CS & IT ENGINEERING

Database Management System

Lecture No. 11

By- Ravindrababu Ravula Sir

Recap of Previous Lecture

Topic File Org and Indexing

Topic Intro of B, B+ trees

Topics to be Covered

Topic	B-Tree Definition
Topic	B+ Tree
Topic	Node Structure of B+ Tree Node
Topic	Left Biasing and Right Biasing in B+ Tree
Topic	Sequential Access of a Range of Record
Topic	Problems
Topic	Bulk Loading B^+ Tree

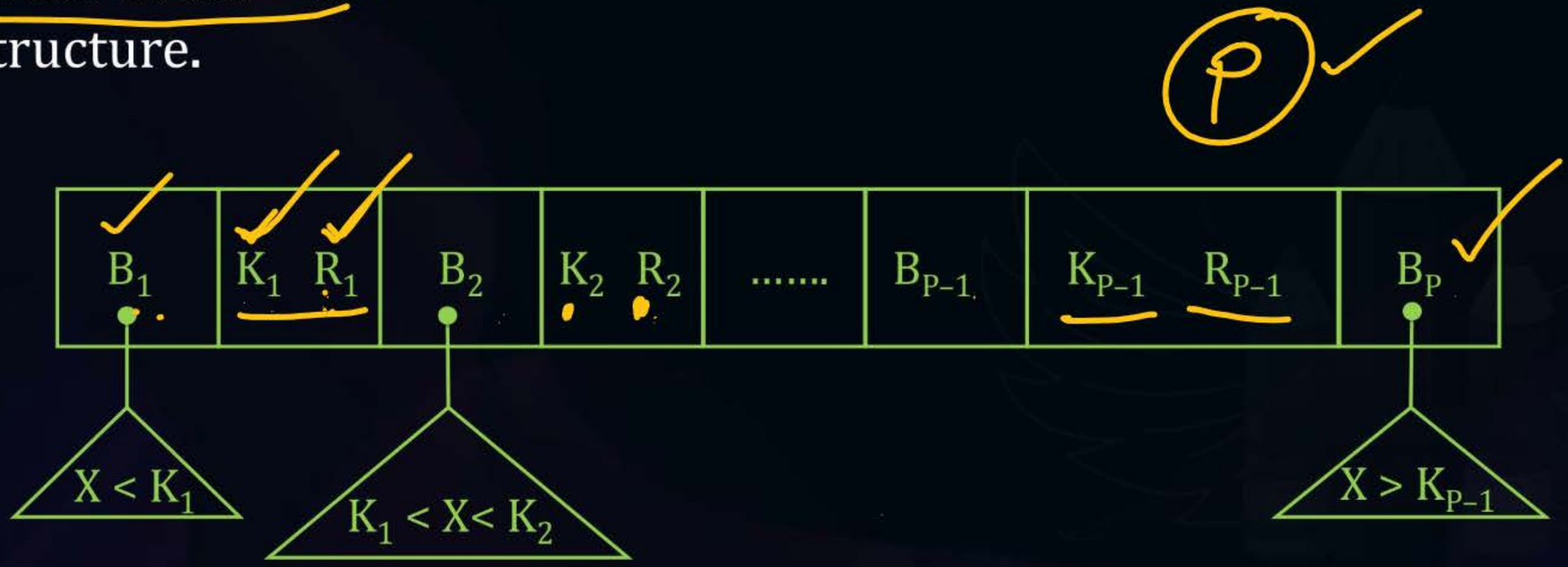
Topic: B-Tree Definition

B-Tree Definition:

Maximum possible block pointers which can be stored in a B-Tree node is the Order of that block.

Assume the Order = P.

1. Node Structure.



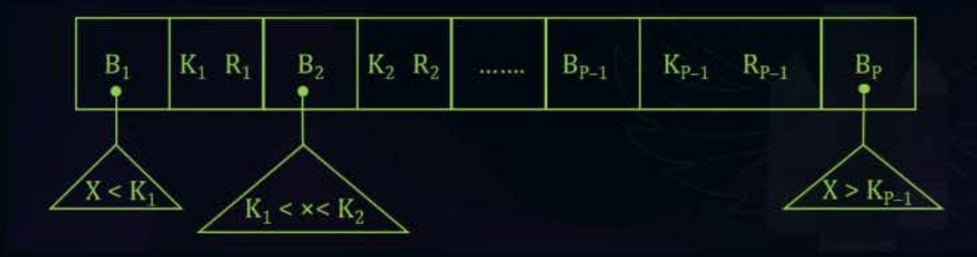
Topic: B-Tree Definition

Say the order is P.

⇒ Then there will be P block pointer.

⇒ P block pointers = (P-1) Keys

= (P-1) Record/Data pointer.



Block pointer / Tree pointer:

Pointer which point to index node or Tree node

Data pointer / Record pointer:

Pointer which point to database record or DB block.

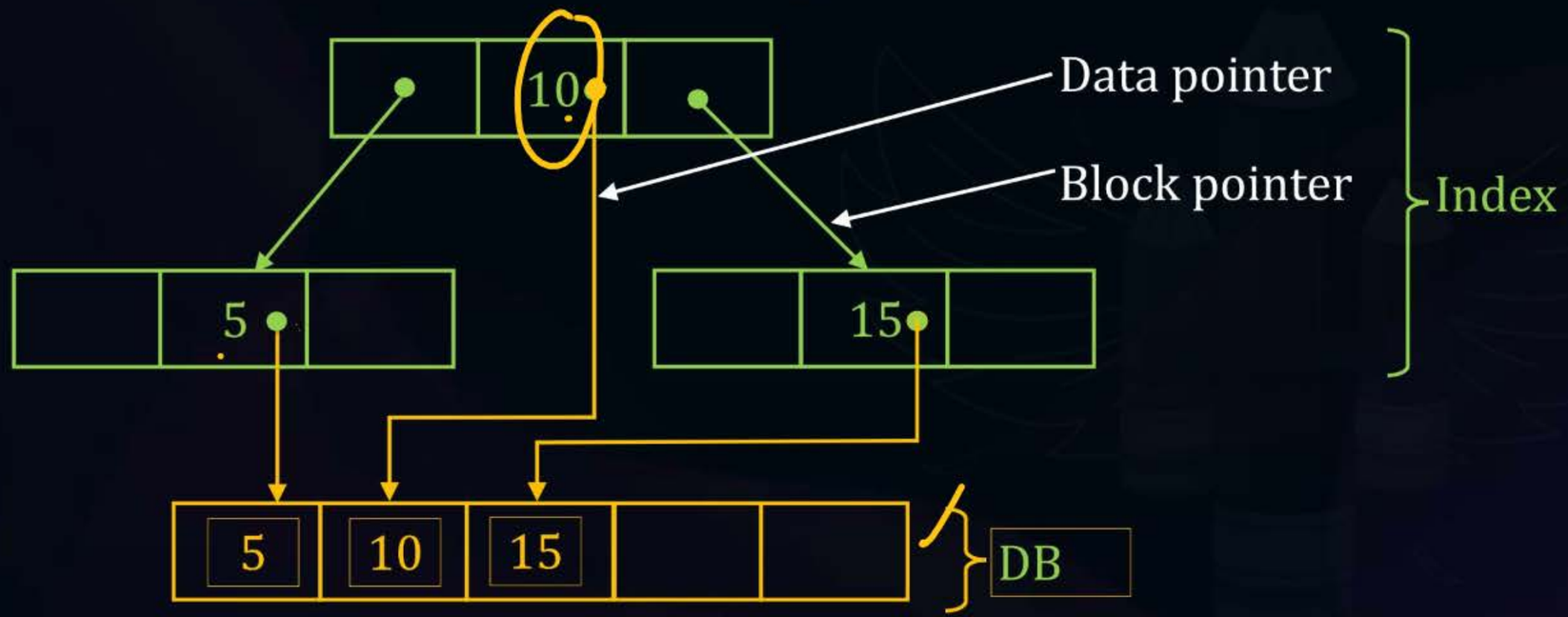
Topic: B-Tree Definition

⇒ If there is one Key in the node, we will have two block pointers.

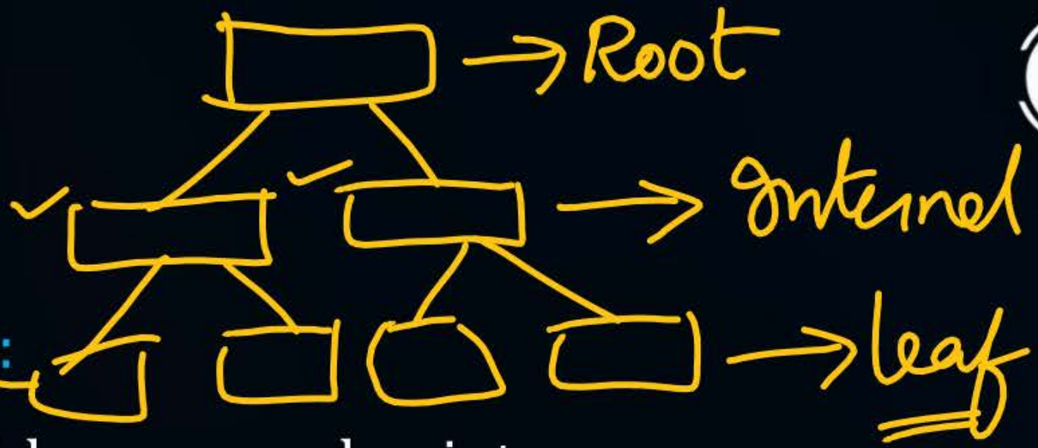
If there are two keys in the node, we will have three block pointers.

Hence, no. of Keys will be one less than the no. of block pointers.

Examples:



Topic: B-Tree Definition



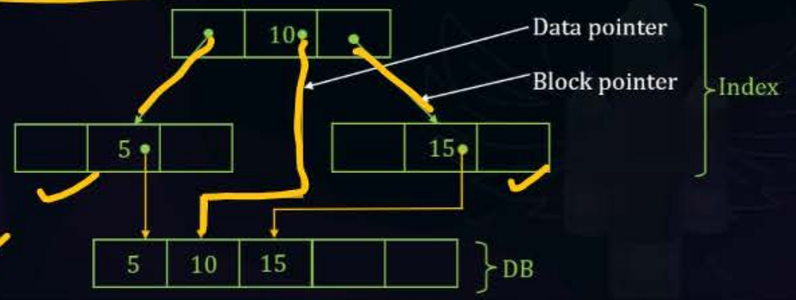
2. Every internal node except root, must have:

   At Least $\lceil P/2 \rceil$ Block Pointers with $\lceil P/2 \rceil - 1$ keys, record pointers

   At Most P Block Pointers with (P-1) Keys, record pointers

3. Root node must have:

   - At Least 2 Block Pointers with 1 Key, R_p
   - At Most P Block Pointers with (P-1) Keys, R_p



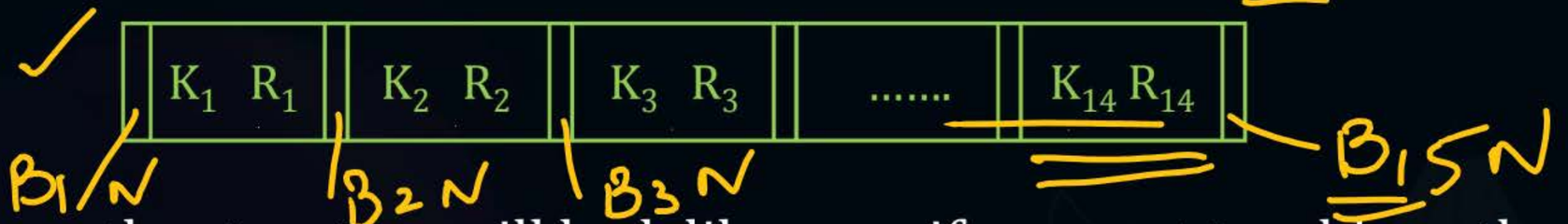
4. Every Leaf node must be at same level.

All these 3 points are the balancing condition for B Trees.

Topic: B-Tree Definition

Example:

Order P: 15 (Max Block pointer)

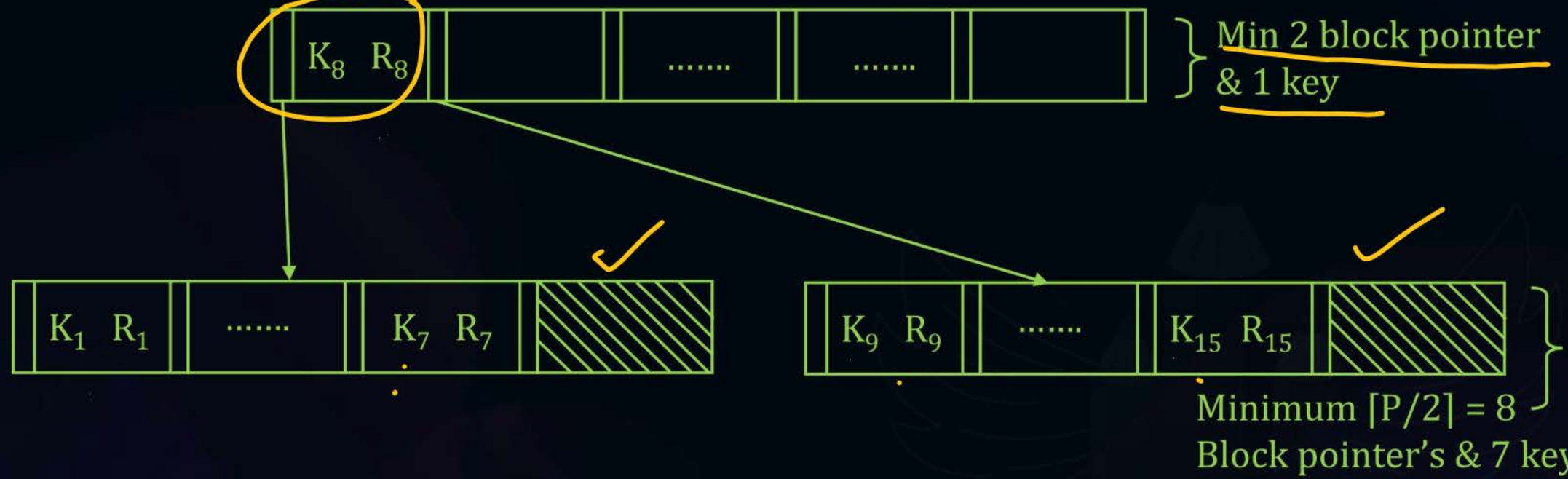


This is how the structure will look like, now if we want to add another key K_{15} R_{15}, then we will have to do **Node Splitting**.

$\Rightarrow$ $K_1 < K_2 < K_3 < .... K_8 < ... < K_{15}$

Topic: B-Tree Definition

We will split nodes from the middle node, in this case it is K_8.



⇒ We did node splitting so that we could add incoming record, but we should be mindful about Balancing Condition, the three points we discussed earlier.

Topic: B-Tree Definition

Except Root, every node should be 50% occupied.

⇒ If the occupancy is below 50%, for internal node then it would mean there is no need for node splitting and the incoming record can be added to the original nodes.

