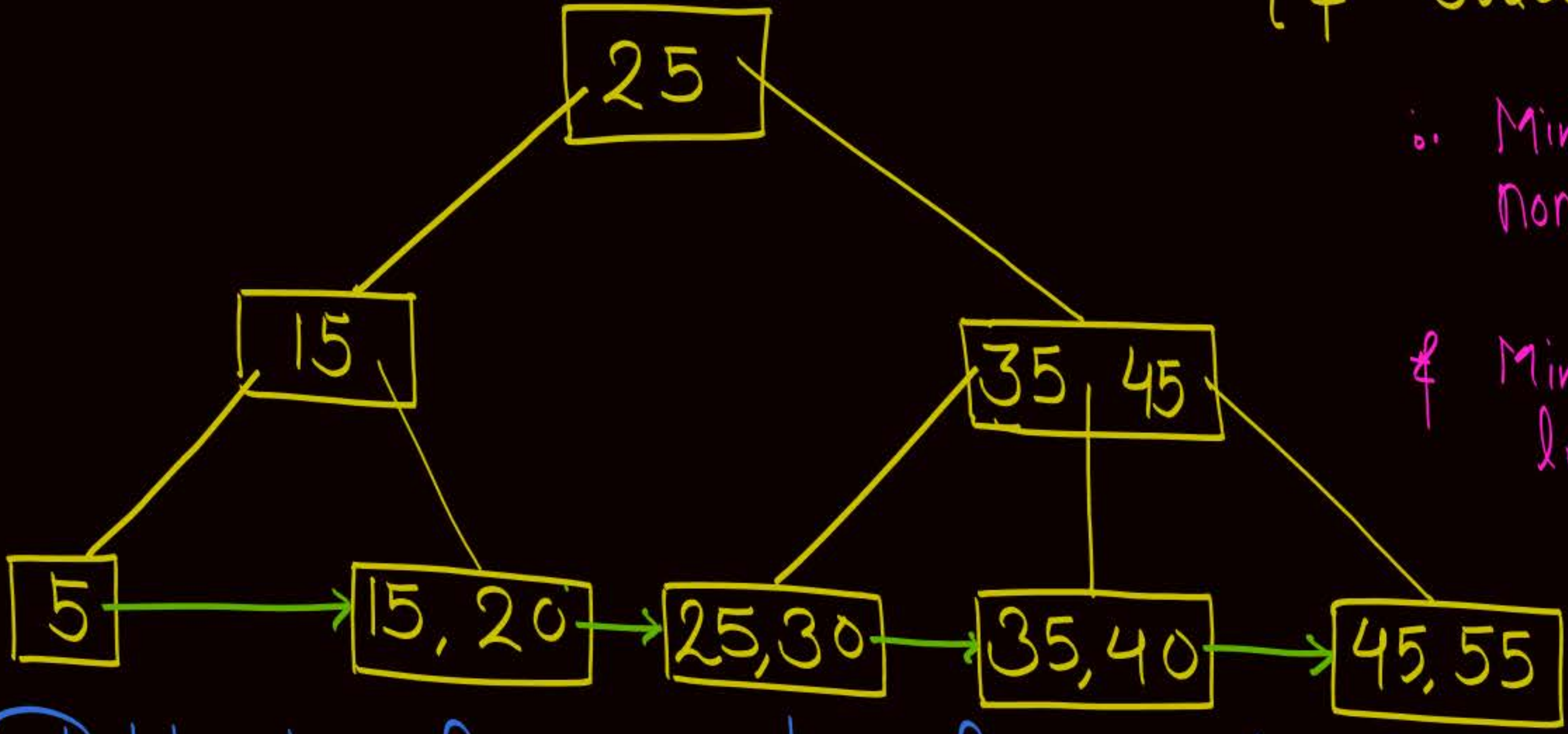
⇓

∴ Min no. of keys in a non-root & non-leaf node $= \left\lceil \frac{3}{2} \right\rceil - 1$

$= 1$

& Min no. of keys in leaf node $: \left\lceil \frac{m}{2} \right\rceil = \left\lceil \frac{2}{2} \right\rceil = 1$

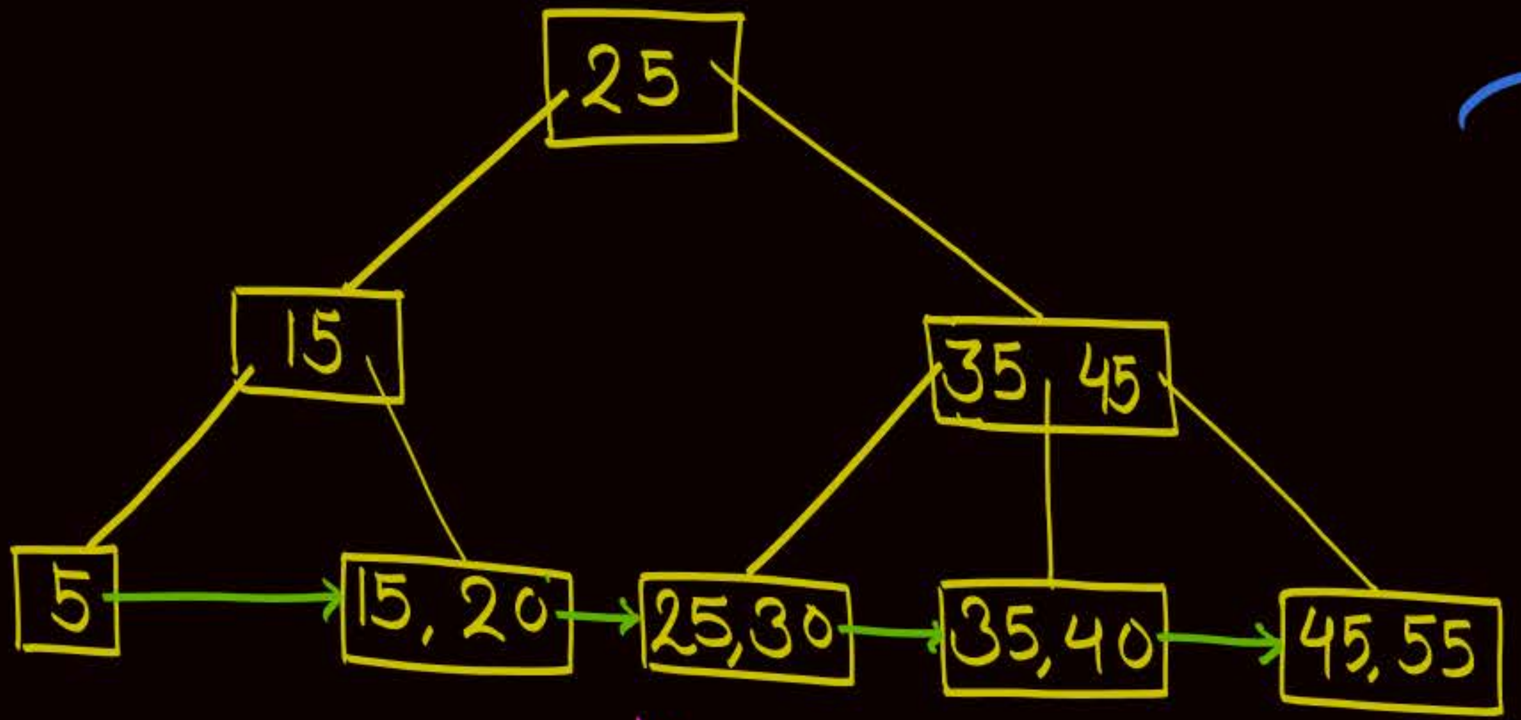
Min no. of keys a root node can have $= 1$



