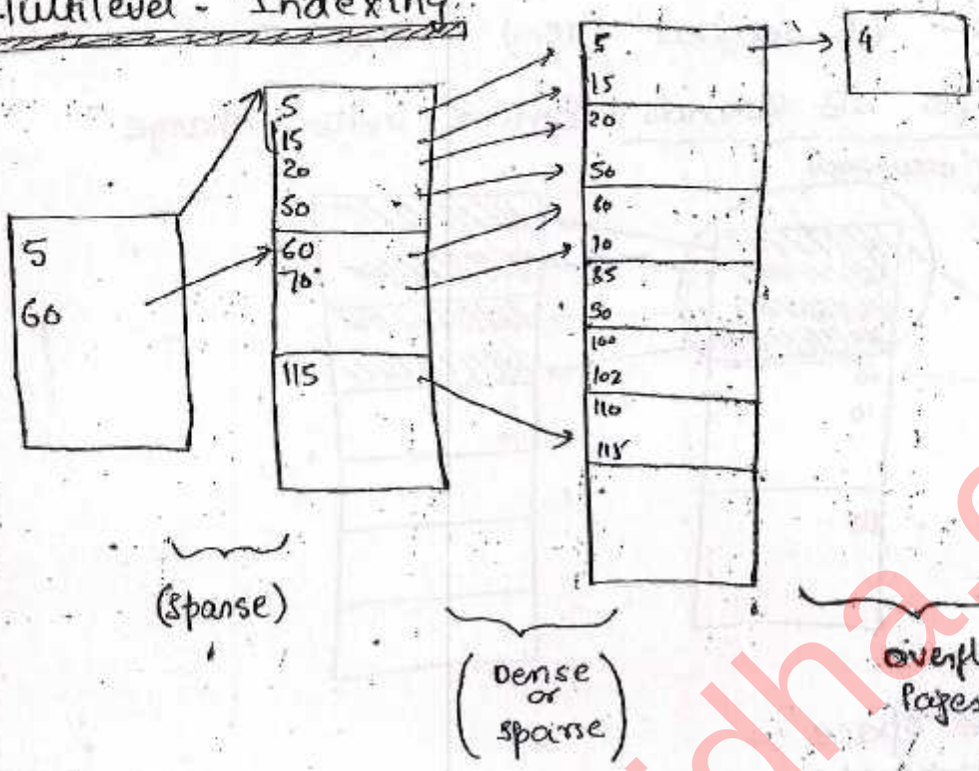
1st Nov 2011

Multilevel - Indexing



PROBLEM

⇒ Insert record with key 4 — can't be inserted, now as insertion occurs in order

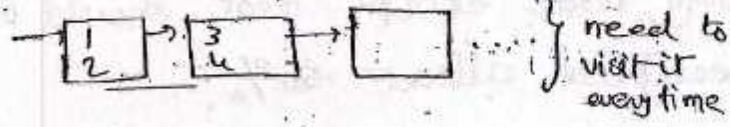
↳ Possibilities (Insertion)

① Shifting of DB — results entire index change, DB file change.

② Overflow pages :-

Adv:- No need to change index file

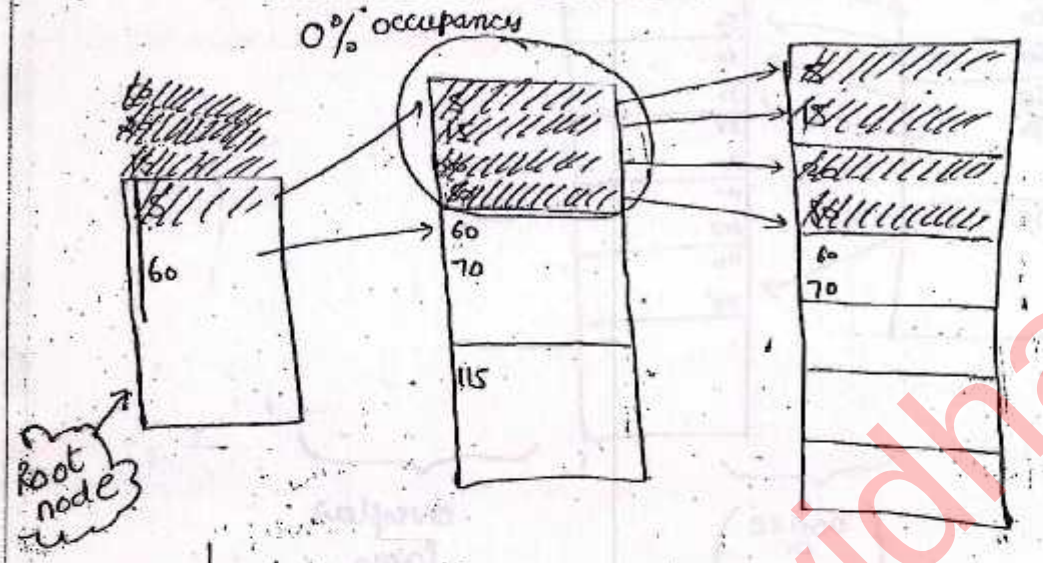
Disadv:- If overflow pages are huge, then it results more I/O cost.