Inspiring Stories : Tongan Castaways

Background: **6 boys from Tonga went out to sea.**

Struggles: **Boat drifted on island for 15 months.**

Achievements: **Built huts, caught fish, kept fire alive.**

Impact: **Survived through teamwork and hope.**

Topic: B-Tree Definition

✓ Key, Rp → 4

#Q. Construct B Tree with Order P: 5 (max bp) bp → Block pointers & Sequence of Keys: 40, 60, 20, 50, 70, 90, 80,75,10,15,25,55,65, 68, 85, 95, 98

Topic: B-Tree Definition

#Q. Construct B Tree with Order P: 5 (max bp) bp → Block pointers & Sequence of Keys: 40, 60, 20, 50, 70, 90, 80,75,10,15,25,55,65, 68, 85, 95, 98

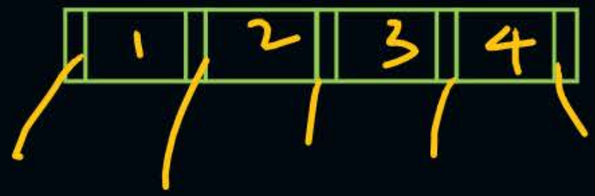
Sol: Order P is 5 → 5 Block pointer

⇒ We can have → 4 keys

→ 4 record /data pointer

Topic: B-Tree Definition

#Q. Construct B Tree with Order P: 5 (max bp) bp → Block pointers & Sequence of Keys: 40, 60, 20, 50, 70, 90, 80,75,10,15,25,55,65, 68, 85, 95, 98



Sol: Order P is 5 → 5 Block pointer

⇒ We can have → 4 keys

→ 4 record /data pointer

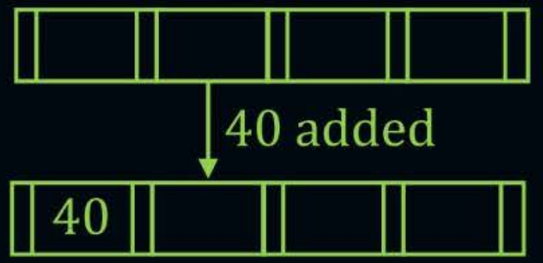
Topic: B-Tree Definition

#Q. Construct B Tree with Order P: 5 (max bp) bp → Block pointers & Sequence of Keys:
40, 60, 20, 50, 70, 90, 80,75,10,15,25,55,65, 68, 85, 95, 98

Sol: Order P is 5 → 5 Block pointer

⇒ We can have → 4 keys

→ 4 record /data pointer



Topic: B-Tree Definition

#Q. Construct B Tree with Order P: 5 (max bp) bp → Block pointers & Sequence of Keys: 40, 60, 20, 50, 70, 90, 80,75,10,15,25,55,65, 68, 85, 95, 98

Sol: Order P is 5 → 5 Block pointer

⇒ We can have → 4 keys

→ 4 record /data pointer

Topic: B-Tree Definition

#Q. Construct B Tree with Order P: 5 (max bp) bp → Block pointers & Sequence of Keys: 40, 60, 20, 50, 70, 90, 80,75,10,15,25,55,65, 68, 85, 95, 98

Sol: Order P is 5 → 5 Block pointer

⇒ We can have → 4 keys

→ 4 record /data pointer

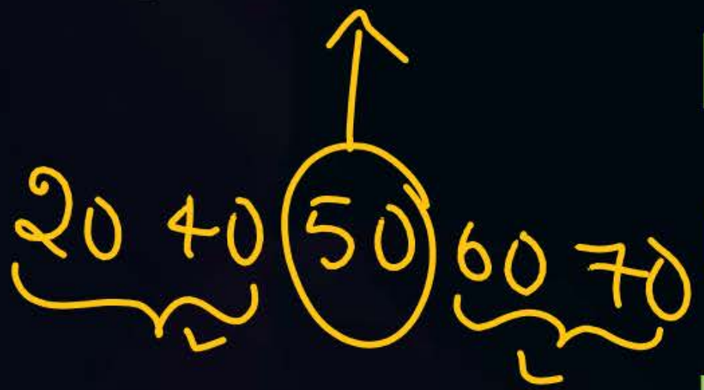
Topic: B-Tree Definition

#Q. Construct B Tree with Order P: 5 (max bp) bp → Block pointers & Sequence of Keys: 40, 60, 20, 50, 70, 90, 80,75,10,15,25,55,65, 68, 85, 95, 98

Sol: Order P is 5 → 5 Block pointer

⇒ We can have → 4 keys

→ 4 record /data pointer



Topic: B-Tree Definition

#Q. Construct B Tree with Order P: 5 (max bp) bp → Block pointers & Sequence of Keys: 40, 60, 20, 50, 70, 90, 80,75,10,15,25,55,65, 68, 85, 95, 98

Sol: Order P is 5 → 5 Block pointer

⇒ We can have → 4 keys

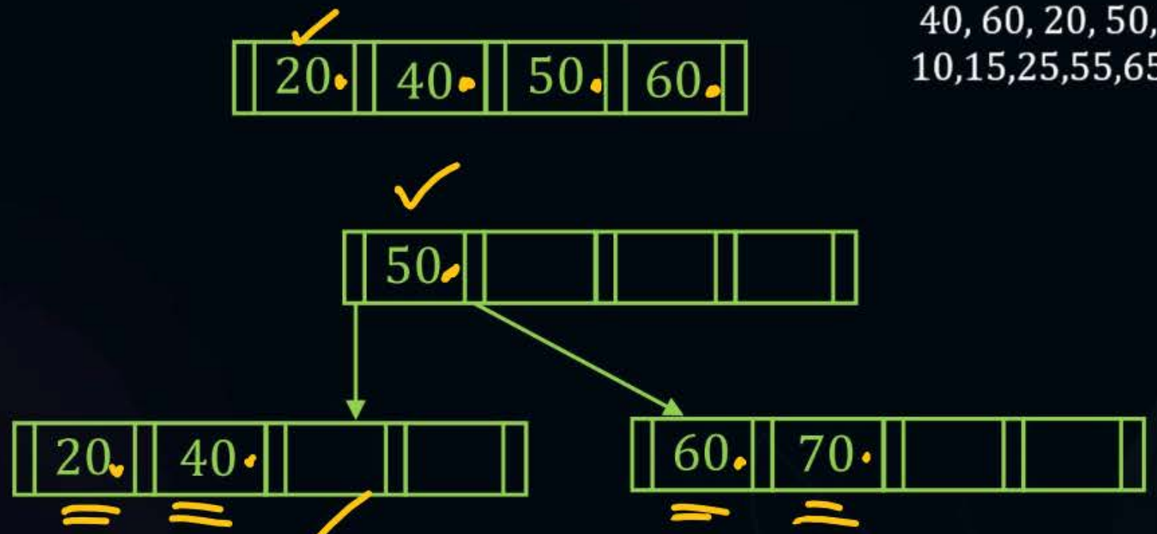
→ 4 record /data pointer

Fully occupied,
can't add 70 without node splitting.

Topic: B-Tree Definition

#Q. Construct B Tree with Order P: 5 (max bp)
bp → Block pointers & Sequence of Keys:
40, 60, 20, 50, 70, 90, 80,75,
10,15,25,55,65, 68, 85, 95, 98

NODE SPLITTING :



Say, we also want to add 90, 80, 75

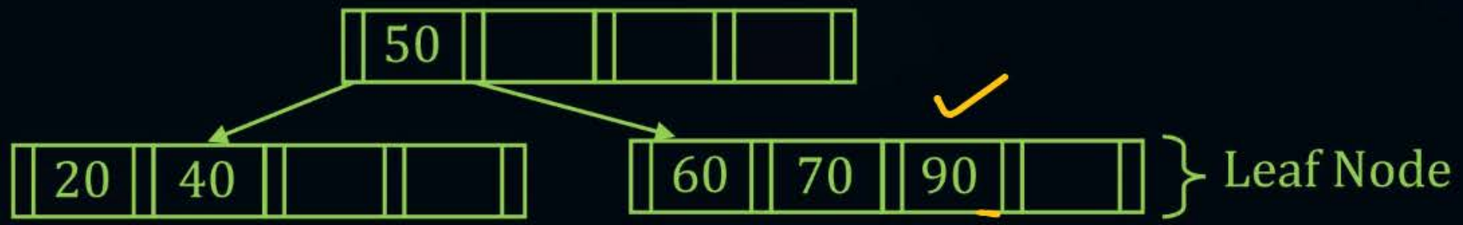
⇒ For the incoming Keys, they should **not** be directly placed at root level, even though they can be accomodated in root level, because it will make the tree unbalanced.

⇒ Key must also should not get directly into internal node.

⇒ Key must only be inserted into **leaf node**.

Topic: B-Tree Definition

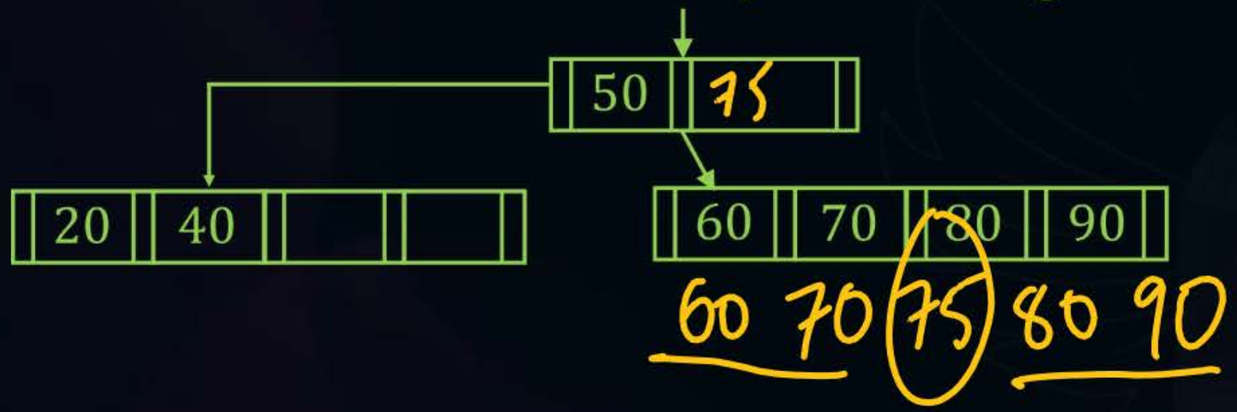
#Q. Construct B Tree with Order P: 5 (max bp)
bp → Block pointers & Sequence of Keys:
40, 60, 20, 50, 70, 90, 80,75,
10,15,25,55,65, 68, 85, 95, 98



Topic: B-Tree Definition

#Q. Construct B Tree with Order P: 5 (max bp)
bp → Block pointers & Sequence of Keys:
40, 60, 20, 50, 70, 90, 80, 75,
10, 15, 25, 55, 65, 68, 85, 95, 98

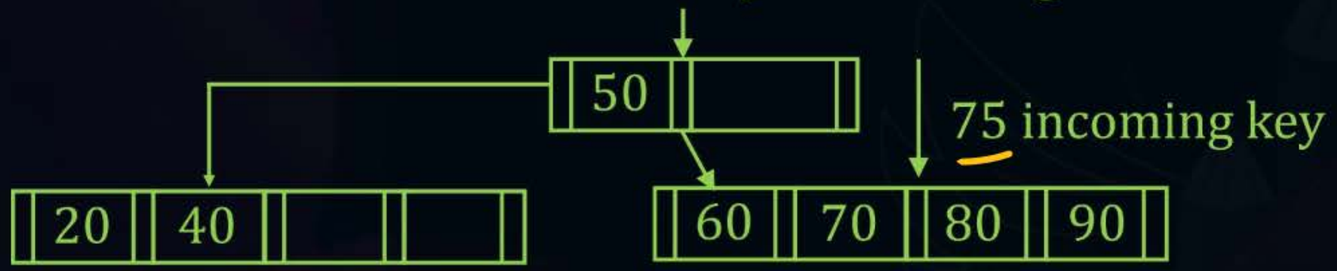
We will have to shift 90, one position to right



Topic: B-Tree Definition

#Q. Construct B Tree with Order P: 5 (max bp)
bp → Block pointers & Sequence of Keys:
40, 60, 20, 50, 70, 90, 80,75,
10,15,25,55,65, 68, 85, 95, 98

We will have to shift 90, one position to right



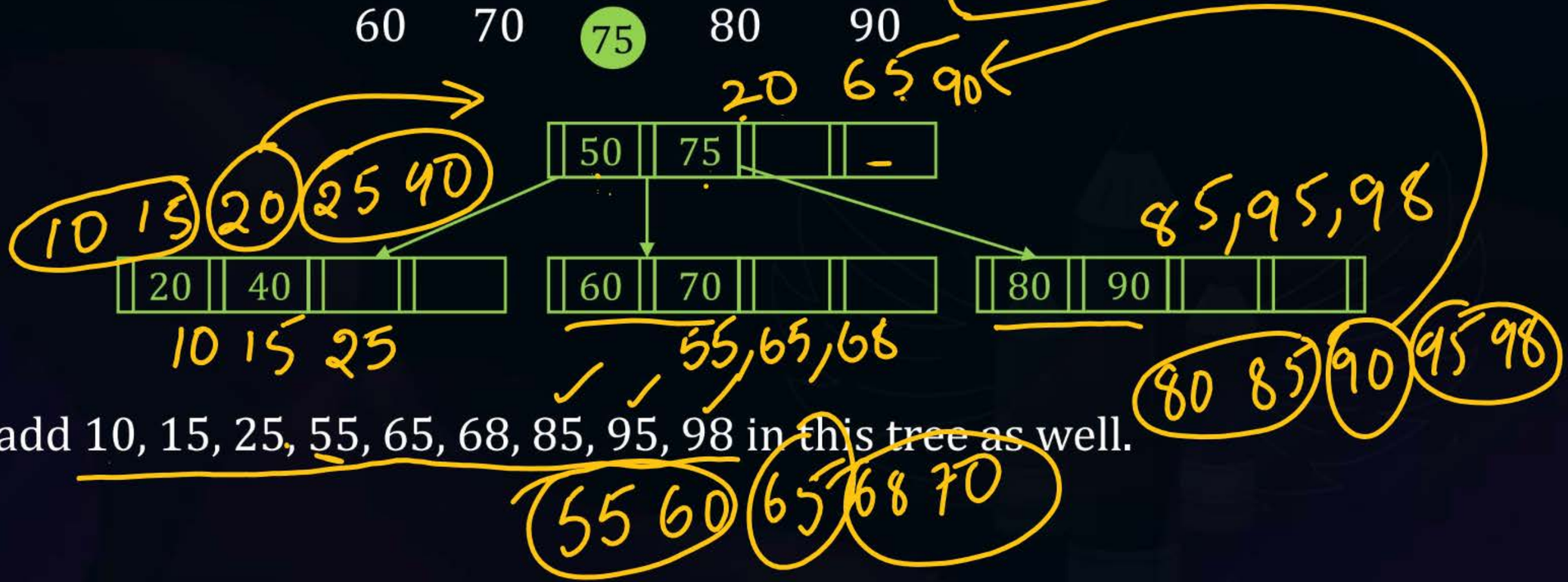
Topic: B-Tree Definition

#Q. Construct B Tree with Order P: 5 (max bp)
bp → Block pointers & Sequence of Keys:
40, 60, 20, 50, 70, 90, 80,75,
10,15,25,55,65, 68, 85, 95, 98

To accommodate 75, we will have to split the node

60 70 80 90 ← 75

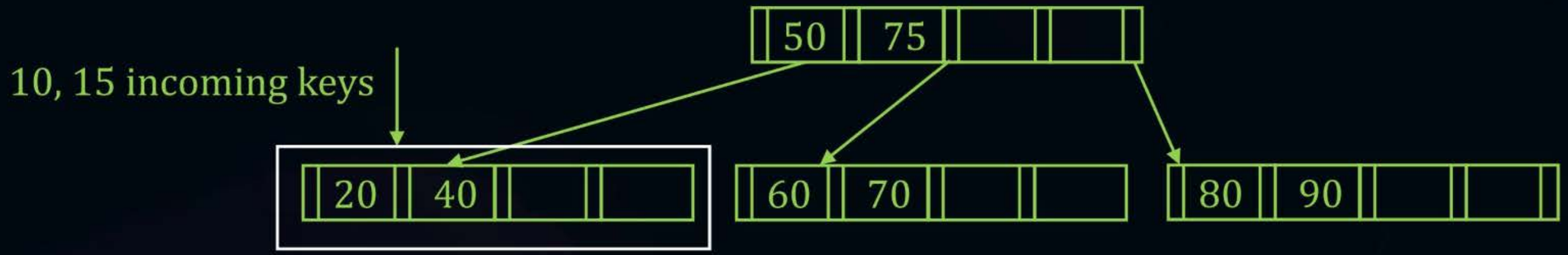
60 70 75 80 90



Now, let's add 10, 15, 25, 55, 65, 68, 85, 95, 98 in this tree as well.