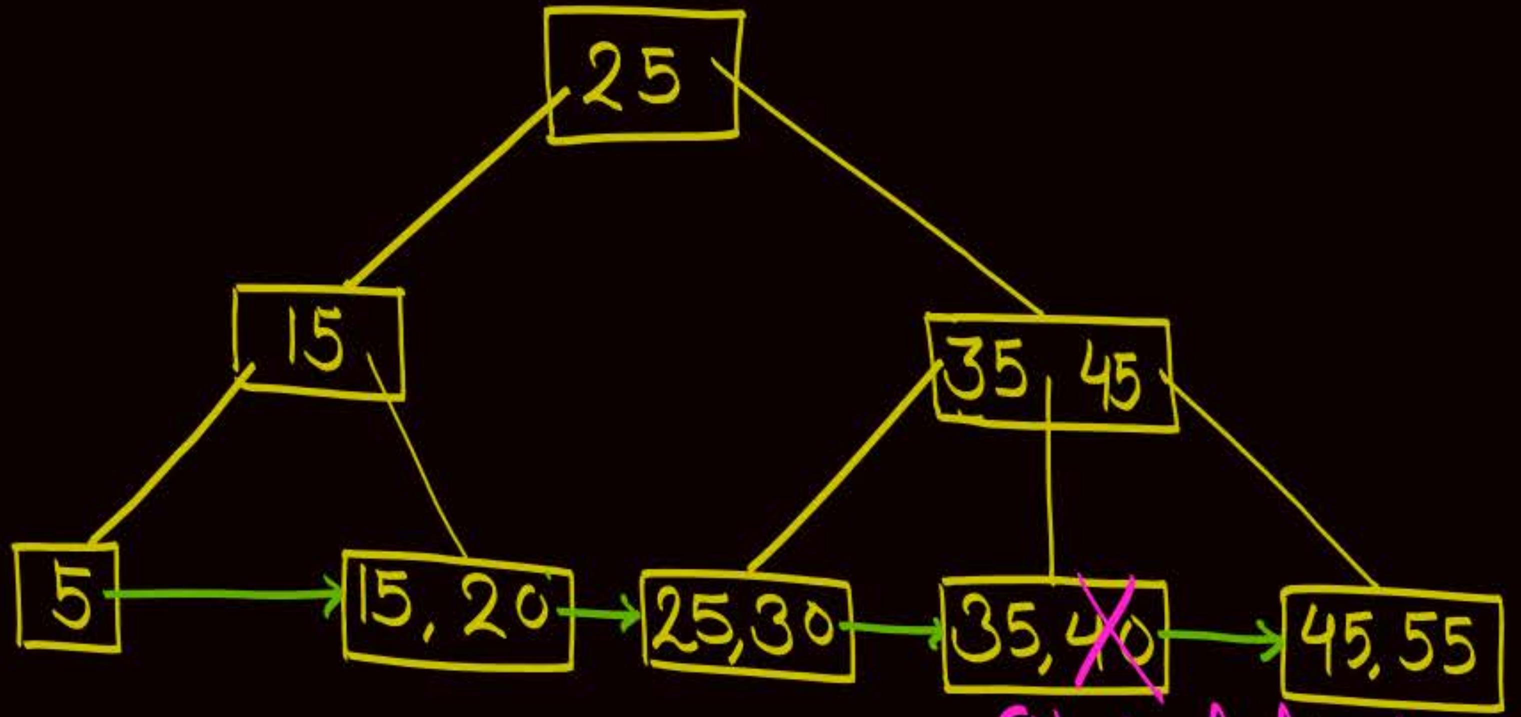
Delete the following keys from B+ tree

40, 5, 45, 35, 25, 55 in the same order

Delete: → 40, 5, 45, 35, 25, 55

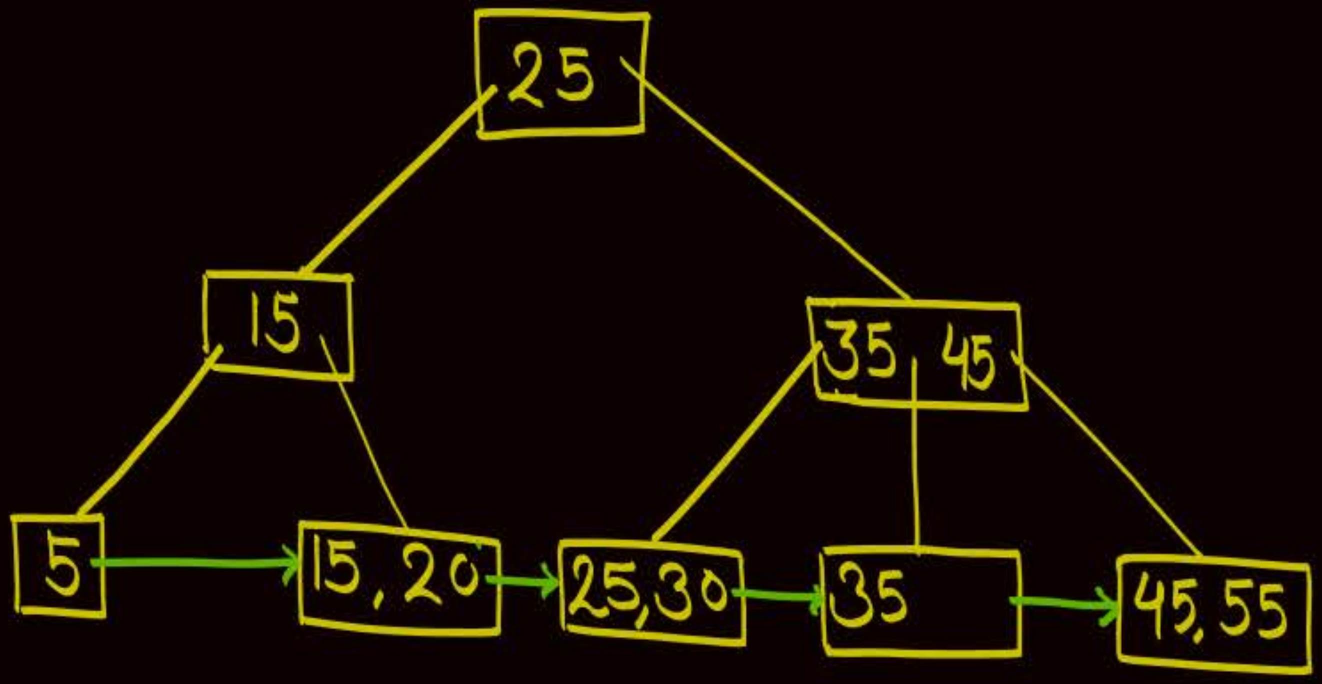


⇓ delete '40'

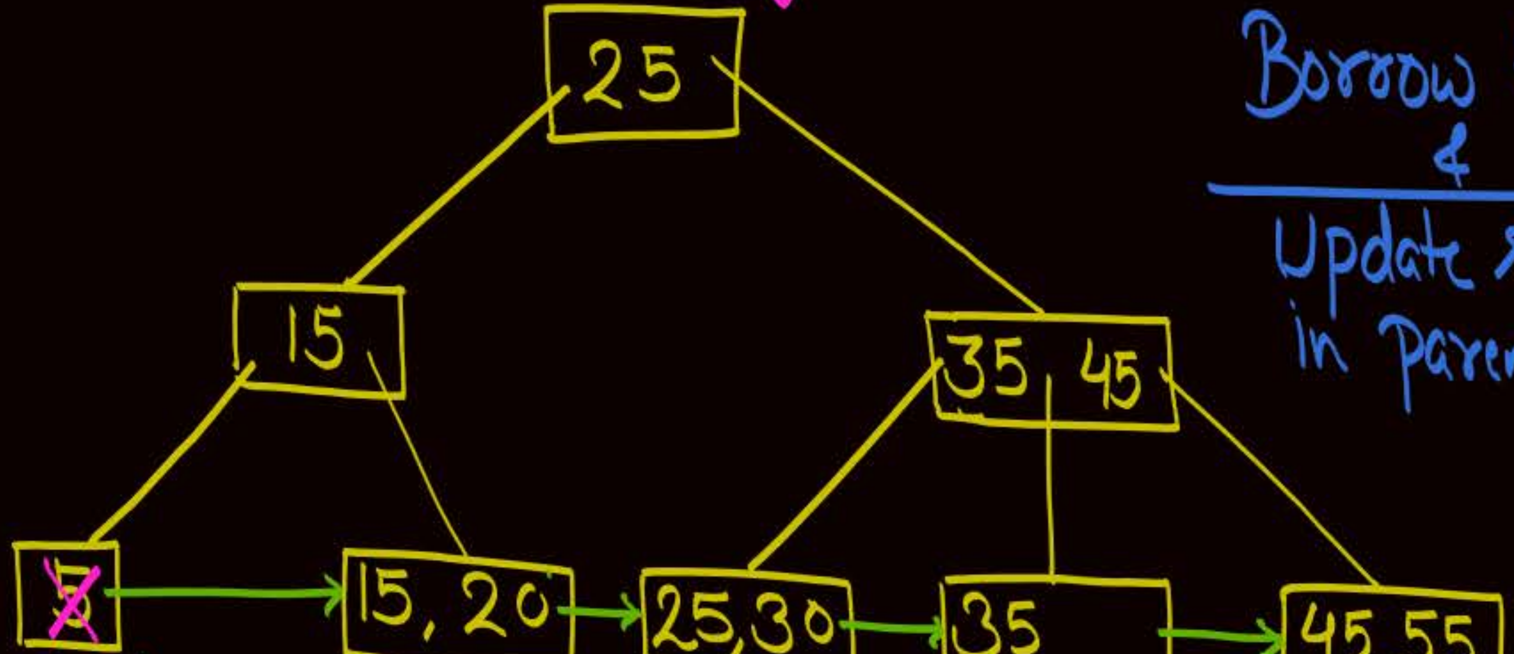


(No underflow) ∴ stop

Delete: → 40 5, 45, 35, 25, 55

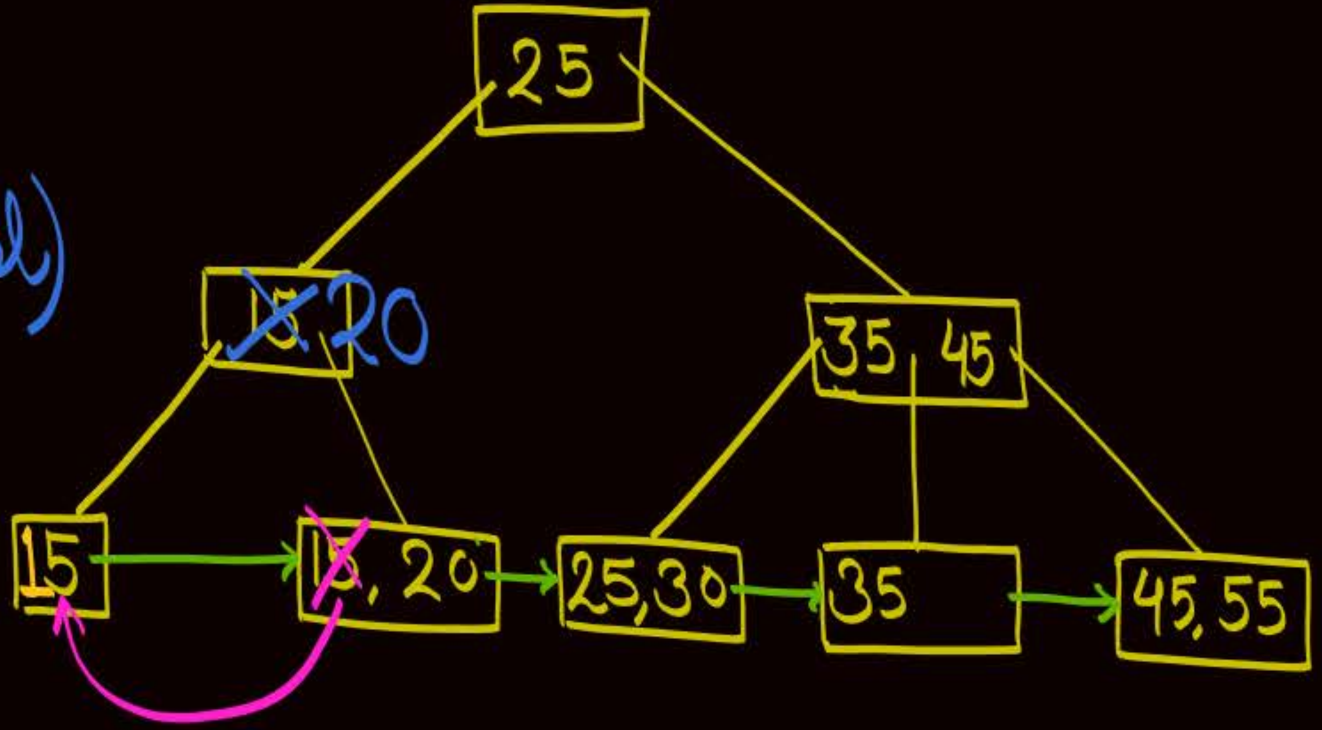


⇓ delete '5'