Disadv:-

Deletion:- (5 deleted then)

(I) Shifts DB Records: Entire index change



↳ Leave space :-

Consecutive deletions results 0% occupancy for any Index table.

Dynamic Multilevel Indexing:-

Goals:- ↳ To overcome limitations of normal multilevel Indexing.

(I) Insertion / Deletion, it should not result in entire index change.

(II) No concept of overflow pages

(III) Every index node except root should be occupied atleast 50%.

Standard Dynamic Multilevel Index are:-

(I) B Tree
(II) B^+ Tree

Balanced Search Tree (Height Restricted Trees)

B-TREE Index

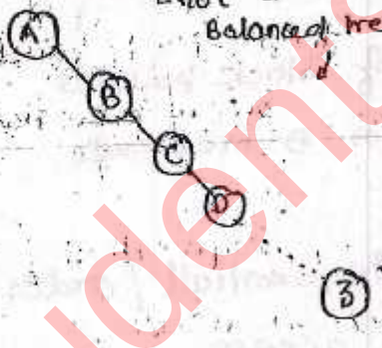
B - Balanced Search tree.

» The height of B-tree should not be more than $O(\log N)$.

» Examples of B-Tree
- » AVL Tree
- » B/B^+ Tree
- » 2-3 Tree

$\text{Height} \leq O(\log_2 N))$

» Binary search tree
└ not a Balanced tree



⇒ Time complexity of searching in BST

$O(\text{height of tree})$

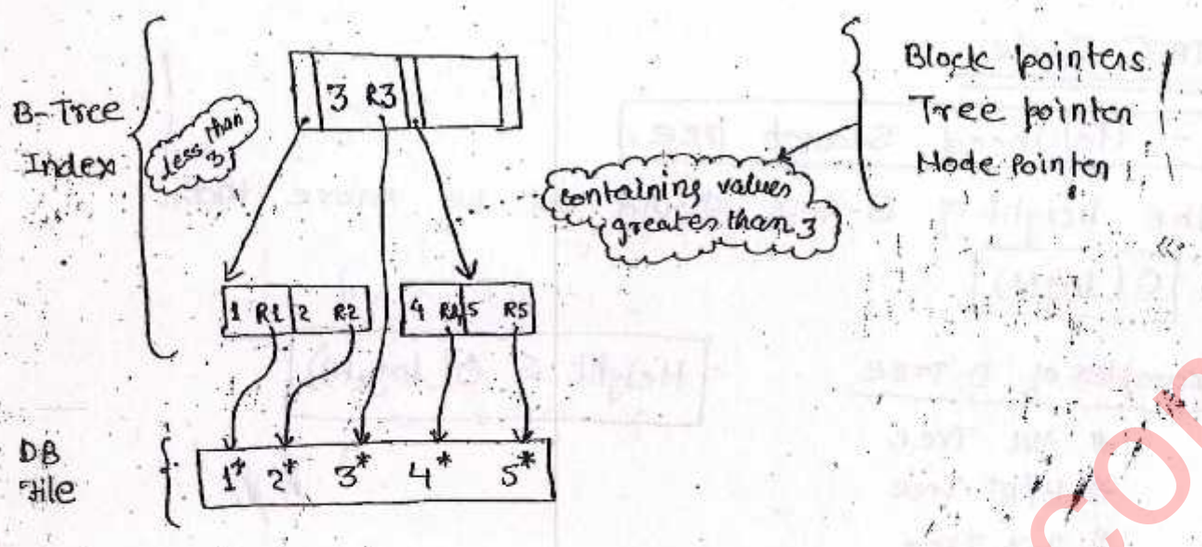
└ mini height = $\log N$
max. height = N.

$O(\log n)$

worst case
$O(N)$

Dynamic Multilevel Indexing (B/B+ Tree)

Leaf Key of B-tree consist of Key + Record pointer (data pointer / value pointer)