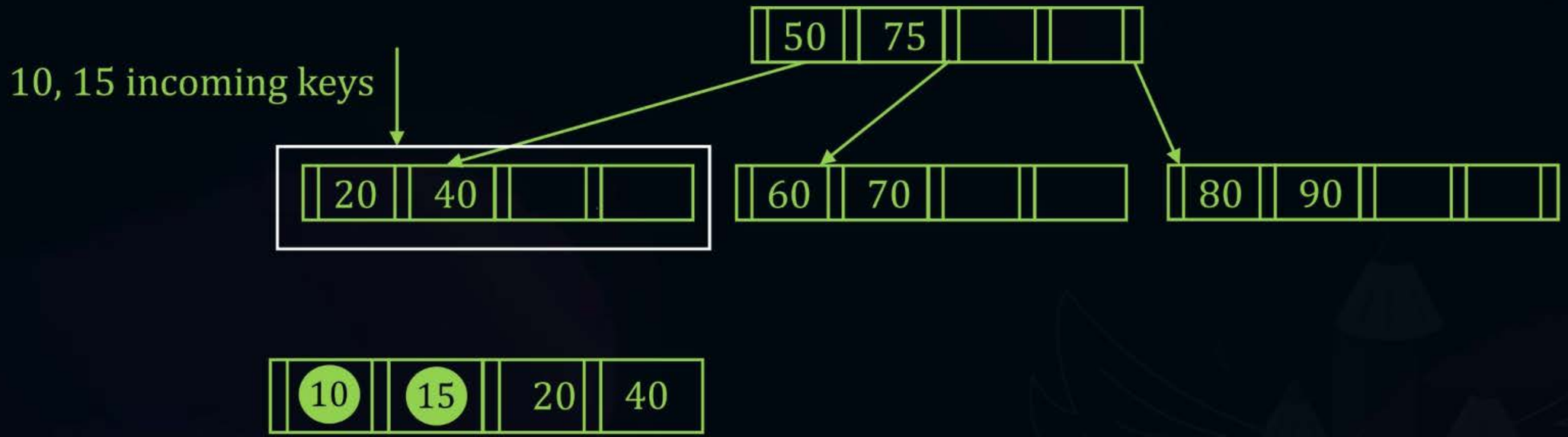
Topic: B-Tree Definition

#Q. Construct B Tree with Order P: 5 (max bp)
bp → Block pointers & Sequence of Keys:
40, 60, 20, 50, 70, 90, 80,75,
10,15,25,55,65, 68, 85, 95, 98



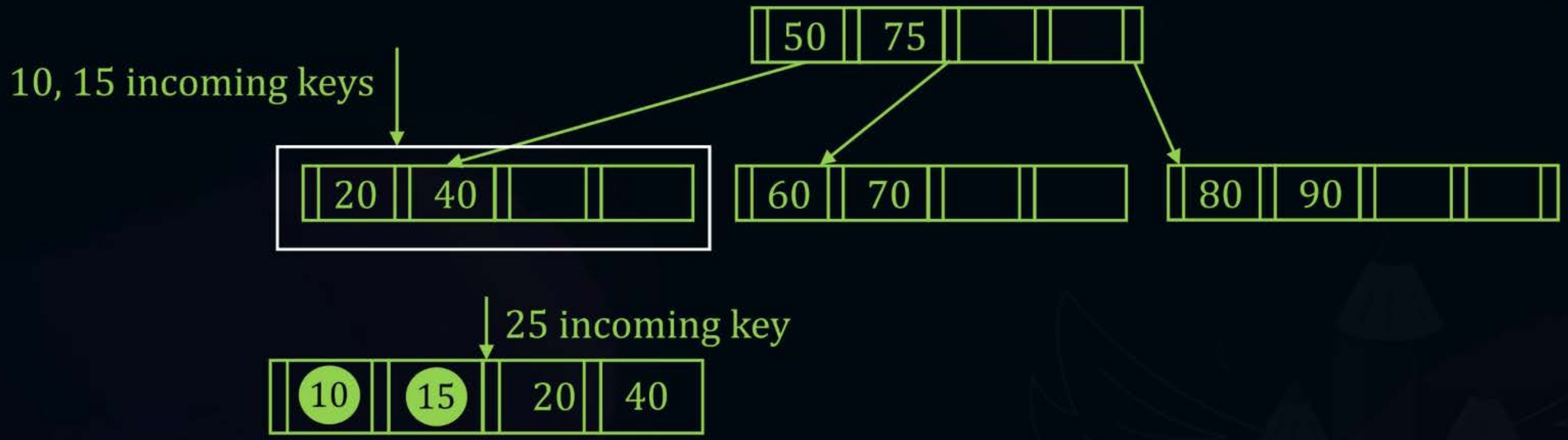
Topic: B-Tree Definition

#Q. Construct B Tree with Order P: 5 (max bp)
bp → Block pointers & Sequence of Keys:
40, 60, 20, 50, 70, 90, 80,75,
10,15,25,55,65, 68, 85, 95, 98



Topic: B-Tree Definition

#Q. Construct B Tree with Order P: 5 (max bp)
bp → Block pointers & Sequence of Keys:
40, 60, 20, 50, 70, 90, 80,75,
10,15,25,55,65, 68, 85, 95, 98

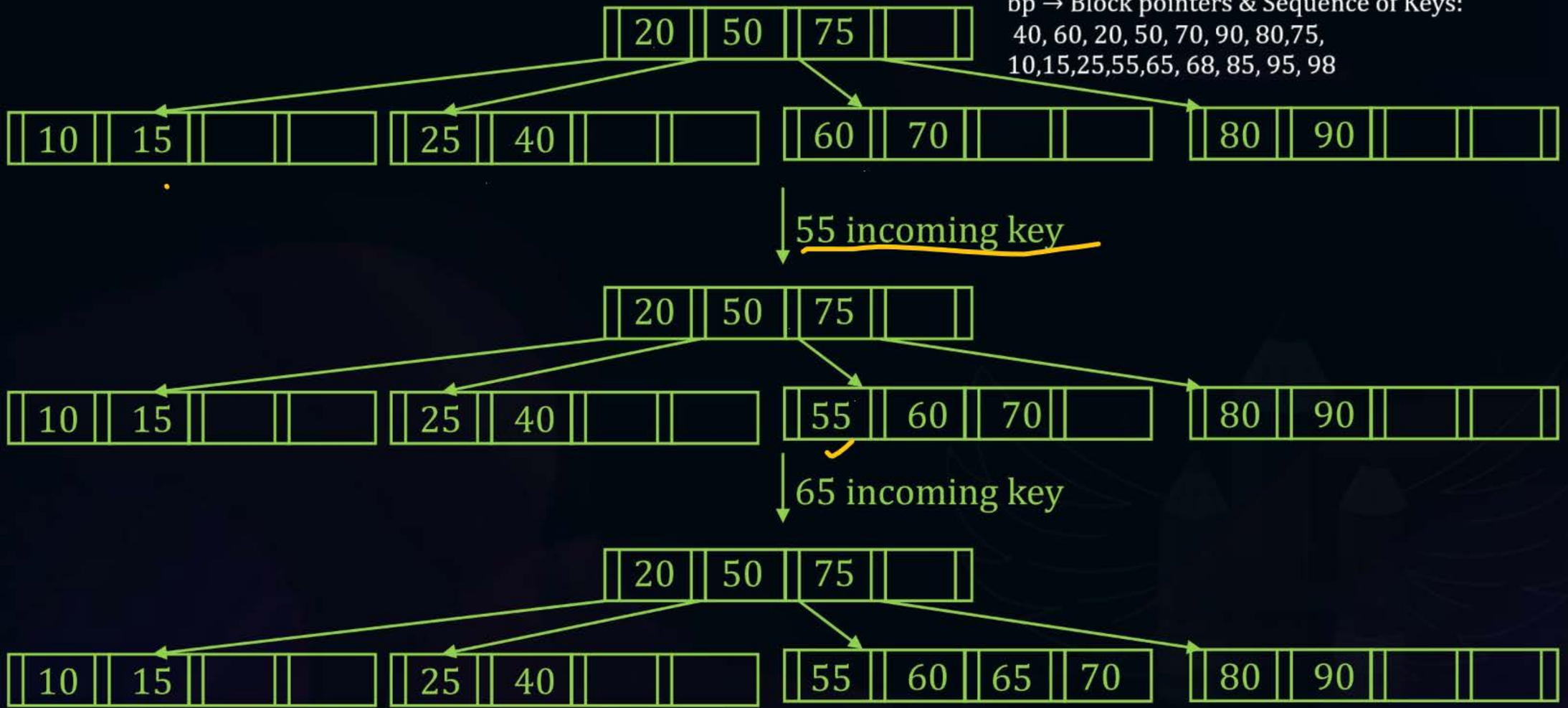


Node splitting will happen

10, 15, 20 25, 40

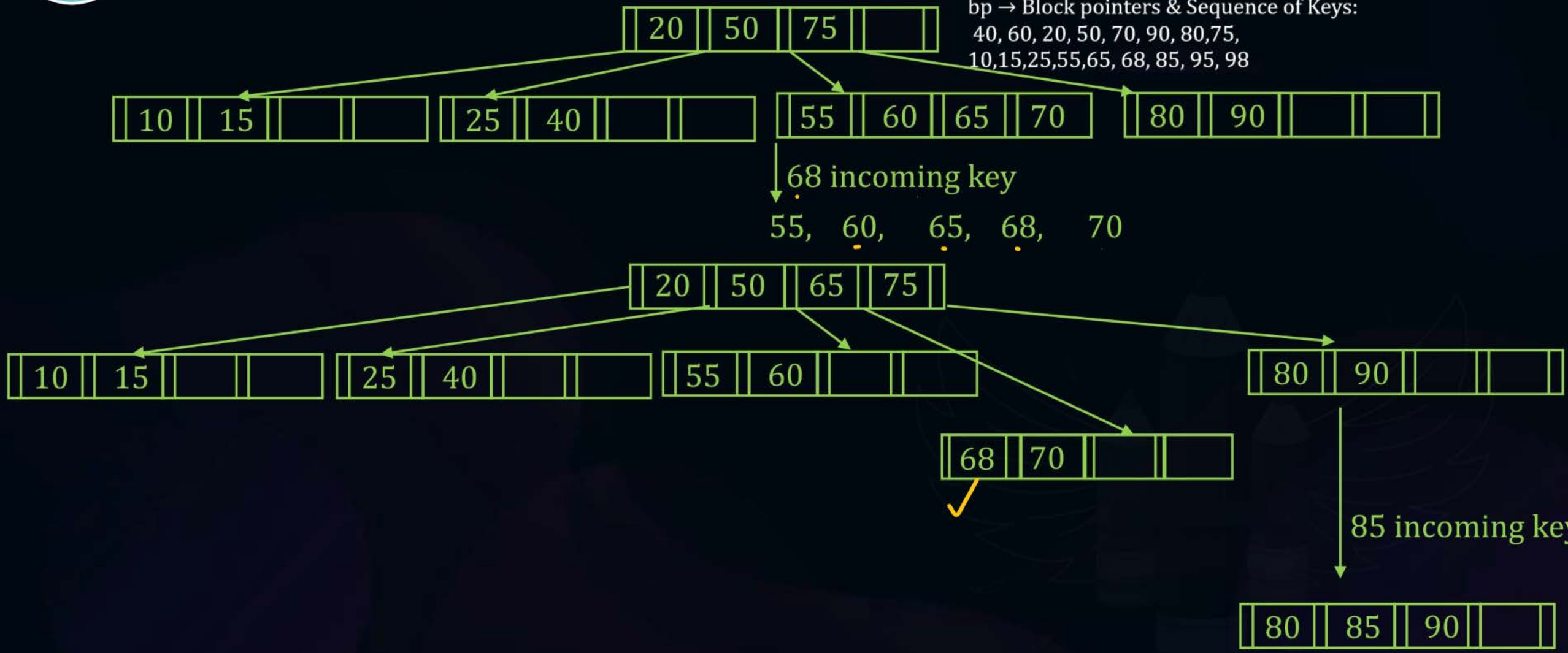
Topic: B-Tree Definition

#Q. Construct B Tree with Order P: 5 (max bp)
bp → Block pointers & Sequence of Keys:
40, 60, 20, 50, 70, 90, 80,75,
10,15,25,55,65, 68, 85, 95, 98



Topic: B-Tree Definition

#Q. Construct B Tree with Order P: 5 (max bp)
bp → Block pointers & Sequence of Keys:
40, 60, 20, 50, 70, 90, 80,75,
10,15,25,55,65, 68, 85, 95, 98