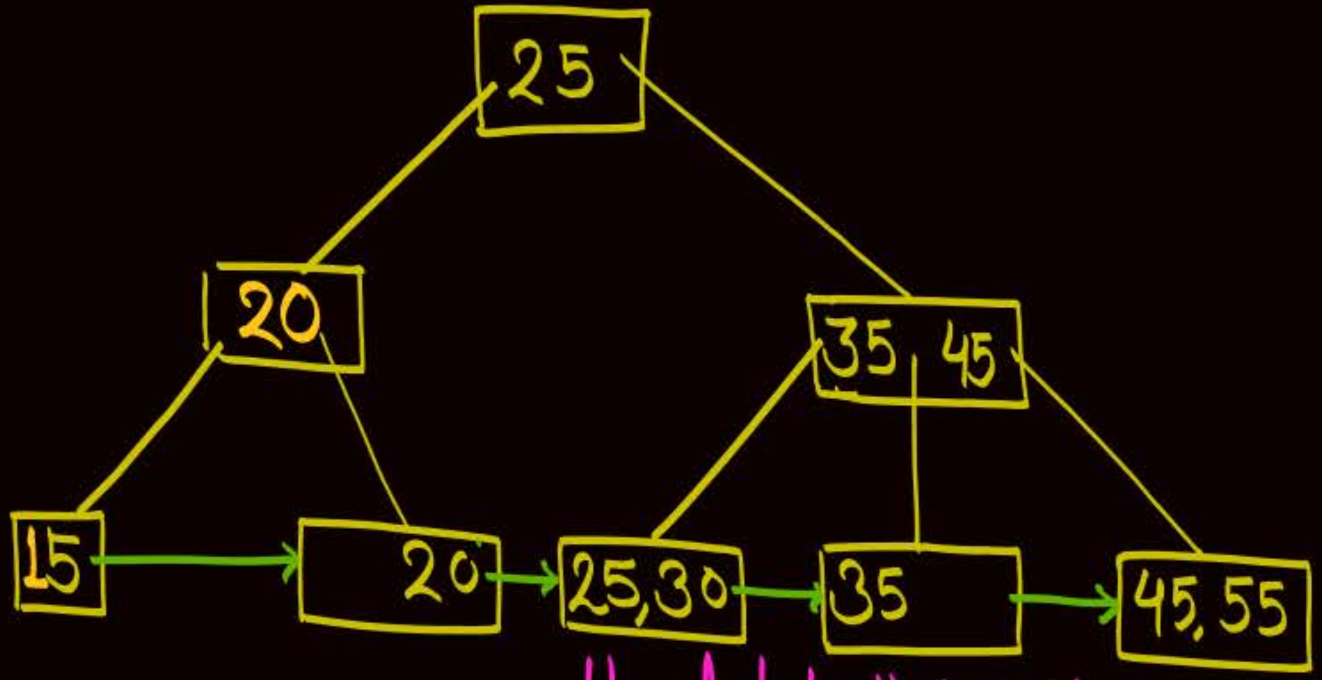


Underflow
& Right sibling can help ∴ Borrow

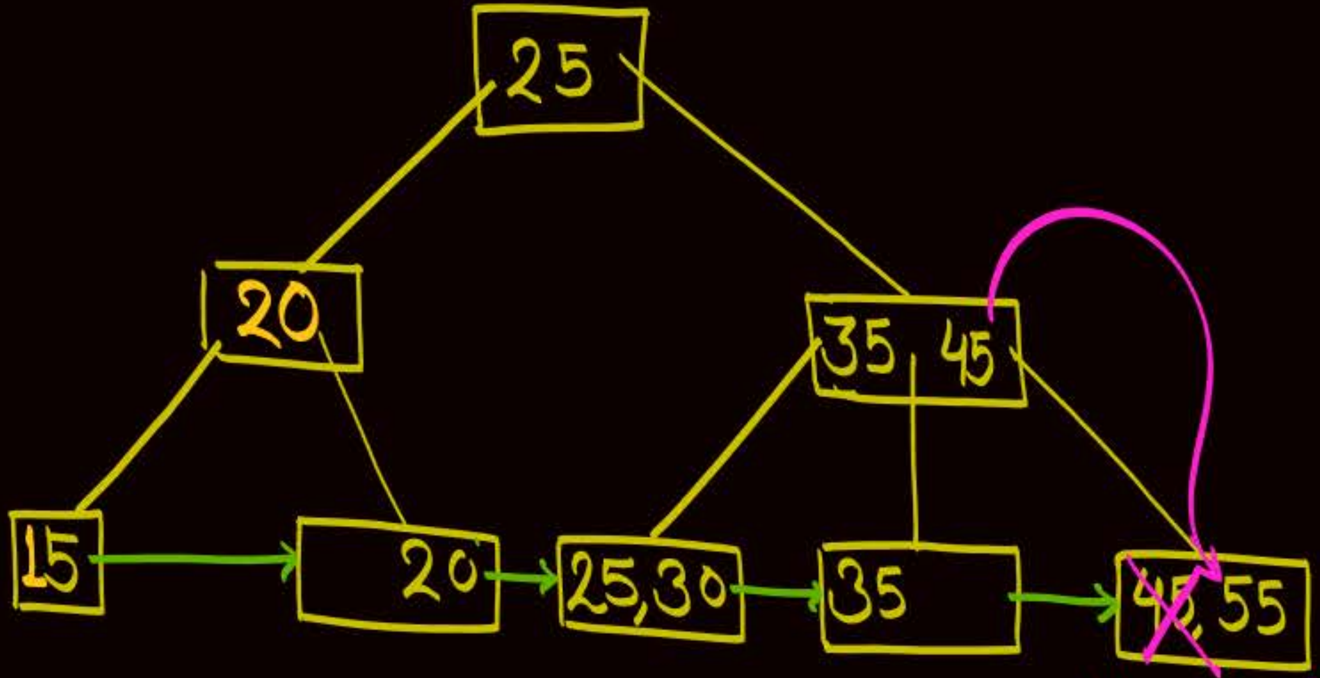
Borrow from Right
&
Update separator
in parent (if required)



Delete: → 40, 5, 45, 35, 25, 55

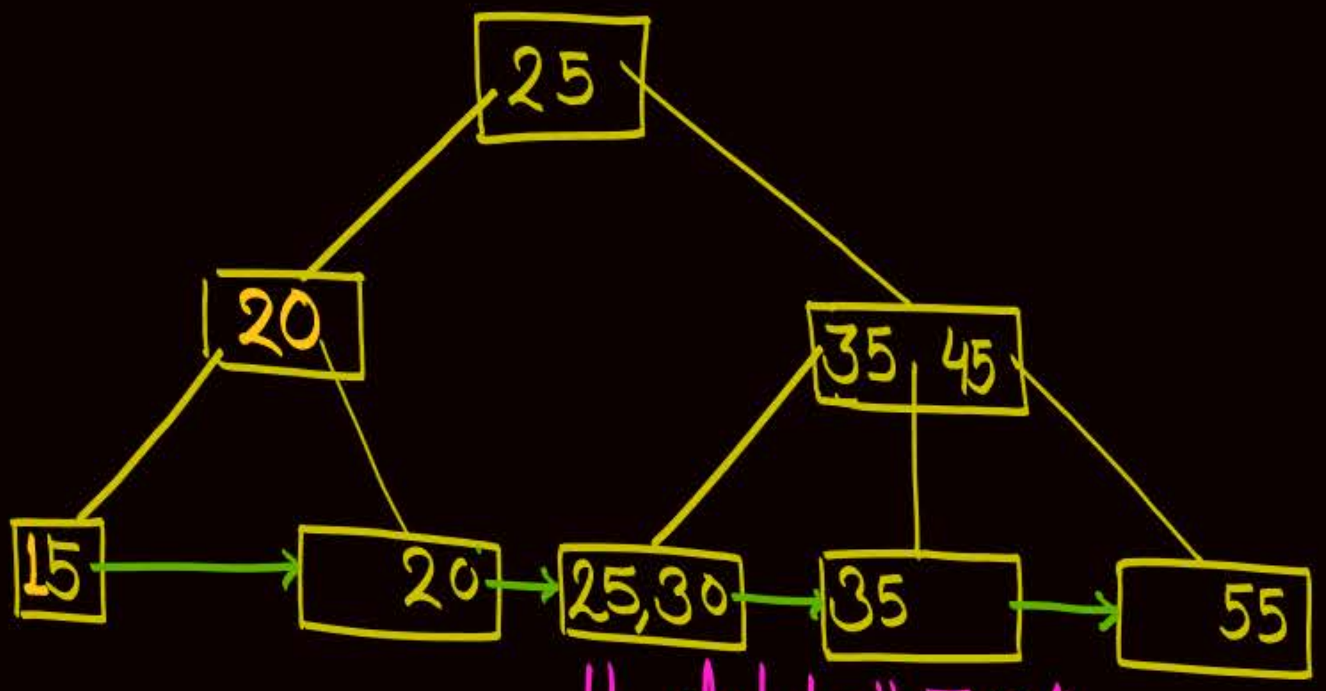


⇓ delete "45"