Topic: B-Tree Definition

Topic: B-Tree Definition

Topic: B-Tree Definition

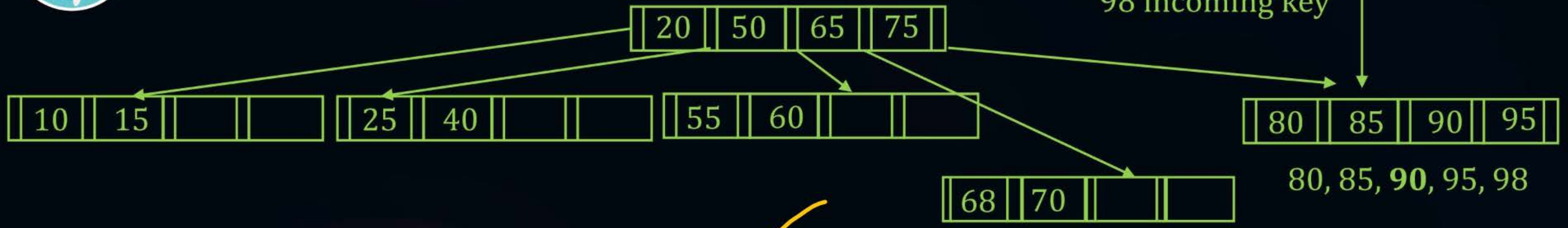
20, 50, **65**, 75, 90

80, 85, **90**, 95, 98

Topic: B-Tree Definition

20, 50, **65**, 75, 90

98 incoming key



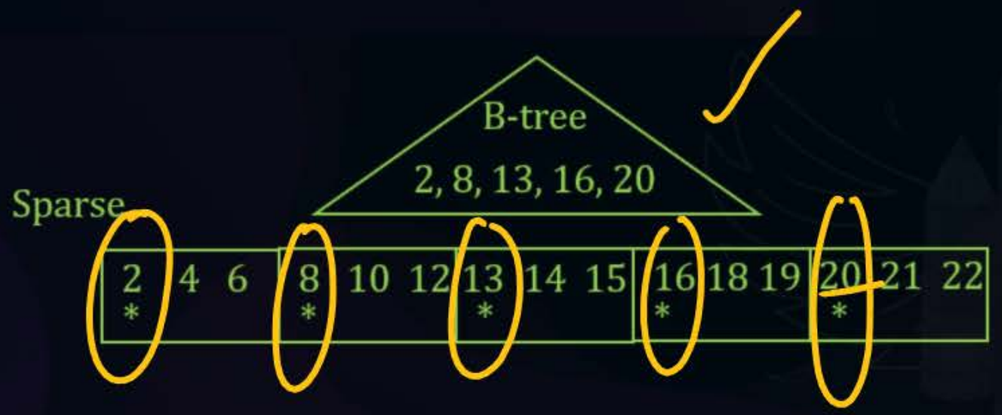
80, 85, **90**, 95, 98

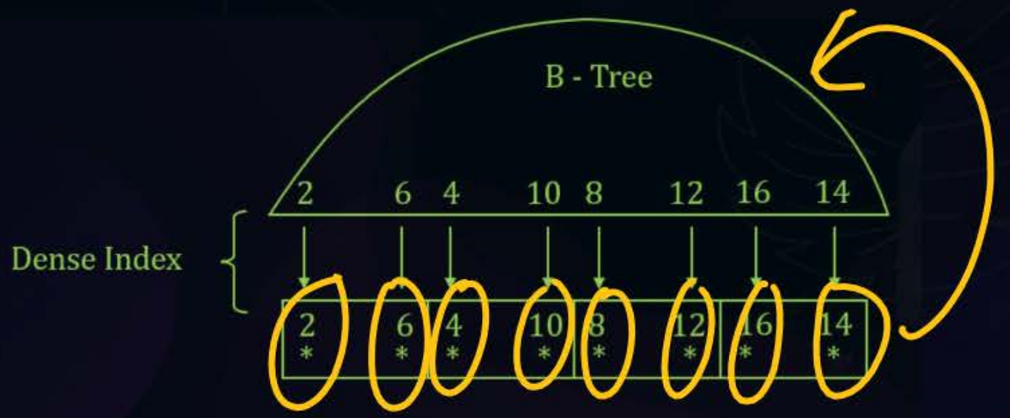
Topic: B-Tree Definition

⇒ B Tree index can be sparse index or dense index based on how database file is ordered.

1. If ordered based on Search Key. ⇒ Sparse Index
2. If Unordered ⇒ Dense Index.

1. Ordered based on Search Key.
2. If Database file is used for indexing is unordered field, then corresponding B-Tree index become dense index.





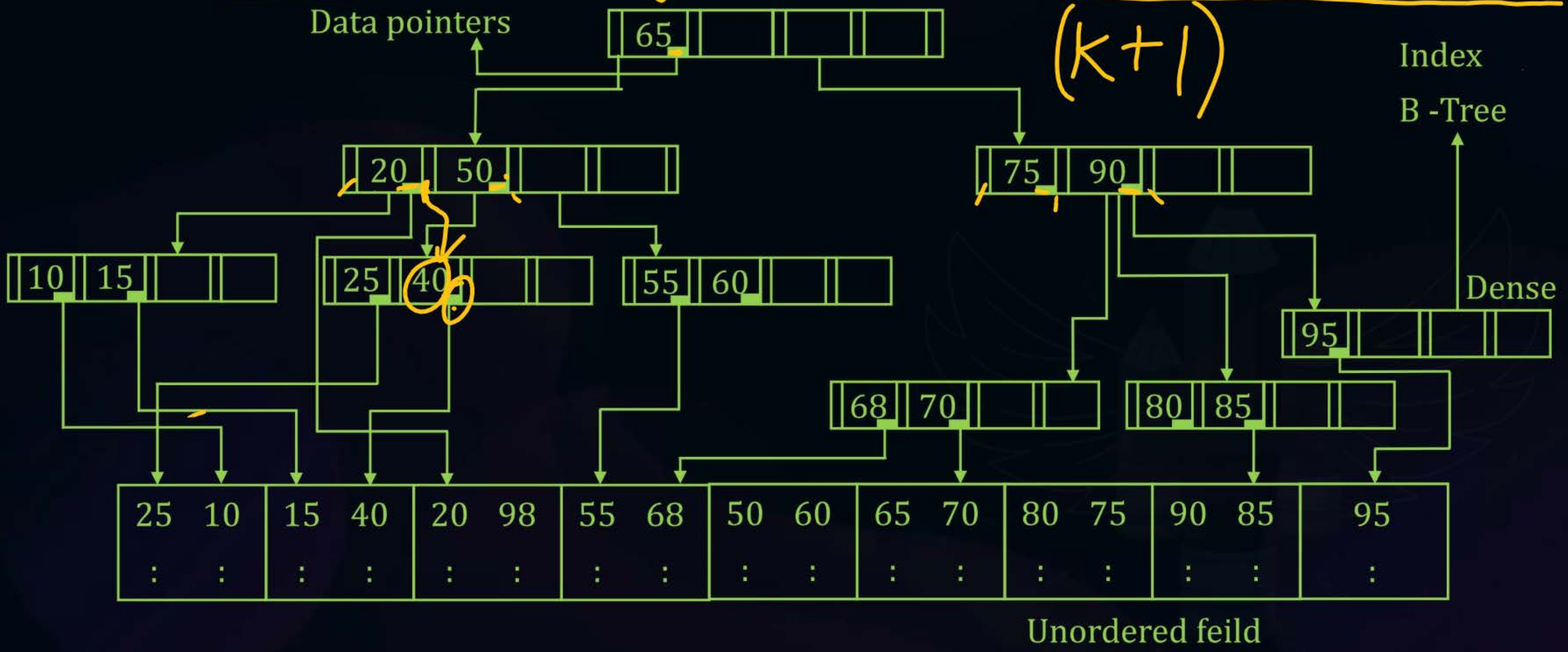
Topic: B-Tree Definition

$P = 5 \quad \lceil P/2 \rceil = 3$

$3 - 1 = 2$ k, v

Let Say the data in DB file is Unordered. This is how it will organized in Index Tree."

(k+1)



Topic: B-Tree Definition

If the attribute that we use for indexing is unordered in DB file, then corresponding indices become DENSE index, hence, we will need to build inde each key of the record.

The data pointers present in index file points to database and this index file database files both are stored in DISK.

Topic: B-Tree Definition

Conclusion:

1. **Advantage:**
   - B-Tree index is best suitable for randomly accessing a record [single record].
   - Example: Select * From R WHERE A = 40;
   - For finding 40, we will have to go through 3 levels, and then accessing from DB file.
   - Worst case I/O cost: ((K+1)) blocks.
2. **Disadvantage:**
   - B-Tree index is not suitable for Sequential access of a Range of records.
   - Example: Select * From R WHERE A ≥ 68 and A ≤ 85 (This is a Range, say X records)
   - We will have to find each record individually.
   - Worst case I/O cost: (X(K+1)) blocks.
   - Each record needs to be accessed in the range, from root to DB file.

Inspiring Stories : Nenets Nomads

Background: **Native people of Russian Arctic.**

Struggles: **Live in –50°C, move across tundra.**

Achievements: **Travel 1,000 km each year, keeping traditions alive.**

Impact: **Show human strength in extreme cold.**

Topic: B-Tree Definition

- To avoid this drawback, small change is introduced in the structure of B Tree, which is named as B^+ Tree.
- The Goal of B^+ Tree is that it should be suitable for both Random access of record, as well as sequential access of range of record.

Topic: B^+ Tree

Definition:

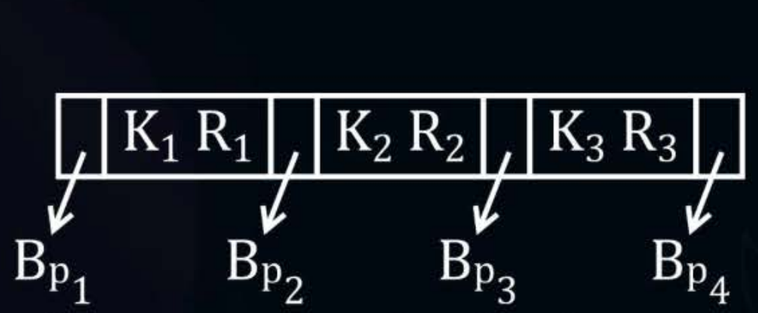
- Order of B^+ Tree: Maximum Possible Pointers which can be stored in a B+ Tree node.

Topic: B^+ Tree

1. Node Structure:

- Node structure for both Internal node and leaf node are same in B Tree.

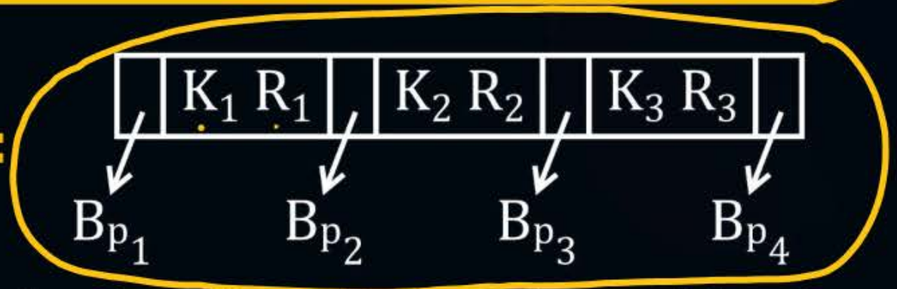
  Order p : 4



- We have different Node structure for leaf node and internal node in case of B^+ Trees.

Topic: Node Structure of B+ Tree Node

Internal Node of B Trees:



B+

Leaf Node: we have a set of pairs (key, R_p) and one block pointer which points to next (adjacent) leaf node and every key should be present in the leaf node

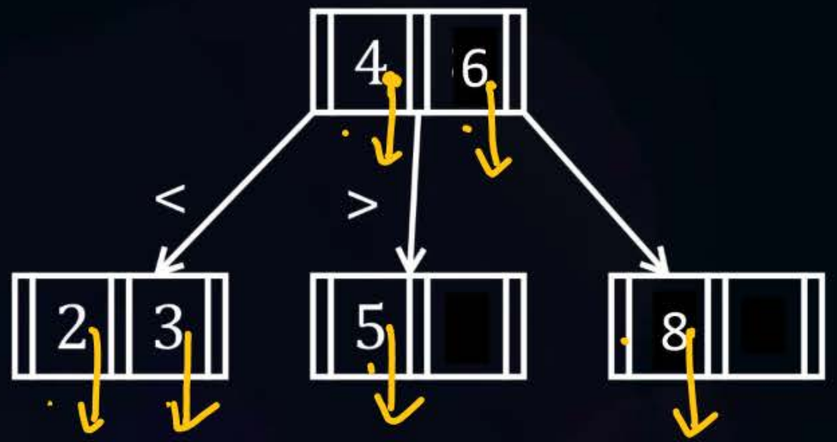
Internal Node: For internal node of B+ tree, we have only keys and block pointers.

Topic: Node Structure of B+ Tree Node

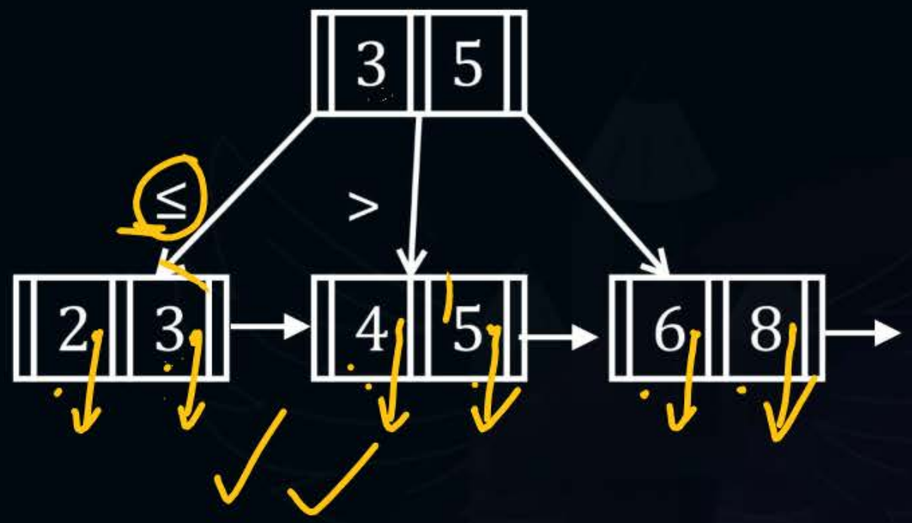
Difference in Structure:

All keys are present in the leaf nodes for B+ trees

B Tree

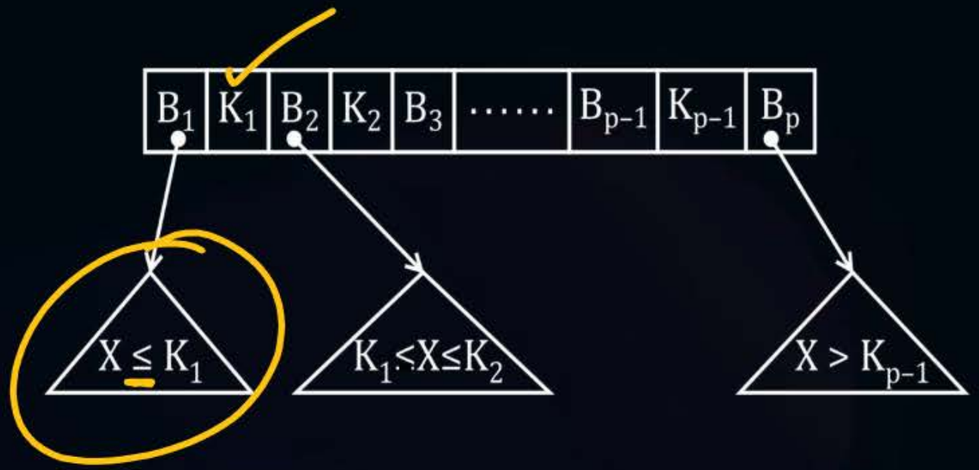


B+ Tree

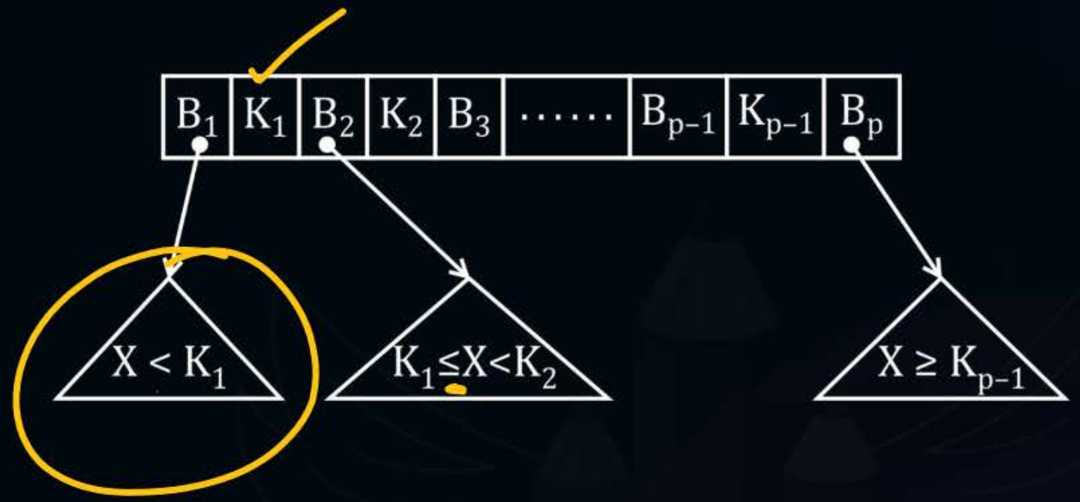


Topic: Left Biasing and Right Biasing in B^+ Tree

Left Biasing:



Right Biasing:



Topic: B+ Tree

Balancing condition of B+ Trees is same as B Tree:

2. Every internal node except Root must have : (let the order be p)

   → Atleast $\lceil p/2 \rceil$ block pointer's with $\lceil p/2 \rceil - 1$ key.

   → Atmost p block pointer's with (p – 1) keys.

3. Root can be:

   → Atleast 2 block pointers with 1 key.

   → Atmost p block pointers with (p – 1) keys.

4. Every leaf node must be at same level.

Topic: Leaf Node Splitting

Order p : 15

→ Say we want to insert $K_{15}R_{15}$ in the below nodes:

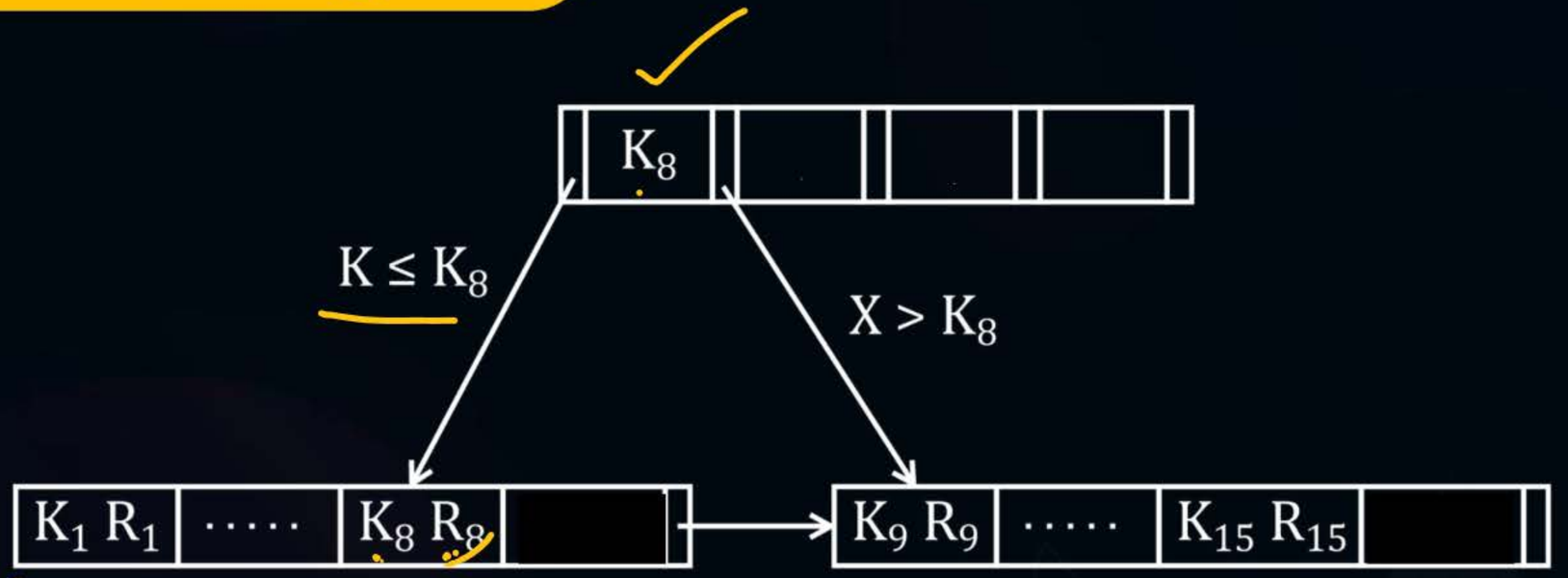
→ We have no empty node so we will have to split the nodes just like we did in B Tree node.

Topic: Leaf Node Splitting

→ But while splitting a leaf node all the keys and record pointers will remain in the leaf level but the largest key from the left block side will be promoted to upper level (left biasing)

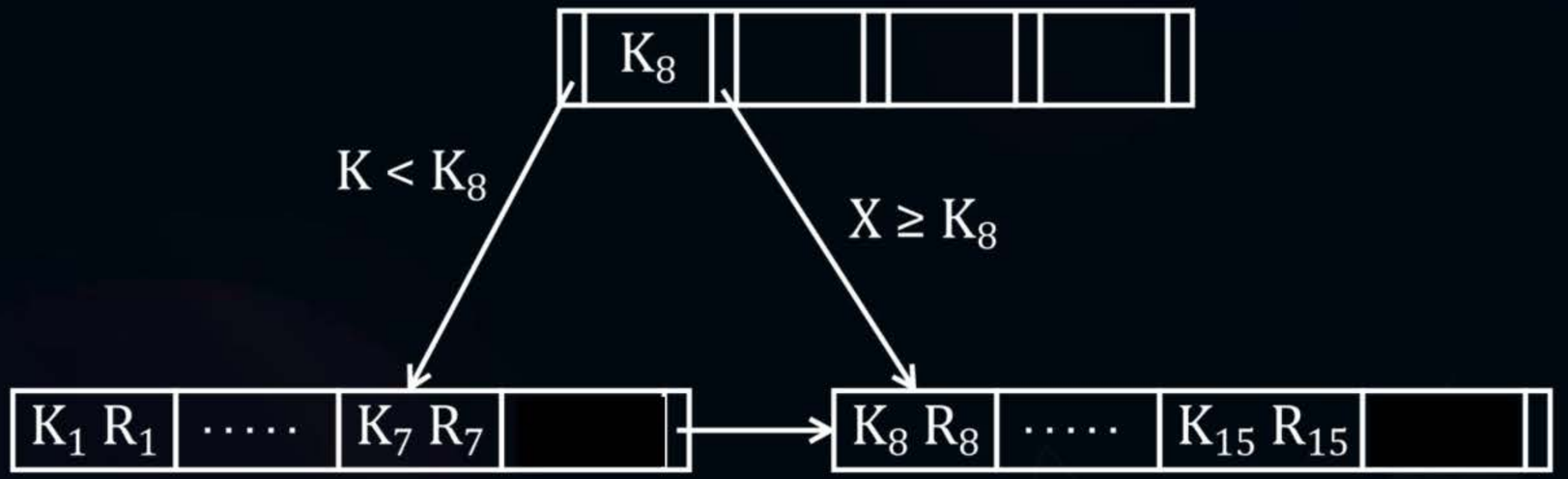
$$K_1 < K_2 < \ldots < K_8 < \ldots < K_{15}$$

Topic: Left Biasing



- Observe the Parent key K_8 is part of the left sub tree, making it Left Biased.

Topic: Right Biasing



- Primary key present in Right subtree, making it Right Biased.

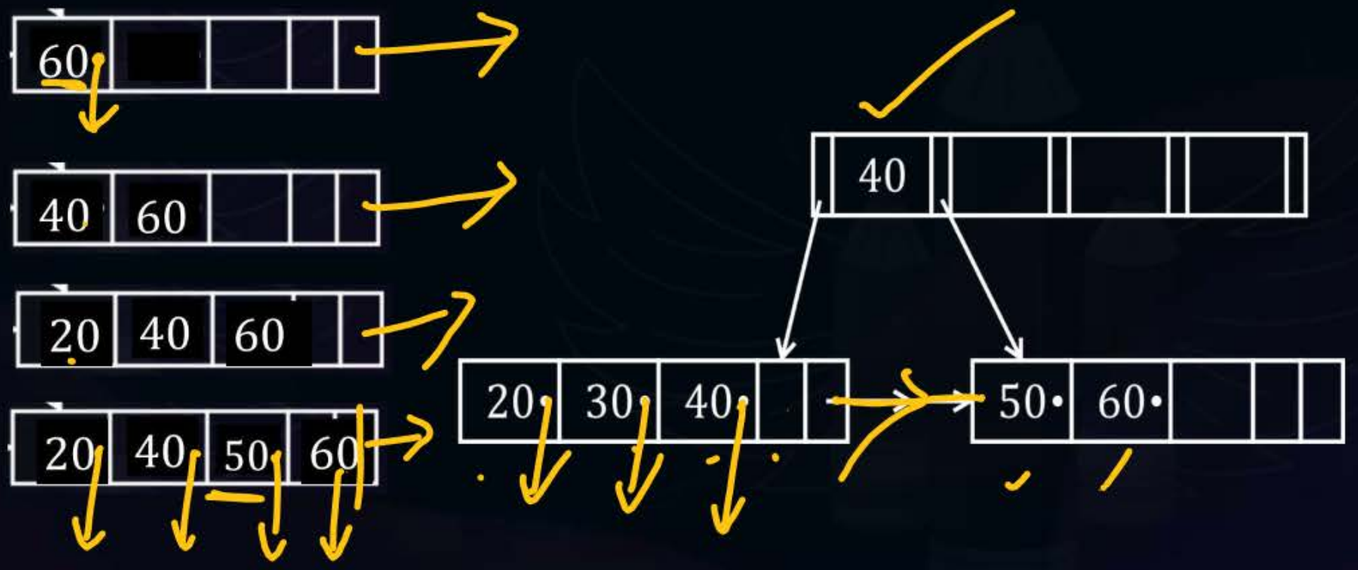
Topic: B+ Tree

Construct B+ Tree with Order P : 5 [Max Pointers per node]