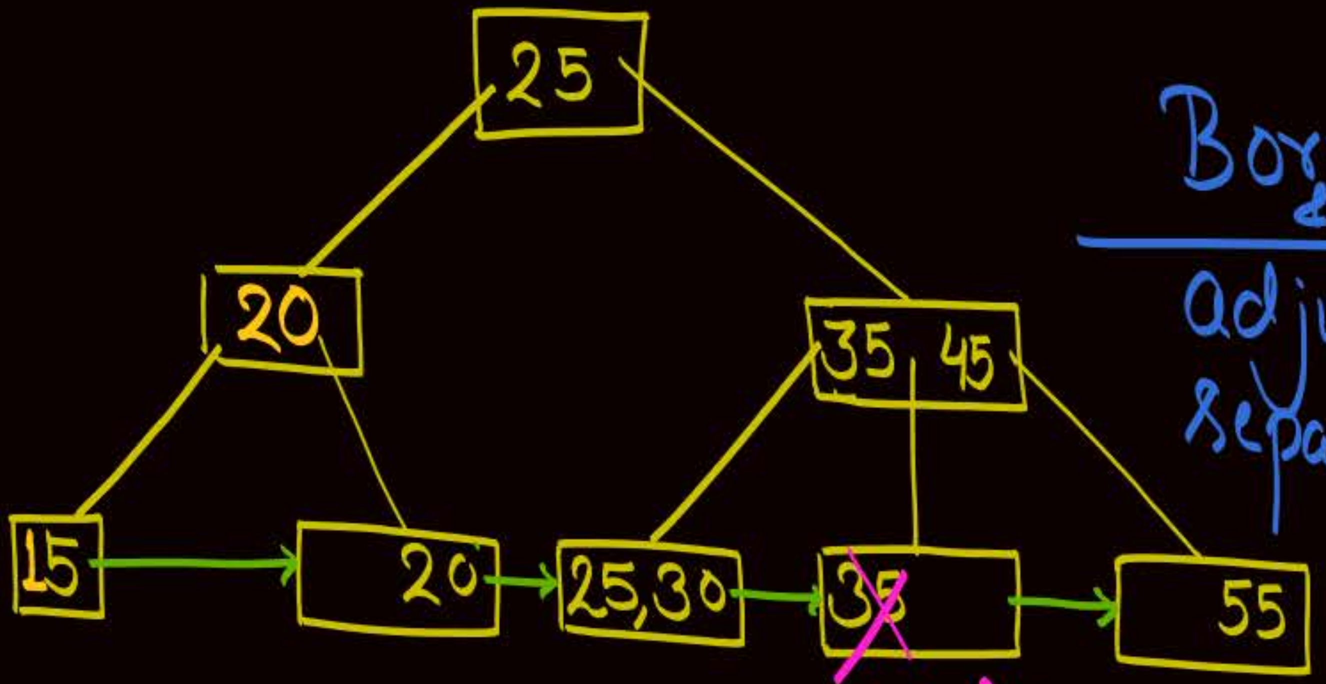


No underflow {∴ stop}

Delete: → 40, 5, 45, 35, 25, 55

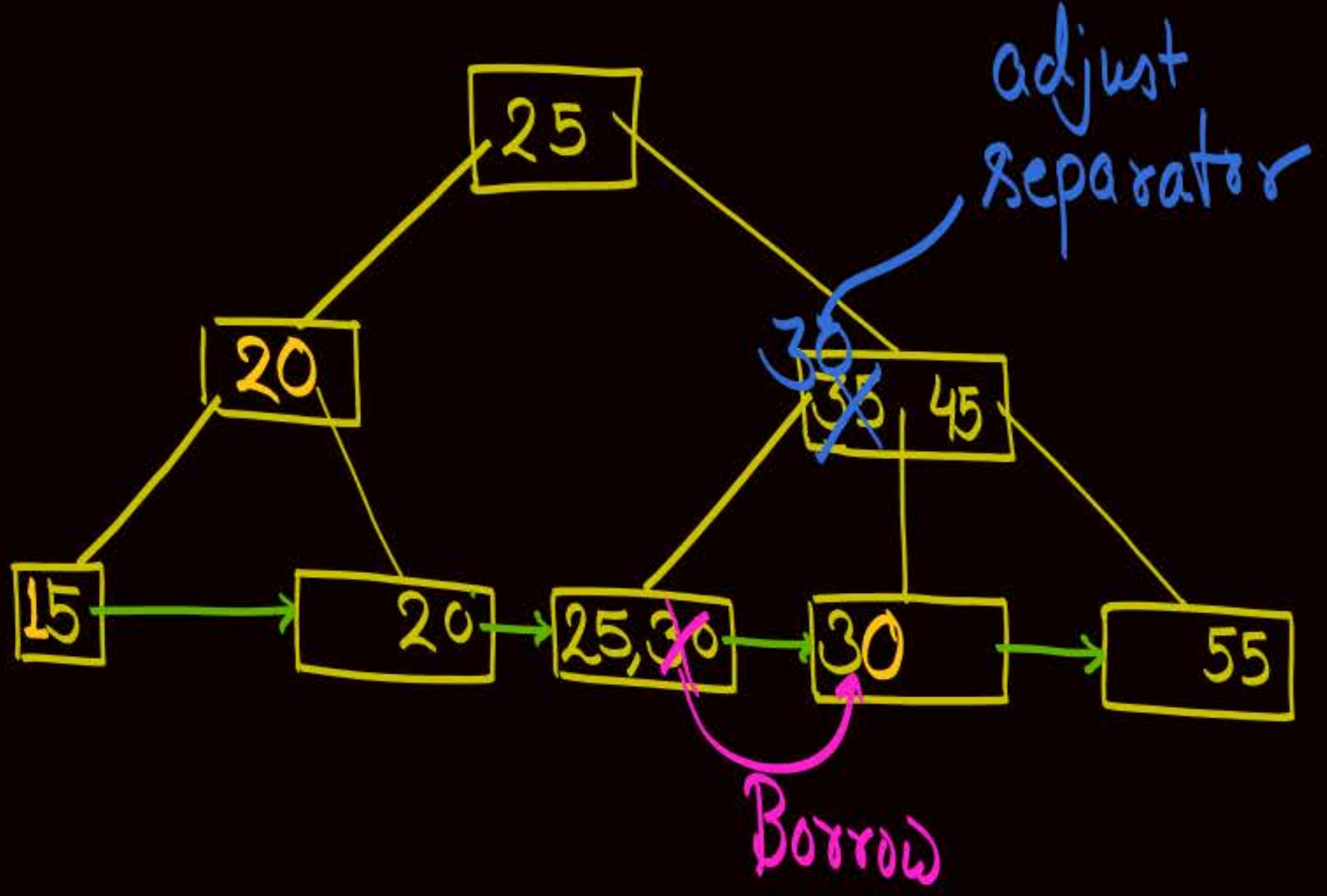


⇓ delete "35"

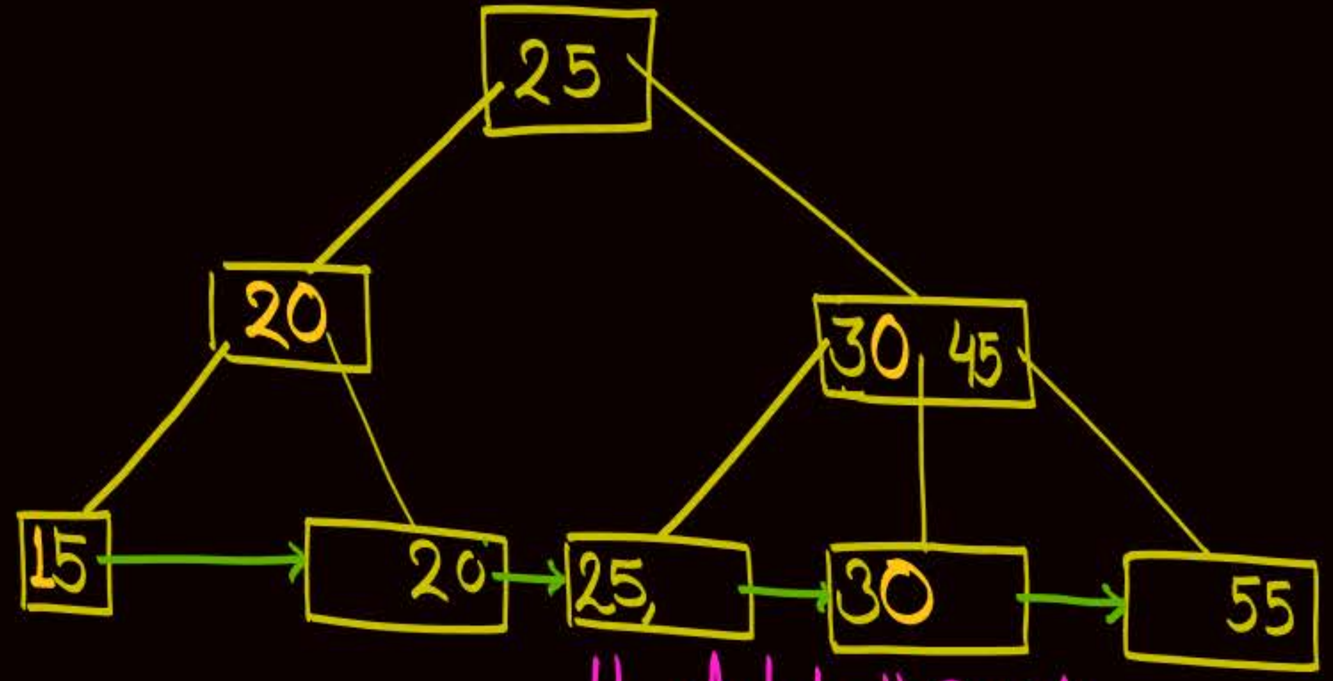


Underflow
left sibling can help

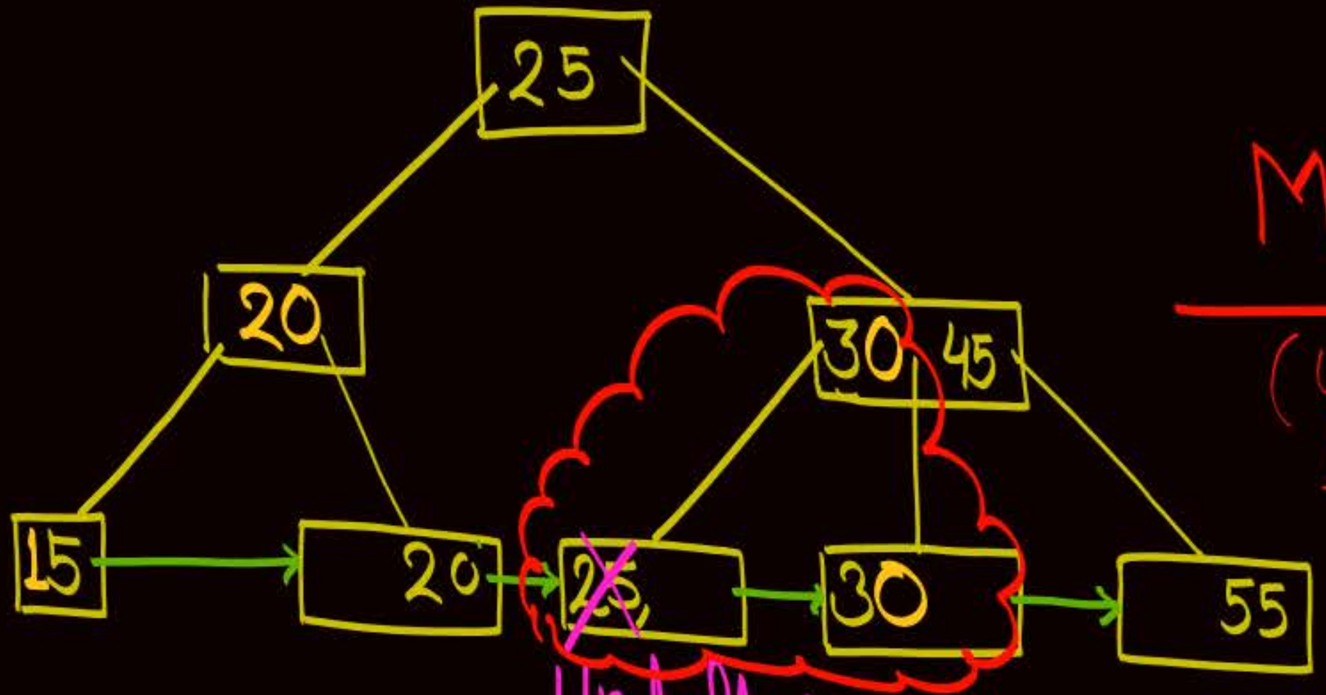
Borrow & adjust the separator key →



Delete: → 40, 5, 45, 35, 25, 55



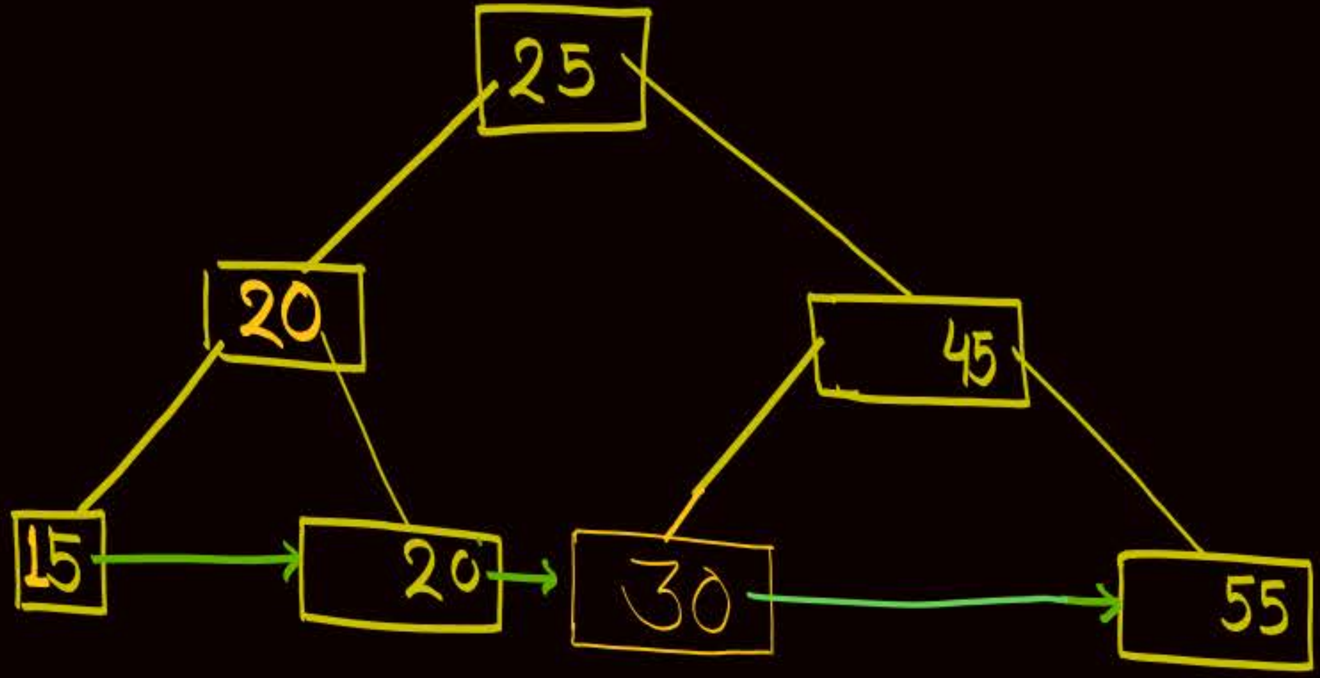
⇓ delete "25"



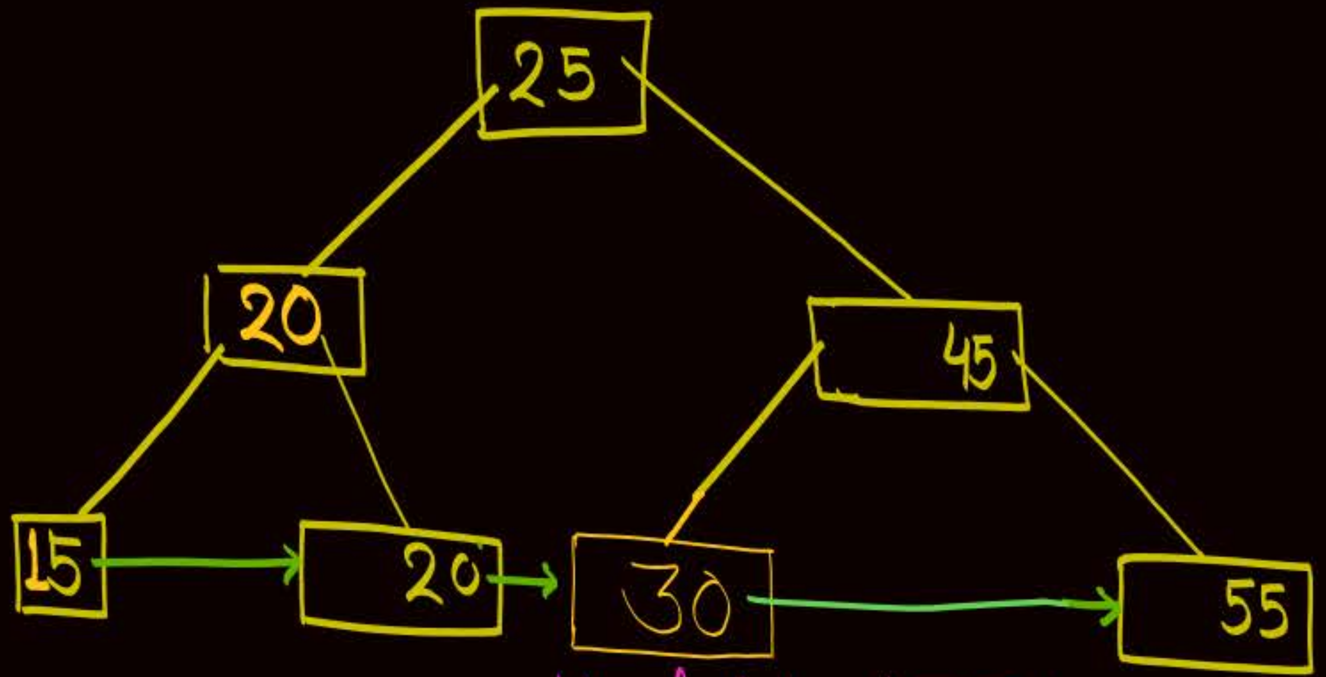
Underflow
& No sibling can help
∴ Merge

Merge →

(While merging leaf nodes key from Parent is removed)