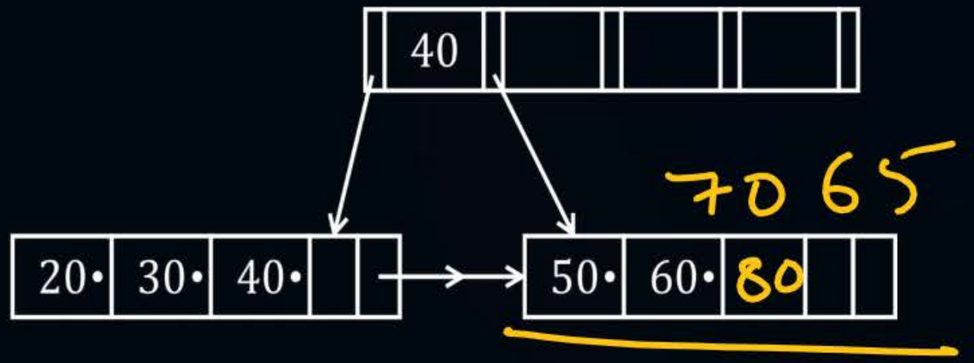
Sequence of keys : 60, 40, 20, 50, 30

20 30 40 50 60

These are data pointers,
will always be present



Topic: B+ Tree



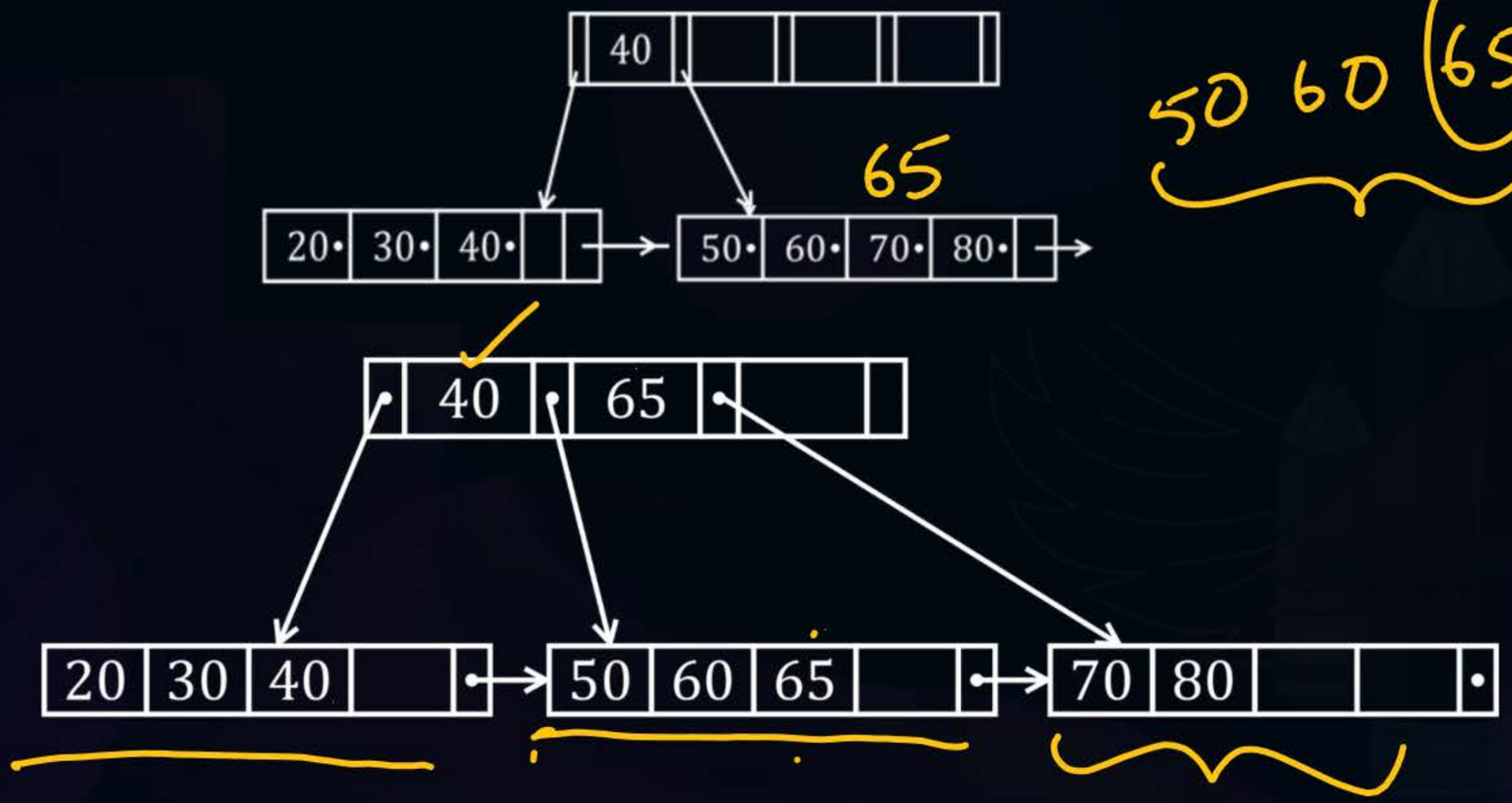
Further add 80, 70, 65

adding 80

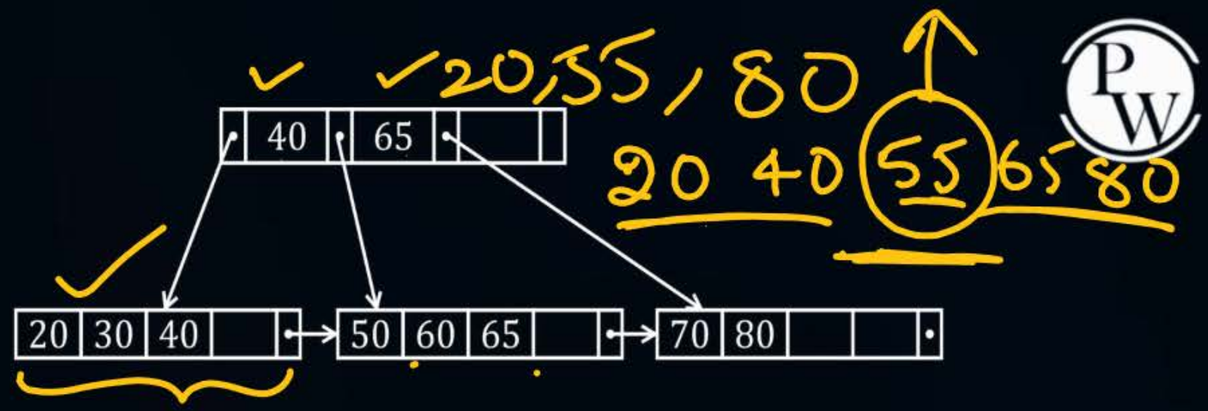
adding 70

Topic: B+ Tree

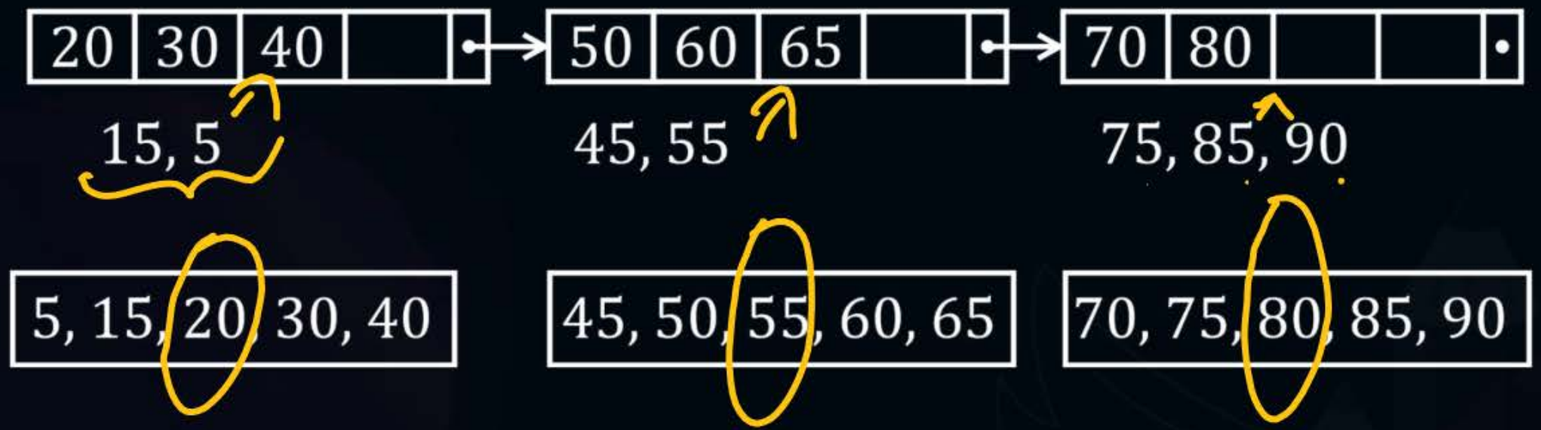
→ To add 65, we will need to split the tree nodes.



Topic: B+ Tree



- Further add 15, 5, 45, 55, 75, 85, 90



Topic: B+ Tree

→ Note that atmost we will be able to store 4 keys per node. So when we split leaf node, 20, 55 and 80 goes to the root internal node level.

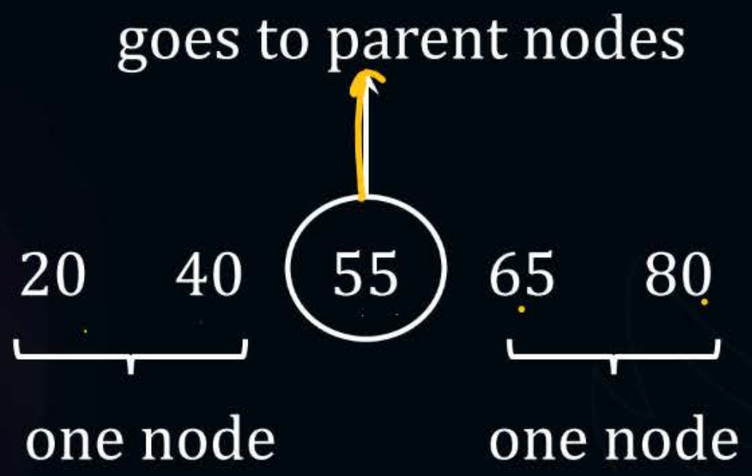
- At the internal node level, we already have 

now along with them, we will accommodate 20, 55 and 80.

20 40 55 65 80

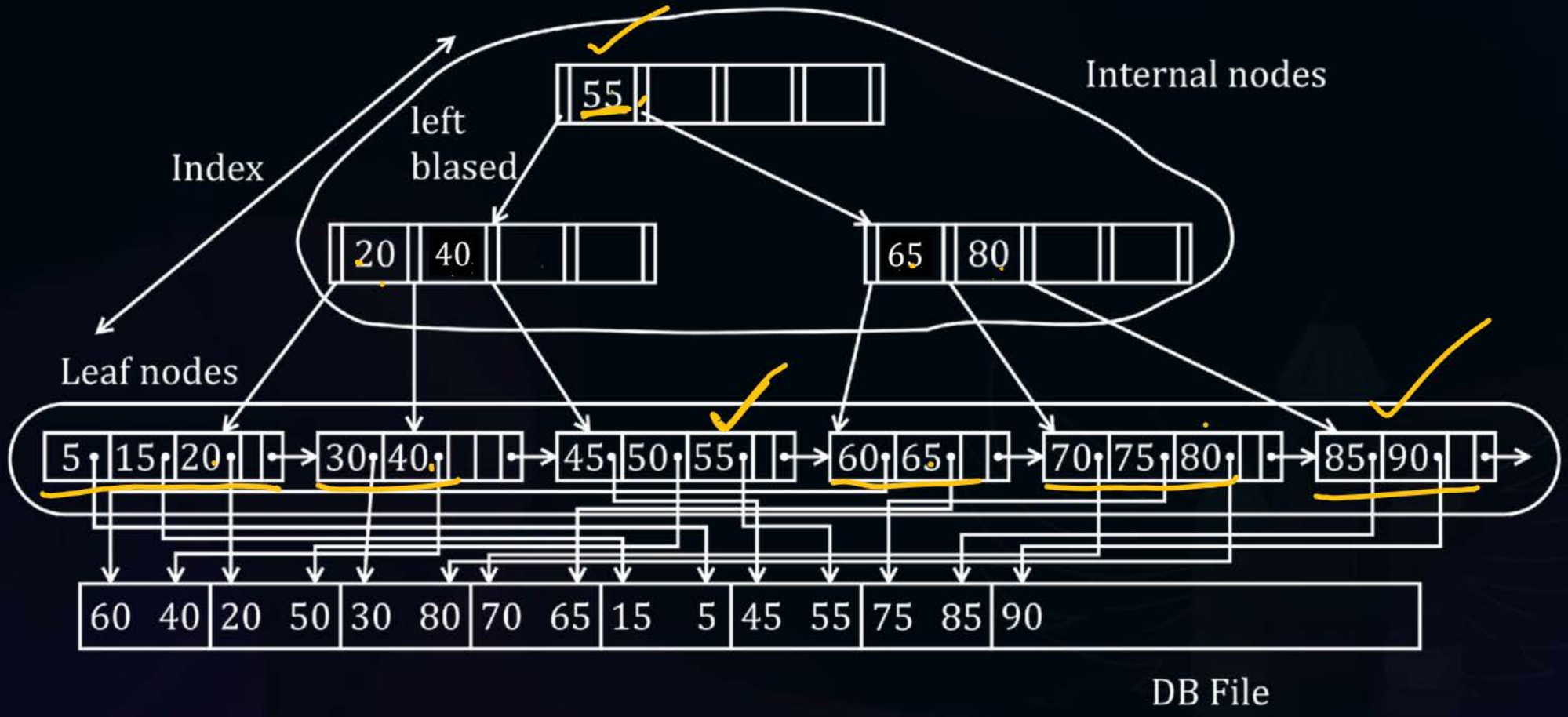
Topic: B+ Tree

→ When we are splitting leaf node, middle key should be in one of the leaf node. But when we are splitting internal node, middle key goes to parent, keys less than middle key becomes one node, keys more than middle key becomes another node.



- Leaf node splitting : All keys remain in leaf level.
- internal node splitting : Middle key move up.

Topic: B+ Tree


Internal nodes
Index
left
blased
55
20
40
65
80
Leaf nodes
5
15
20
30
40
45
50
55
60
65
70
75
80
85
90
60
40
20
50
30
80
70
65
15
5
45
55
75
85
90
DB File

Inspiring Stories : Bajau Nomads

Background: **People who live mostly on the sea.**

Struggles: **Depend fully on ocean for food and life.**

Achievements: **Can dive 60m deep, hold breath for minutes.**

Impact: **Prove human body can adapt to sea life.**

Topic: B^+ Tree

B^+ Tree index is best suitable for both:

1. Random access of any one record.
2. Sequential access of a range of record.

Topic: Random Access of Any One Record

- To access a record, we will have to pass through K levels of Index and then 1 for accessing from DB file.

→ SELECT * FROM R

WHERE A = 55;

→ Worst Case : ~~(K + n(1)+ number of leaf nodes accessed) blocks~~

(K+1) ~~where n is the number of keys in the range~~

Topic: Sequential Access of a Range of Record

- Unlike B Tree, we have block pointers in leaf nodes, which makes it much easier to do sequential access since we can directly access adjacent leaf nodes.

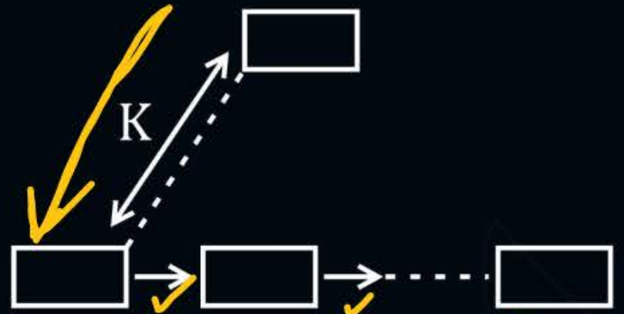
Topic: Sequential Access of a Range of Record

→ SELECT * FROM R

WHERE A ≥ 60 and A ≤ 75

(60, 75)

- Say the range has X element



X(K+1)

⇒ (K + 1 + number of leaf nodes accessed + X(1)) blocks

where X is the number of keys in the range

Accessing element becomes easier from block pointers

Topic: Observation

- Number of Internal node keys = Number of leaf nodes – 1
- Complete Tree must be either Left Biasing or Right Biasing.

Topic: Problems

Find best possible order of B/B$^+$ Tree Node

Questions:

Block Size: 1024 bytes

Search Keys: 10 bytes

Block pointer: 8 bytes

Record pointer: 9 bytes

[Order p: Max possible block pointers which can be stored in B Tree node]

Topic: Problems

Block Size: 1024 bytes
Search Keys: 10 bytes
Block pointer: 8 bytes
Record pointer: 9 bytes

#Q. What is best possible order of B-Tree Node?

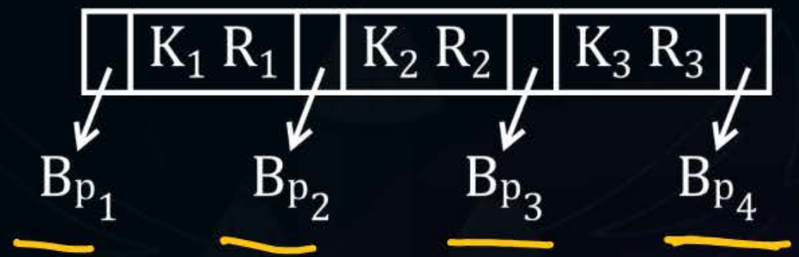
Sol. One disk block in secondary memory is allocated for BTree node.

It's already given that node can store p block pointers at max.

Block pntr (B_P) → p

key + records pntr → p –1

$P* Bp + (P -1) [K + R_P]$ Block (B tree node)



$P * Bp + (p - 1) [k + Rp] \leq$ block size

$p * 8 + (p - 1) [10 + 9] \leq 1024$

$27p \leq 1043$

$p = \left\lfloor \frac{1043}{27} \right\rfloor = 38$ Node can store atmost 38 block pointers and 37 keys.

Topic: Problems

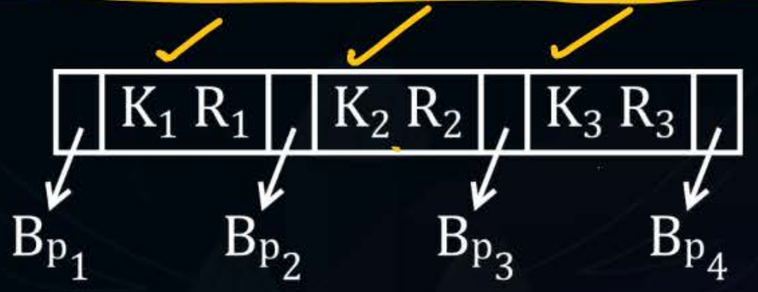
#Q. B = 512 bytes [Block size]

K = 5 bytes [Search size]

Bp = Rp = 10 bytes

order p: let p be the maximum possible keys. That can be stored in a B-Tree node.

What is best possible order of B-tree node?



Sol: $(p+1)\,bp + p\,(k + Rp)$ B-tree node Block

$(P + 1)\,10 + P\,[5 + 10] \leq 512$

$25\,p \leq 502$

$p = \left\lfloor \frac{502}{25} \right\rfloor = 20$

20 is max keys possible in b tree node.