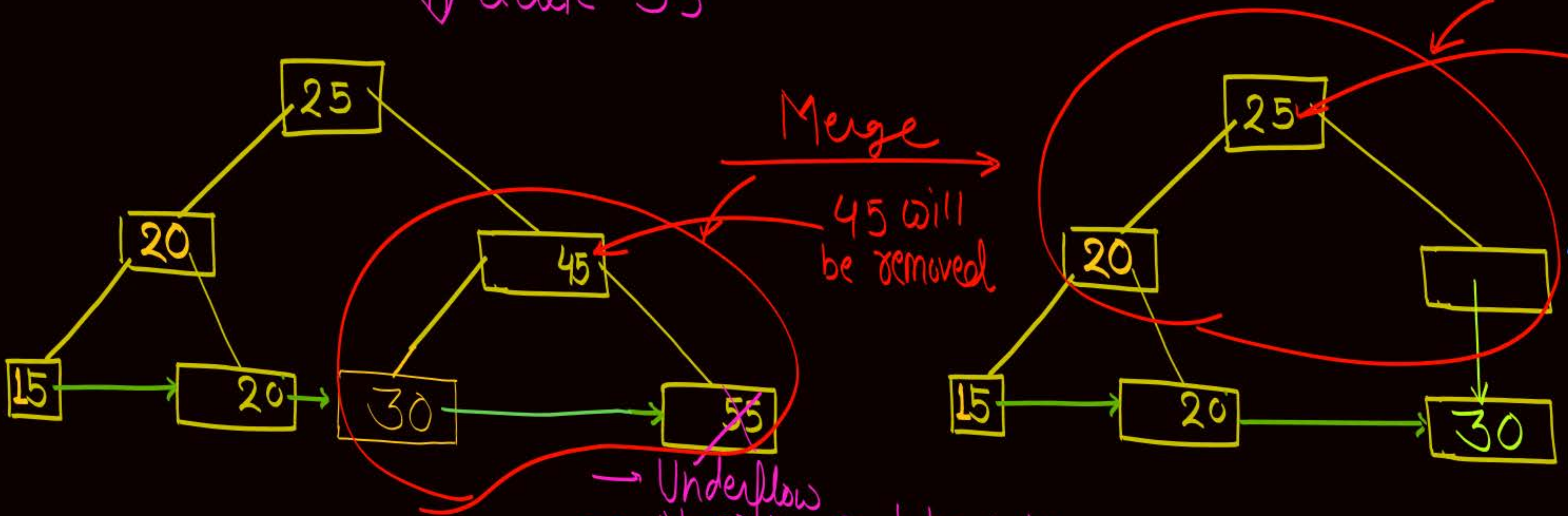


Delete: → 40, 5, 45, 35, 25, 55



⇓ delete '55'

Merge



Merge

45 will be removed

25 will be retained

→ Underflow

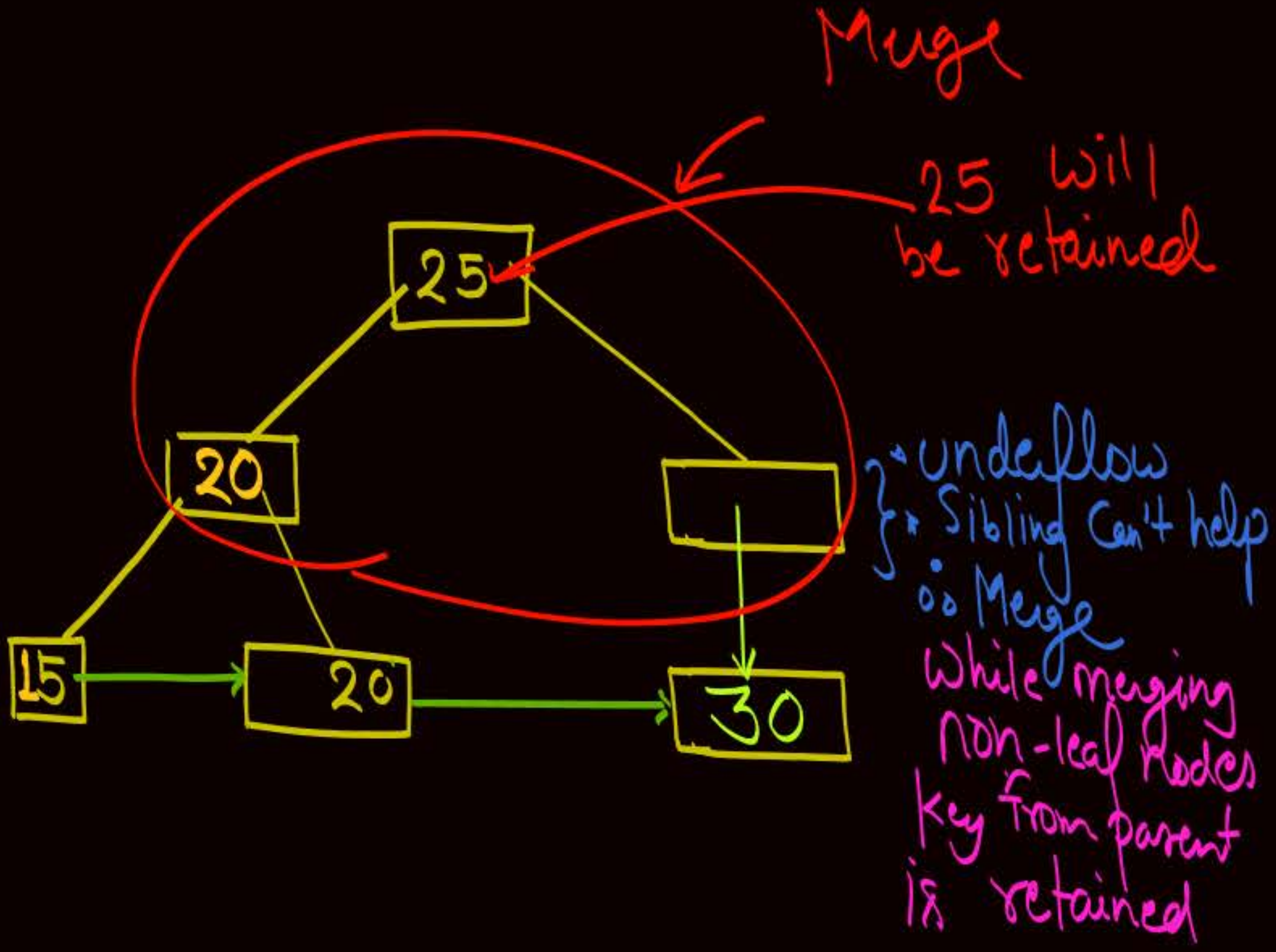
→ No sibling can help ∴ Merge

- underflow
- Sibling can't help

∴ Merge

while merging non-leaf nodes key from parent is retained

Delete: → 40, 5, 45, 35, 25, 55

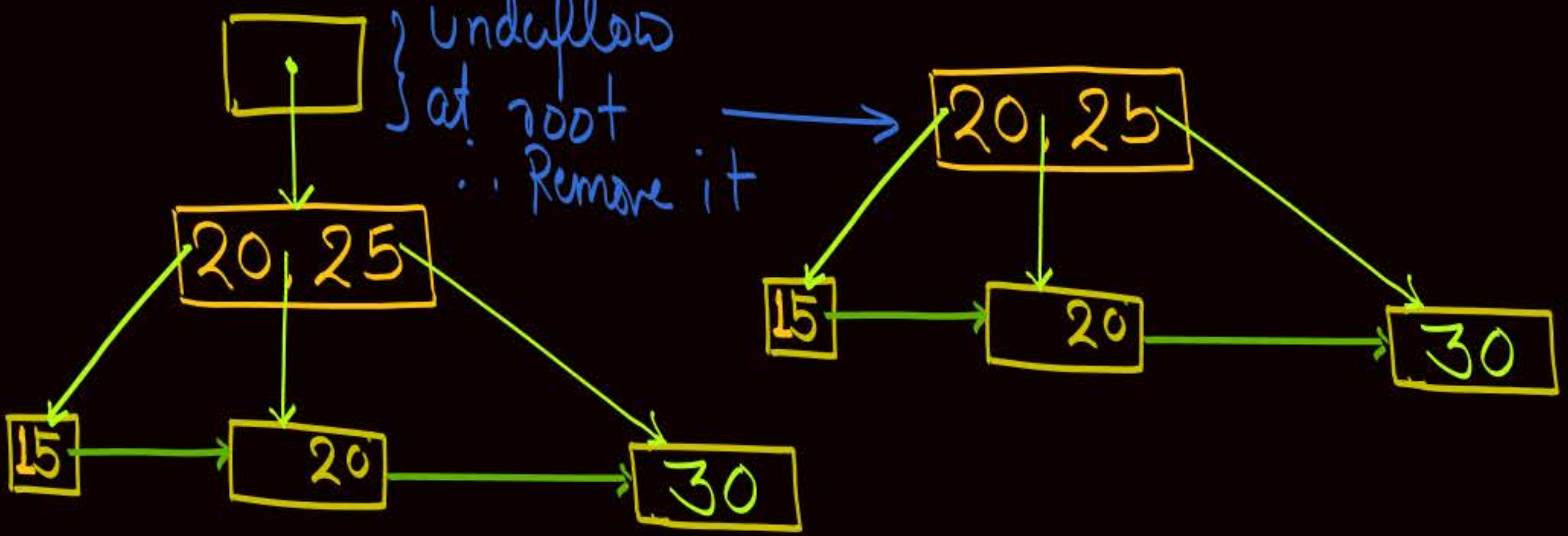


Merge

25 will be retained

} • underflow
• Sibling Can't help
∴ Merge

While merging non-leaf nodes key from parent is retained



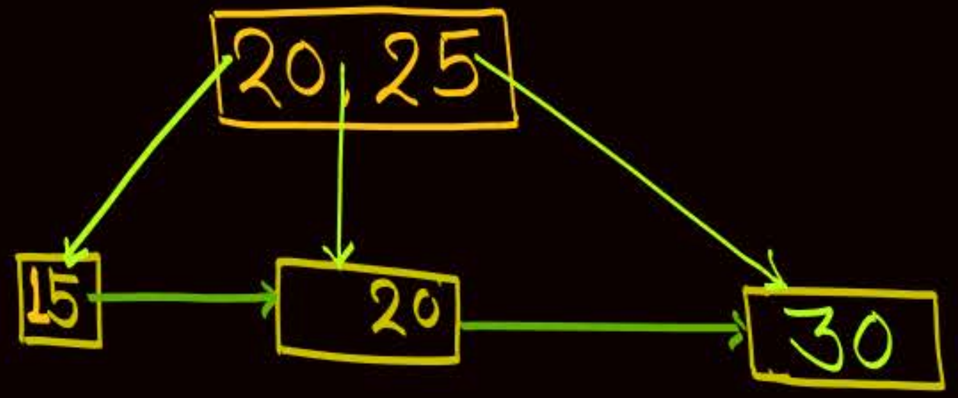
} underflow at root
∴ Remove it

Delete: → 40, 5, 45, 35, 25, 55

⇓

Final tree

⇓



Another definition for Order of a Node of B^+ tree.

Note :→

Some author define the order of a node of B+ tree as 'd' where

d = Minimum no. of Keys a non-root node must have.

In this case :

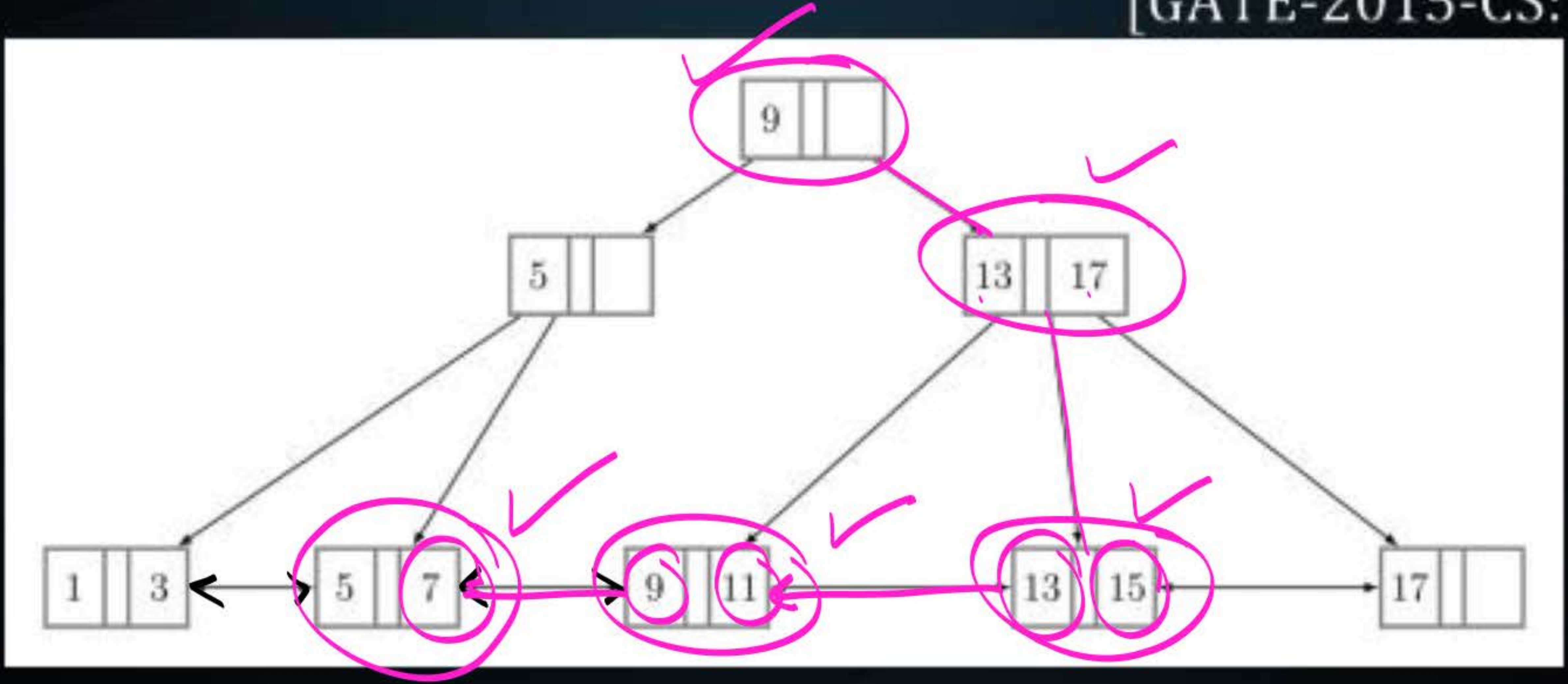
- Maximum no. of Keys a node can have = $2 \cdot d$
- Maximum no. of child pointer a node can have = "$2d + 1$"

NAT

i.e. $d = 1$, ∴ Max No. of Child ptr $= 2d+1 = 3$

- With reference to the B^+tree index of order 1 shown below, the minimum number of nodes (including the Root node) that must be fetched in order to satisfy the following query: "Get all records with a search key greater than or equal to 7 and less than 15" is 'Ans = 5"

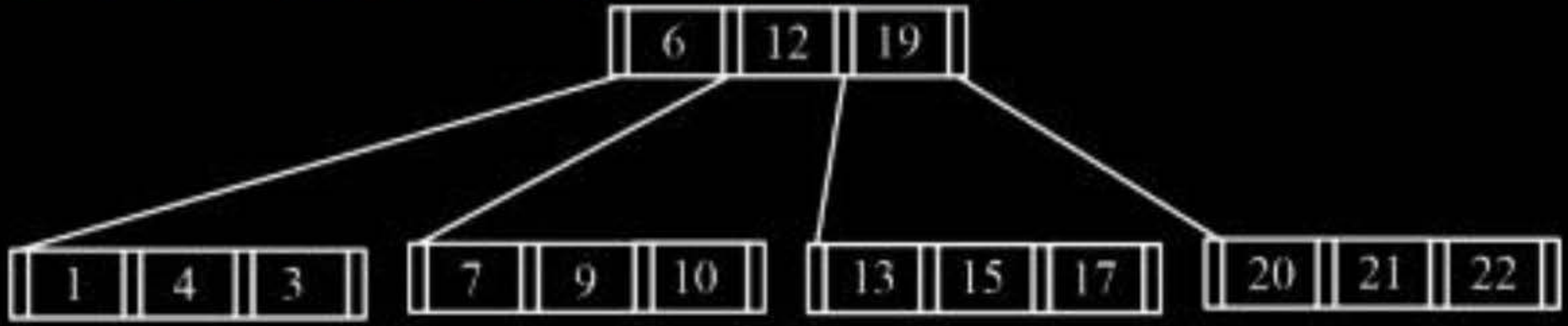
[GATE-2015-CS: 2M]



[MSQ]

[1 Mark]

#Q. Consider the following $B+$ tree with 5 nodes, in which a node can store at most 3 key values. The value 23 is now inserted in the $B+$ tree. Which of the following options(s) is/are CORRECT?

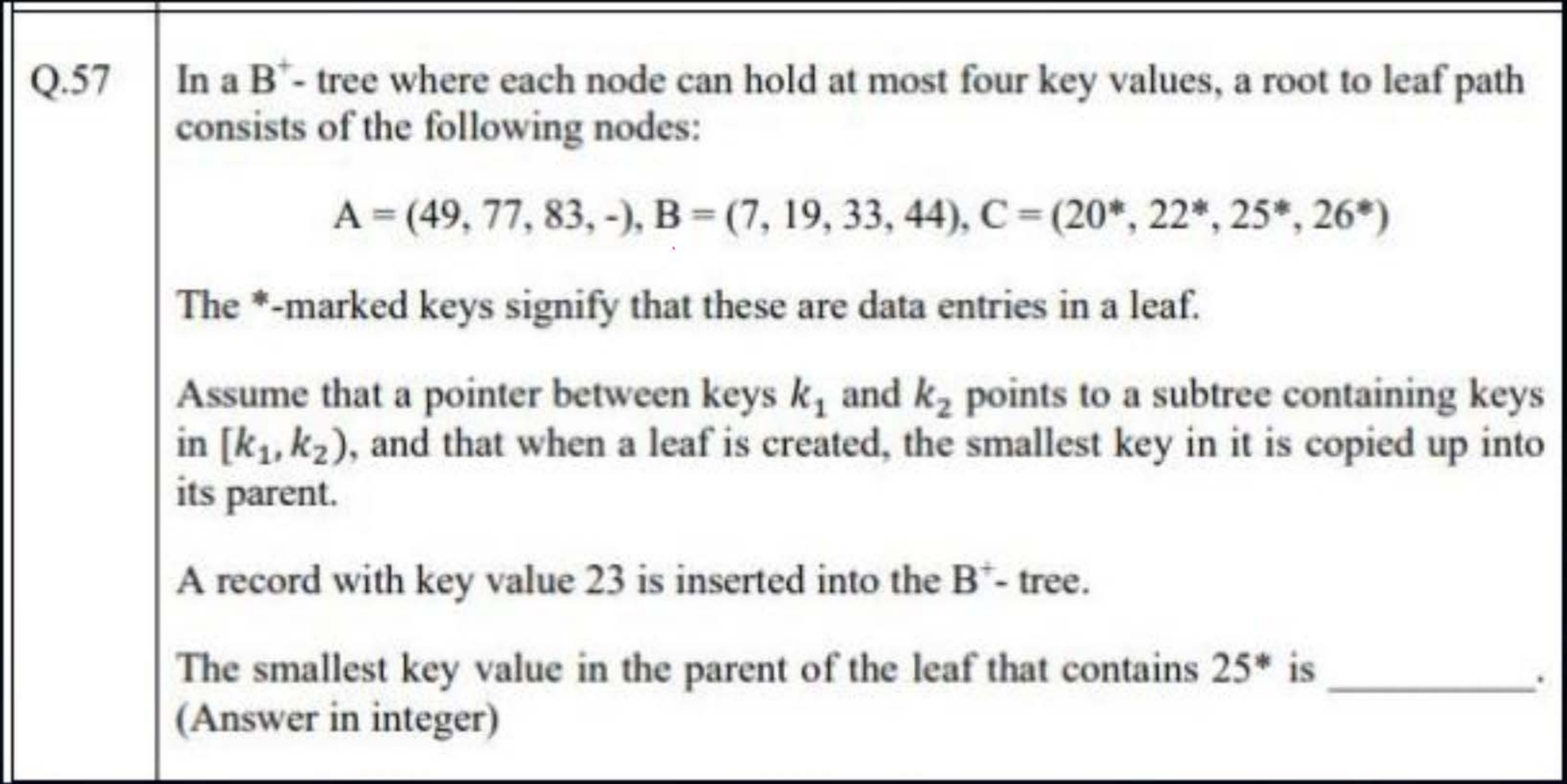


A None of the nodes will split

B At least one node will split and redistribute.