Topic: Problems

#Q. Block Size: 1024 bytes

Bp : 8 bytes [Block pointer]

Rp : 9 bytes [Record pointer]

Key : 10 bytes

Order p: max possible pointers which can be stored in B^+ Tree node.

What is the possible best possible order:

i. In Internal node of B^+ Tree.

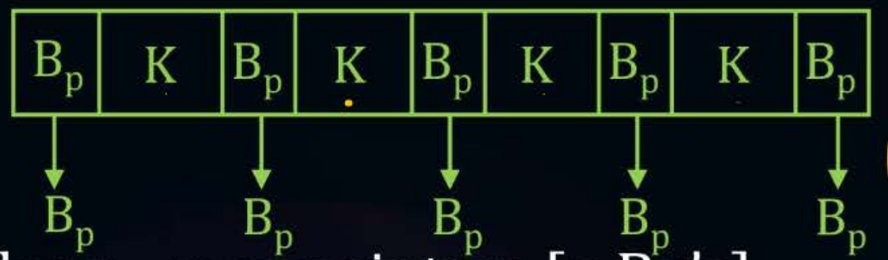
ii. In leaf node of B^+ Tree.

Topic: Problems

Block Size: 1024 bytes
Bp : 8 bytes [Block pointer]
Rp : 9 bytes [Record pointer]
Key : 10 bytes
Order p: max possible pointers which can be stored in B+ Tree node.

Sol:

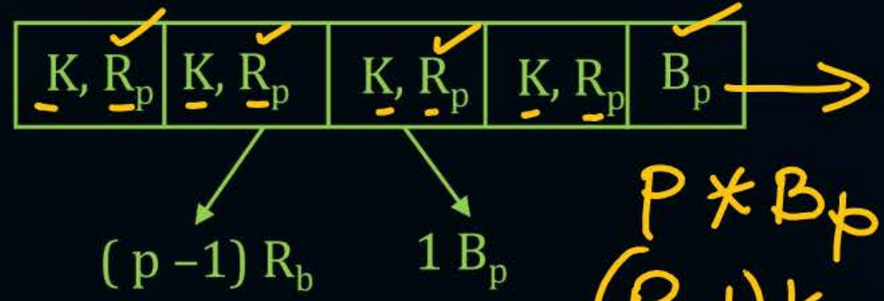
Internal Node:



P*Bp
(P-1)k

Order p : max pointers [p Bp's]

Leaf Node:



P * Bp
(P-1)k

p * Bp + (p –1) keys — Block (B+ Tree Node)

p * Bp + (p – 1) keys ≤ Block size
p * 8 + (p – 1) 10 ≤ 1024
18p ≤ 1034 p ≤ 1034

$$p = \left\lfloor \frac{1034}{18} \right\rfloor = 57 \text{ max Bp}$$

(P –1) [K + Rp] + Bp — Block

(P –1) [K + Rp] + Bp ≤ Block size
(P –1) [10 + 9] + 8 ≤ 1024
19 p ≤ 1035

$$p = \left\lfloor \frac{1035}{19} \right\rfloor = 54$$

p = 54 max Bp — Pointer

Topic: Problems

Note:

If size of Bp = size of Rp

then

(B^+ Tree internal node order) = (B^+ tree leaf node order)

Topic: Problems

#Q. Block: 512 bytes K (Search Key): 5 bytes

$B_b = R_p$

$= 10$ bytes

Order p: max possible keys which can store in B^+ Tree node.

What is best possible order of B^+ Tree node?

Sol: The answer will be the same for Internal node and Leaf node since we have Same Block pointer Size and Record pointer size.

⇒ Order p: max Keys per node.

⇒ Internal Node: (p+1) pointers

Topic: Problems

$(p+1)\,Bp + p^{*}\,keys \le block$

$(p+1)\,10 + p * 5 \le 512$

$15p \le 512$

$p = \left\lfloor \frac{502}{15} \right\rfloor = 33$

$p = 33$

The result will be same for leaf node:

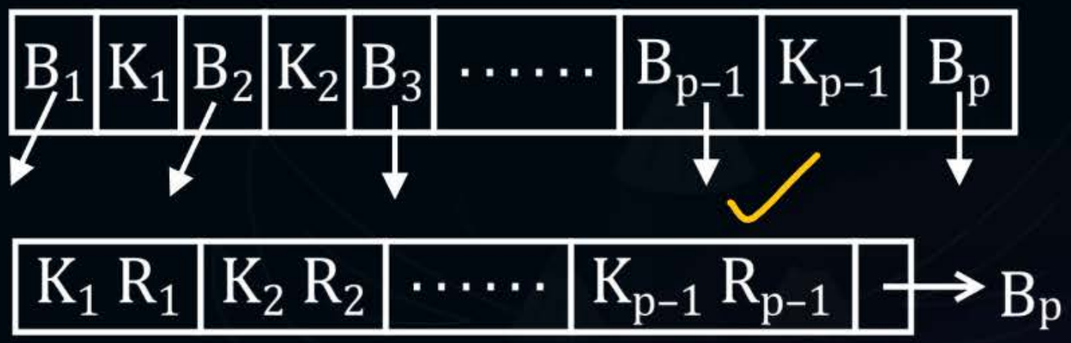
$P[\,k + Rp] + Bp \le$ Block size

Block: 512 bytes K (Search Key): 5 bytes

$B_b = R_p$

$= 10$ bytes

Order p: max possible keys which can store in B^+ Tree node.



Topic: Problems

What can me conclude?

(Block size) B : 1024 bytes

(~~Block~~ size) K : 10 bytes

Key

Rp : 9 bytes (Records pntr)

Bp : 8 bytes (Block pntr)

prepared for DB Index

B Tree Node
order p : 38
max Bp

B+ Tree Node
Internal Node order : 57
Leaf node order : 54

⇒ We can observe that order of B^+ Tree node is much higher then order of B Tree node for block of same size.

⇒ Implying that one B+ Tree node can accommodate much more number of keys then one B Tree node of same size block.

Topic: Problems

⇒ In Short:

If (Block Size of B Tree node = Block size of B+ Tree node then:

1. (order p of B tree node) < (order p of B^+ tree node)
2. (no. of B tree node for n distinct keys)> (No. of B^+ tree node for n distinct keys)

I/O cost

3. (No. of levels of B tree index for n distinct keys) > (No. of levels of B^+ Tree index for n distinct keys)

I/O cost

* Another type of Question:

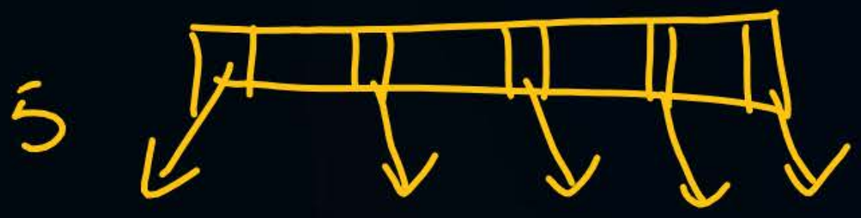
Find minimums/maximum keys in B/B+ Tree for given levels :-

Topic: Problems

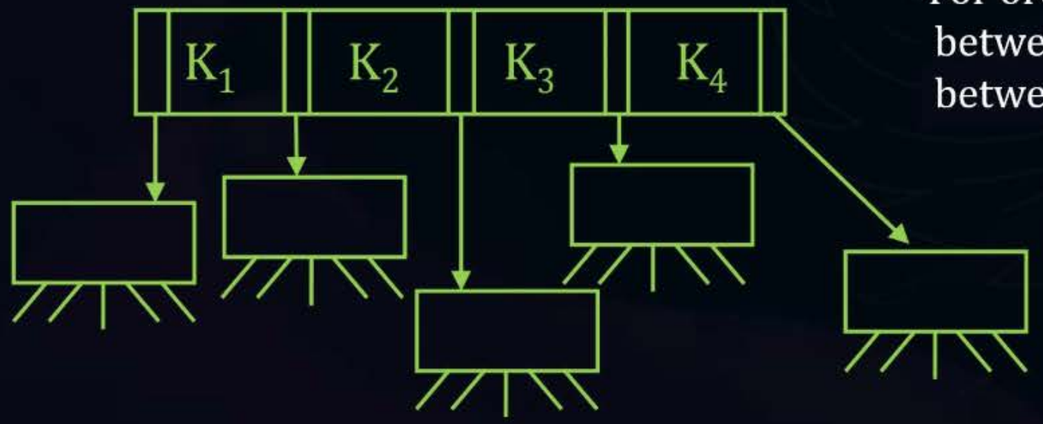
#Q. How many min/max keys and Nodes present in B Tree / B+ Tree with order p: 5 and level 4

For order p:
between 2 to p BP's for Root Node.
between $\lceil p/2 \rceil$ to p BP's for other nodes

Topic: Problems



Level	Max nodes	Max (Block pntr) max Bp's	Max Keys
1	1	5	4
2	5	5×5	5×4
3	25	25×5	25×4
Leaf 4	125	—	125 × 4



For order p:
between 2 to p BP's for Root Node.
between $\lceil p/2 \rceil$ to p BP's for other nodes.

Topic: Problems

Level	Max nodes	Max (Block pntr) max Bp's	Max Keys
1	1	5	4
2	5	5×5	5×4
3	25	25×5	25×4
Leaf 4	125	-	125 × 4

⇒ Max Nodes in B/B+ Tree with order P:5 and level 4 is 156 125 + 25 + 5 + 1

⇒ Max keys in B Tree with order P=5 and level 4 is 624 = 125 * 4 + 25 * 4 + 5 * 4 + 4

⇒ Max keys in B+ Tree with order P=5 and level 4 is 500 125 * 4

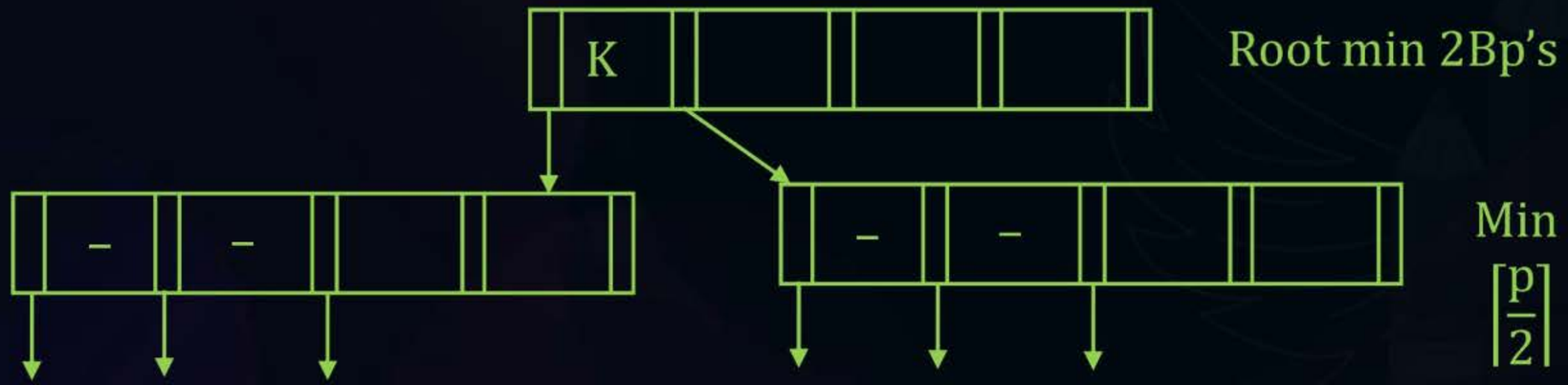
⇒ We will only consider the keys present in last, that is, leaf level because all the distinct keys in case of B+ Tree are present in last level only.

Topic: Problems

$p = 5 \quad \lceil \frac{5}{2} \rceil = 3$ BP
2 keys

⇒ Finding Minimum Keys

⇒ Root must have atleast 1 key and 2 BP's, other internal nodes must have atleast $\lceil \frac{p}{2} \rceil$ Block Pointers (Bp's)



Topic: Problems

Level	Min nodes	Min Bp's	Max Keys
Root 1	1	2	1
2	2	2×3	2×2
3	6	6×3	6×2
Leaf 4	18	-	18 × 02

$$\left\lceil \frac{5}{2} \right\rceil = 3$$

Topic: Problems

Level	Min nodes	Min Bp's	Max Keys
Root 1	1	2	1
2	2	2×3	2×2
3	6	6×3	6×2
Leaf 4	18	-	18 × 2

⇒ Min Nodes in B/B$^+$ Tree with order P: 5 and 4 levels = 1 + 2 + 6 + 18 = 27

⇒ Min Keys in B Tree with order P: 5 and 4 levels = 18*2+6*2+2*2+1 = 53

⇒ Min Keys in B$^+$ Tree with order P: 5 and 4 levels = 18*2 = 36

∵ This is because we should take keys of last level only.

Topic: Problems

#Q. How many Minimum/Maximum Keys & BP's are there in B/B$^+$ Tree with Order P: 5 & levels: 4

Assume Order P: b/w 1 to 2p Keys for Root. — 1 to 10 keys | 2 to 11 Bp

b/w p to 2p keys for other Nodes 5 to 10 key | 6 to 11 BP

(This is the definition of order in this question, definition can change from question to question).

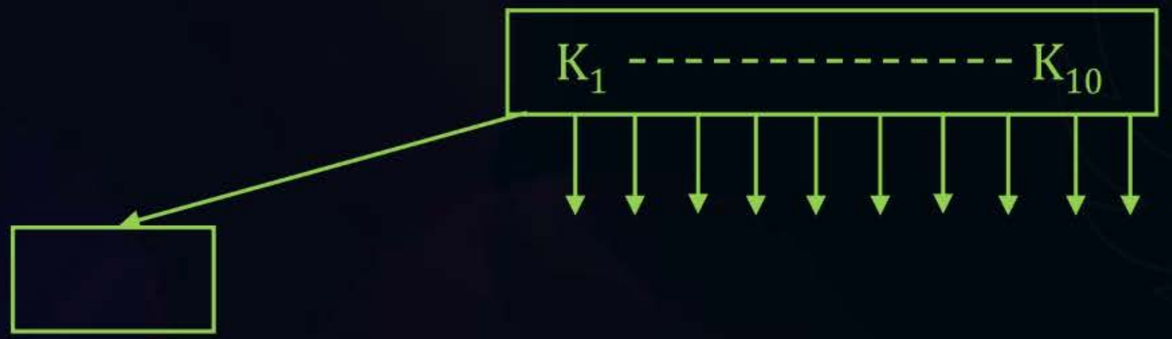
Sol: Maximum Number of keys in any node is 2p = 10 ✓

⇒ If there are 10 keys, number of blocks pointer will be 1 more then that, block pointers

⇒ 10 keys ⇒ (10 + 1) Bp

Topic: Problems

Level	Nodes	Max Bp's	Max Keys
Root 1	1	11	10
2	11	11×11	11×10
3	121	121×11	121×10
Leaf 4	1331	-	1331 × 10



Topic: Problems

Level	Max nodes	Max Bp's	Max Keys
Root 1	1	11	10
2	11	11×11	11×10
3	121	121×11	121×10
Leaf 4	1331	-	1331 × 10

⇒ Max Nodes in B/B$^+$ Tree with order p: 5and level 4 is 1464 (1331+121+11+1)

∵ We add nodes of all level from root to leaf

⇒ Max Keys in B Tree with order p: 5 and level 4 is 14640 (13310+1210+110+10)

∵ we add Keys across all levels from root to leaf."