C The total number of nodes will remain same.

D The height of the tree will increase.



Q.57 In a B$^+$- tree where each node can hold at most four key values, a root to leaf path consists of the following nodes:

$$A = (49, 77, 83, -), B = (7, 19, 33, 44), C = (20^*, 22^*, 25^*, 26^*)$$

The *-marked keys signify that these are data entries in a leaf.

Assume that a pointer between keys k_1 and k_2 points to a subtree containing keys in $[k_1, k_2)$, and that when a leaf is created, the smallest key in it is copied up into its parent.

A record with key value 23 is inserted into the B$^+$- tree.

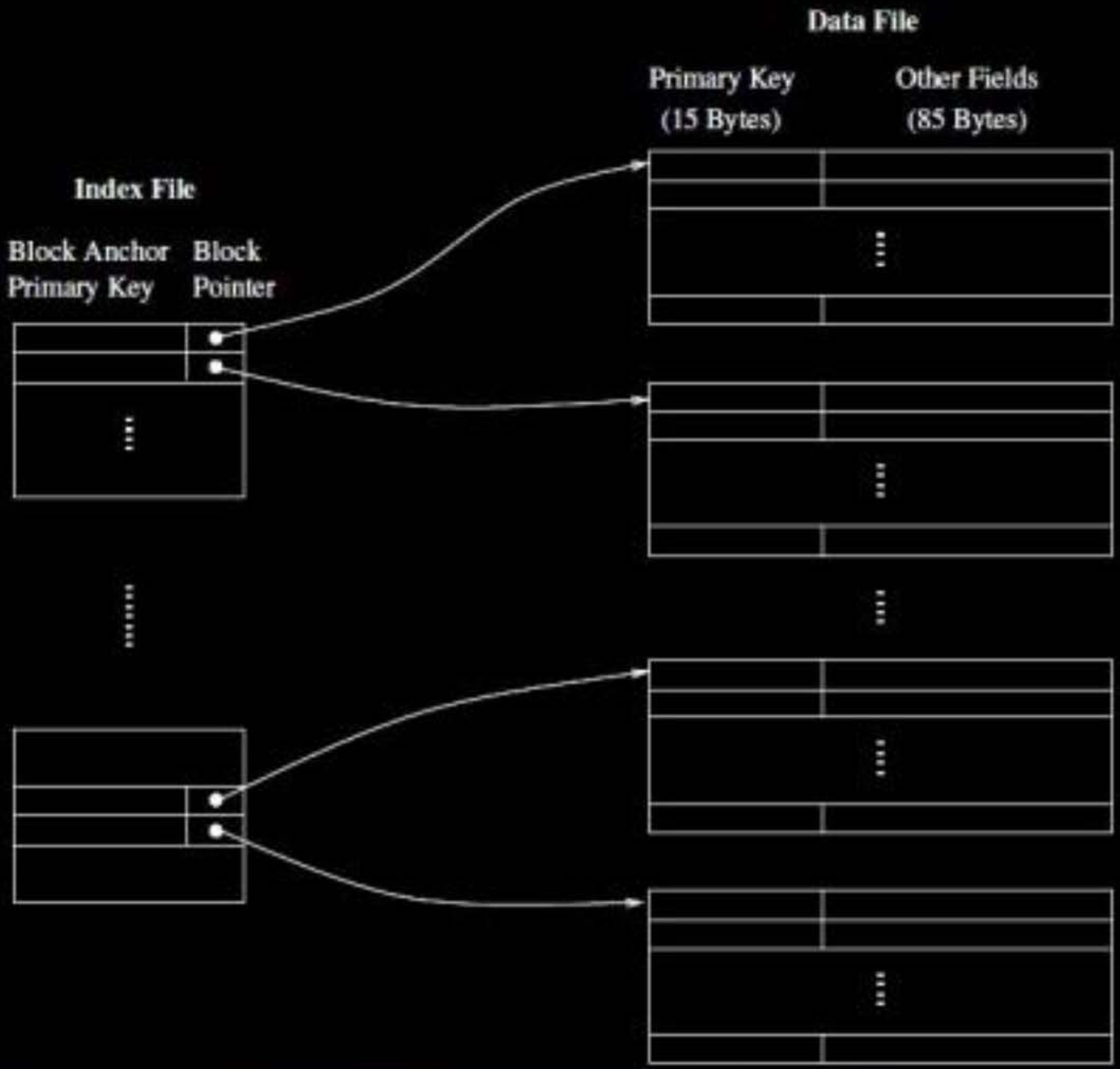
The smallest key value in the parent of the leaf that contains 25* is ____________.
(Answer in integer)

Slide

[NAT]

Consider a database of fixed-length records, stored as an ordered file. The database has 25,000 records, with each record being 100 bytes, of which the primary key occupies 15 bytes. The data file is block-aligned in that each data record is fully contained within a block. The database is indexed by a primary index file, which is also stored as a block-aligned ordered file. The figure below depicts this indexing scheme.

[GATE-2023-CS: 2M]


Data File
Primary Key
(15 Bytes)
Other Fields
(85 Bytes)
Index File
Block Anchor
Primary Key
Block
Pointer

Suppose the block size of the filo system is 1024 bytes, and a pointer to a block occupies 5 bytes. The system uses binary search on the index file to search for a record with a given key. You may assume that a binary search on an index file of b blocks takes $[\log_2 b]$ block accesses in the worst case.

Given a key, the number of block accesses required to identify the block in the data file that may contain a record with the key. in the worst case, is_____.

NAT

A data file consisting of 1,50,000 student-records is stored on a hard disk with block size of 4096 bytes. The data file is sorted on the primary key RollNo. The size of a record pointer for this disk is 7 bytes. Each student-record has a candidate key attribute called ANum of size 12 bytes. Suppose an index file with records consisting of two fields, ANum value and the record pointer to the corresponding student record, is built and stored on the same disk. Assume that the records of data file and index file are not split across disk blocks. The number of blocks in the index file is ______________

[GATE-2021-CS: 1M]

NAT

Consider a database implemented using B+ tree for file indexing and installed on a disk drive with block size of 4 KB. The size of search key is 12 bytes and the size of tree/disk pointer is 8 bytes. Assume that the database has one million records. Also assume that no node of the B+ tree and no records are present initially in main memory. Consider that each record fits into one disk block. The minimum number of disk accesses required to retrieve any record in the database is______.

[GATE-2020 (Set-1): 2M]

NAT

In a B$^+$tree, if the search-key value is 8 bytes long, the block size is 512 bytes and the block pointer size is 2 bytes, then the maximum order of the B$^+$tree is __________

[GATE-2017-CS: 2M]

MCQ

B^+ Tree are considered BALANCED because

[GATE-2016-CS: 1M]

A the length of the paths from the root to all leaf nodes are all equal.

B the lengths of the paths from the root to all leaf nodes differ from each other by at most 1

C the number of children of any two nonleaf sibling nodes differ by at most 1 .

D the number of records in any two leaf nodes differ by at most 1

NAT

Consider a B^+ tree in which the search key is 12 bytes long, block size is 1024 bytes, record pointer is 10 bytes long and block pointer is 8 bytes long. The maximum number of keys that can be accommodated in each non-leaf node of the tree is __________

[GATE-2015-CS: 2M]

MCQ

A file is organized so that the ordering of data records is the same as or close to the ordering of data entries in some index. Then that index is called.

[GATE-2015-CS: 1M]

Ⓐ Dense

Ⓑ Sparse

Ⓒ Clustered

Ⓓ Unclustered

MCQ

Consider a B^+tree in which the maximum number of keys in a node is 5 .
What is the minimum number of keys in any non-root node?

[GATE-2010-CS: 1M]

A 1

B 2

C 3

D 4

MCQ

The following key values are inserted into a B+ -tree in which order of the internal nodes is 3, and that of the leaf nodes is 2, in the sequence given below. The order of internal nodes is the maximum number of tree pointers in each node, and the order of leaf nodes is the maximum number of data items that can be stored in it. The B+ -tree is initially empty.

10, 3, 6, 8, 4, 2, 1

The maximum number of times leaf nodes would get split up as a result of these insertions is

[GATE-2009-CS: 2M]

Ⓐ 2

Ⓑ 3

Ⓒ 4

Ⓓ 5

MCQ

Consider a file of 16384 records. Each record is 32 bytes long and its key field is of size 6 bytes. The file is ordered on a non-key field, and the file organization is unspanned. The file is stored in a file system with block size 1024 bytes, and the size of a block pointer is 10 bytes. If the secondary index is built on the key field of the file, and a multilevel index scheme is used to store the secondary index, the number of first-level and second-level blocks in the multilevel index are respectively-

[GATE-2008-CS: 2M]

- A 8 and 0
- B 128 and 6
- C 256 and 4
- D 512 and 5

MCQ

A clustering index is defined on the fields which are of type

[GATE-2008-CS: 1M]

A non-key and ordering

B non-key and non-ordering

C key and ordering

D key and non-ordering

MCQ

A B-tree of order 4 is built from scratch by 10 successive insertions. What is the maximum number of node splitting operations that may take place?

[GATE-2008-CS: 2M]

A 3

B 4

C 5

D 6

2 mins Summary

Topic Insertion into B+ tree

Topic Deletion from B+ tree

Slide

THANK - YOU

Slide