⇒ Max keys in B$^+$ tree with order p: 2 and level 4: 1331 × 10 = 13310

Topic: Problems

#Q. How many Minimum/Maximum Keys & BP's are there in B/B$^+$ Tree with Order P: 5 & levels: 4

Assume Order P: b/w 1 to 2p Keys for Root.
b/w p to 2p keys for other Nodes

(This is the definition of order in this question, definition can change from question to question).

Sol Cont: Minimum Number of keys in internal node is p = 5

Minimum number of keys root is 1

⇒ For internal node, there are 5 keys, number of blocks pointer will be 1 more then that, block pointers

⇒ 5 keys ⇒ (5 + 1) Bp

Topic: Problems

Level	Nodes	Min Bp's	Min Keys
Root 1	1	2	1
2	2	2×6	2×5
3	12	12×6	12×5
Leaf 4	72	-	72 × 5

Topic: Problems

Level	Nodes	Min Bp's	Min Keys
Root 1	1	2	1
2	2	2×6	2×5
3	12	12×6	12×5
Leaf 4	72	-	72 × 5

"From question, we are given that minimum number of keys for:
Root → 1 Other nodes → b = 5

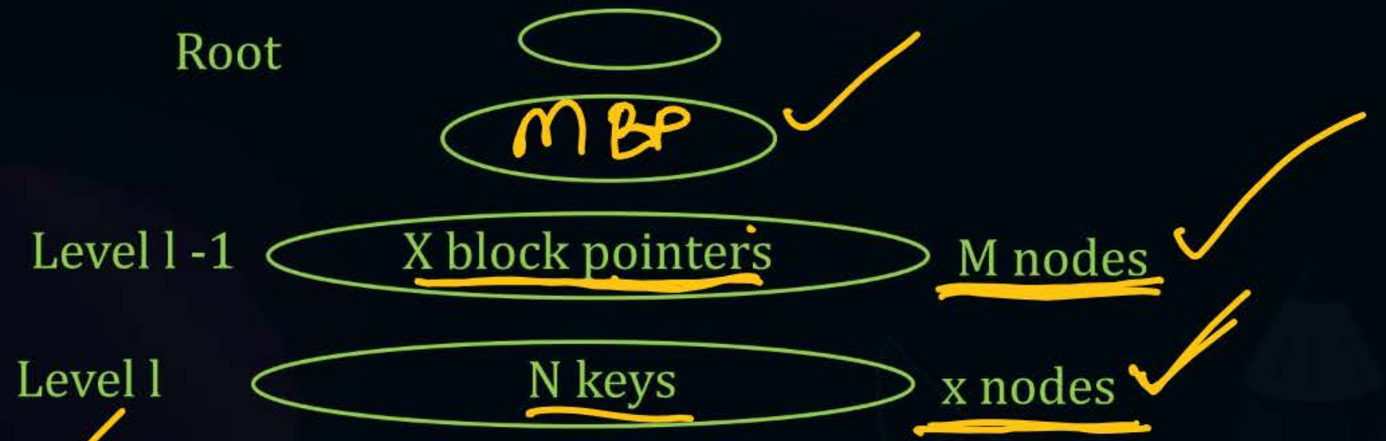
⇒ Min Nodes in B/B$^+$ Tree with order p:5 and 4 levels: 72 + 12 + 2 + 1= 87

⇒ Min Keys in B Trees with order P: 5 and 4 level : 72 * 5 + 12 * 5 + 2 * 5 + 1 = 431

⇒ Min Keys in B$^+$ Tree with order p: 5 and 4 level: 72 * 5 = 360
(we only considers leaf level Keys)

Topic: Bulk Loading B+ Tree

⇒ Construction of B^+ tree in bottom up approach, that is leaf to root node.



Design Leaf Nodes:

⇒ Distribute Keys into Leaf Nodes

⇒ Result: # number of Leaf nodes

Topic: Bulk Loading B^+ Tree

Design Internal Nodes:

If x nodes at level l then x block pntrs at level l–1

→ Distribute block pointers into nodes

→ Result: # of nodes at level l–1

Repeat until we reach root, that is one node.

Points to remember:

→ Based on Keys, we design Leaf Nodes

→ Based on Block Pointers in level l+1, we need to design the Internal Node at level l

Topic: Bulk Loading B+ Tree

#Q. Construct Bulk Loading B+ Tree for:
Keys [1, 2, 3, ..., 24, 25] with order b: 5 (Max Pntrs)

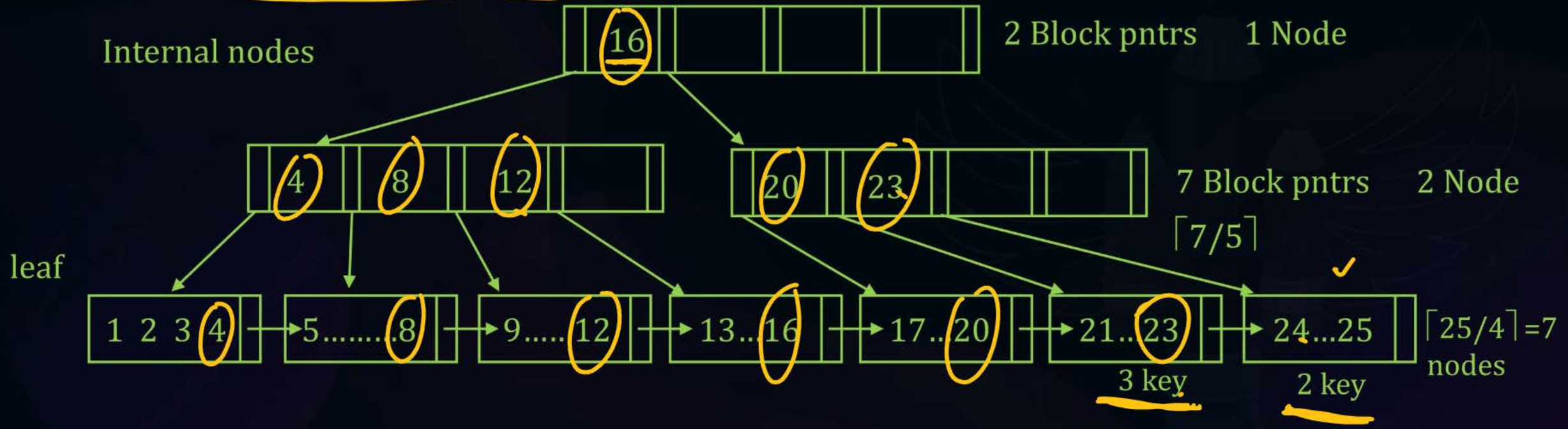
$\lceil \frac{5}{2} \rceil = 3$ ✓

i. How many minimum levels?

⇒ If we want minimum levels, we need maximum keys per node

⇒ max: 4 keys per node, 5 pntrs per node.

⇒ Minimum required level is 3.



Topic: Bulk Loading B+ Tree

P = 5 min pointer = $\lceil \frac{P}{2} \rceil$ = 3

min key = 2

⇒ we need to have atleast 2 Keys in internal node and 3 block pointers.

How many max levels?

Topic: Bulk Loading B^+ Tree

#Q. Numbers of keys used for Index: 20,000 order P: 20 (max pointers per node)
How many min levels/min Nodes are there in B^+ TREE?

Sol: Minimum Levels:

⇒ order given is 20 for pointers per node

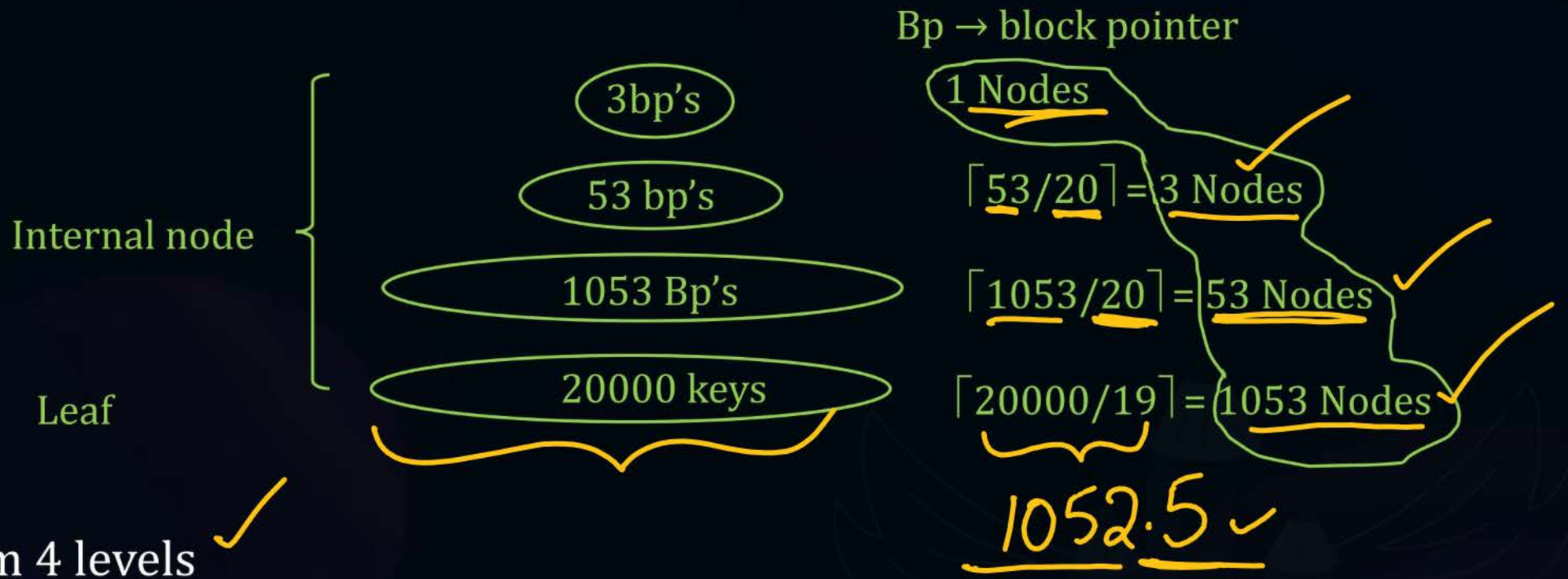
⇒ Max pntrs → 20/node Max keys → 19/node

⇒ Number of nodes required for key in leaf level = $\lceil$ Total keys/19 $\rceil$

⇒ Number of pointers in level l = number of nodes in level l +1 (assume p)

⇒ Number of Nodes required at level l = $\lceil$ p/20 $\rceil$

Topic: Bulk Loading B+ Tree



⇒ Minimum 4 levels

Note:

Finding minimum level

⇒ use ⌈ ⌉ ceil to find number of nodes

Finding maximum level

⇒ use ⌊ ⌋ floor to find number of nodes.

Topic: Bulk Loading B^+ Tree

Note:

Finding minimum level

$\Rightarrow$ use $\lceil\ \rceil$ ceil to find number of nodes

Finding maximum level

$\Rightarrow$ use $\lfloor\ \rfloor$ floor to find number of nodes.

Topic: Bulk Loading B+ Tree

#Q. How many maximum levels/max nodes are possible for B+ tree.

⇒ Max level : Minimum keys per nodes

min will be $\lceil p/2 \rceil = \lceil 20/2 \rceil$ 10 pointers

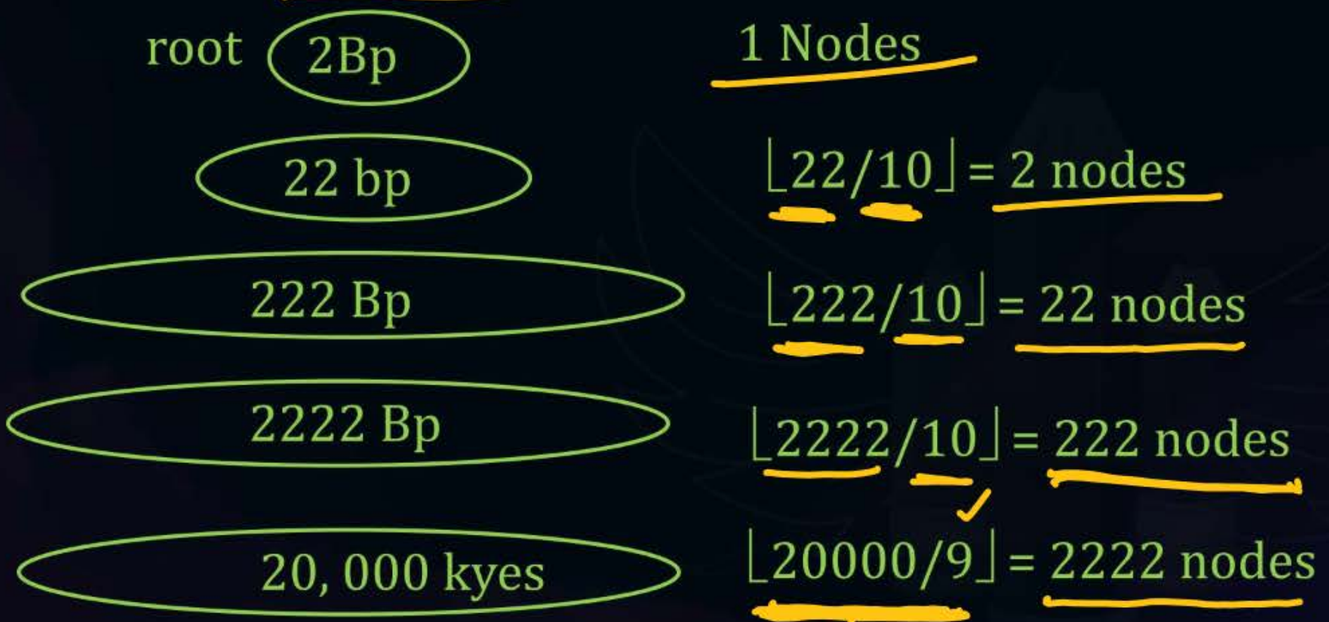
= 9 keys

Maximum nodes

= 2222 + 222 + 22 +2 + 1

= 2469 Nodes

⇒Max levels → 5



Topic: Bulk Loading B^+ Tree

Type of Numerical we can get from B/B^+ Tree:

1. Best possible Order of B/B+ Tree node?
2. If levels are given to us and we need to find min/max keys of B/B+ Tree.
   → We use Top-down Approach
   → Root to Leaf
3. If number of keys are given to us and we need to find min/max levels of B/B+ Tree.
   → We use Bottom - Up Approach (BULK-LOADING)
   → Leaf to Root
4. If number of keys are given to us and we need to find min/max levels of B Tree.
   → We use Top-down Approach
   → Root - to - Leaf

Inspiring Stories : Cholan. Tribe

Background: **Tiny forest tribe in Kerala, <200 people.**

Struggles: **Live cut off from modern world, little contact.**

Achievements: **Use stars, sky, and forest as calendar and guide.**

Impact: **Keep ancient knowledge alive without schools.**

Join

@RBRSIRPW

THANK - YOU