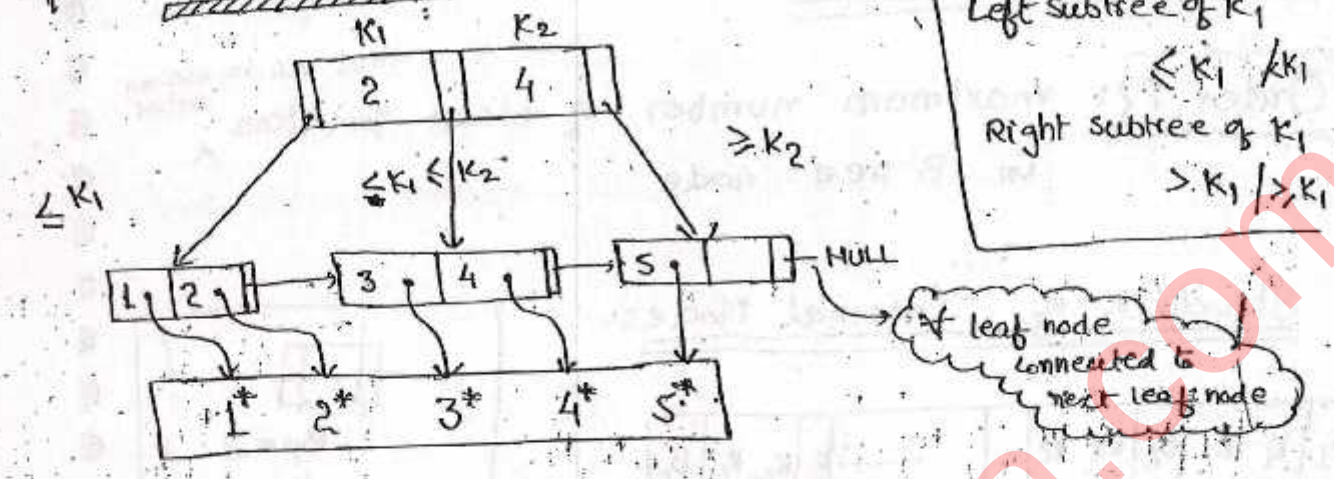


⇒ Advantages :- Best for Random Record Access

$O(\log_p H)$ p : # of block pointers in B-Tree node.

⇒ Disadvantage :- Not best for Sequential / ordered record access.

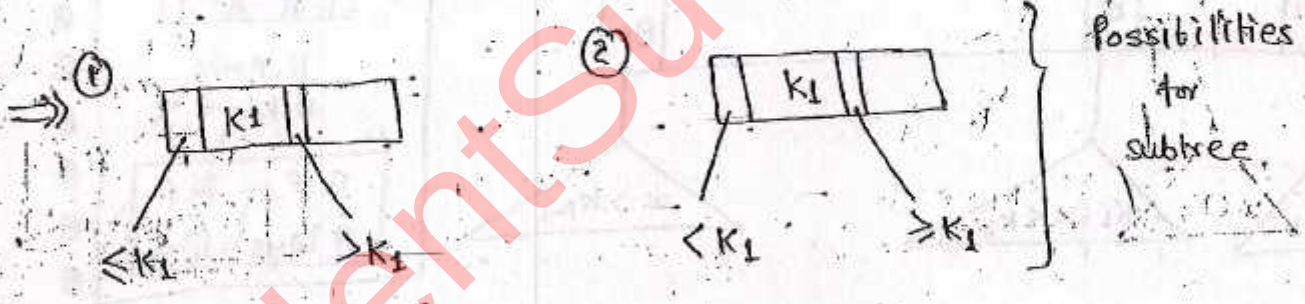
B^+ Tree



Note:-

(1) Every key should be present in the leaf node.

(2) Every leaf node consists of one block pointer which points next leaf node.

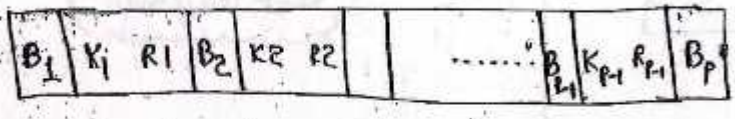


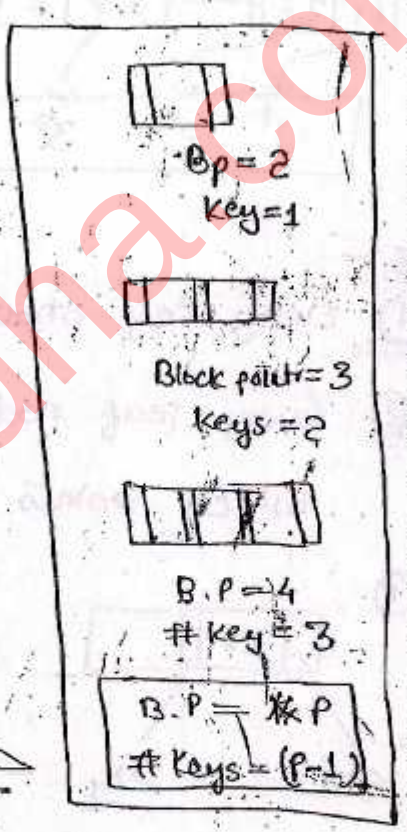
⇒ Gives better performance for both Random Access and Sequential Access.

Definition of B-Tree

» **Order p**: maximum number of block pointers that can be accommodated in B tree node

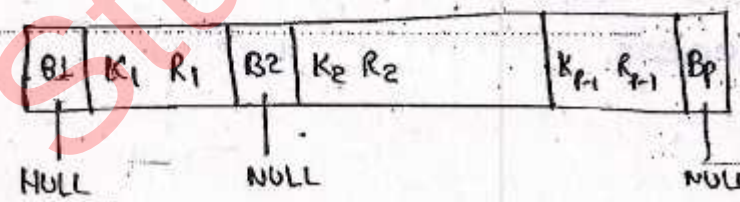
(1) Structure of Internal Node:-





if p block pointers then (p-1) keys and (p-1) ptr should also exist.

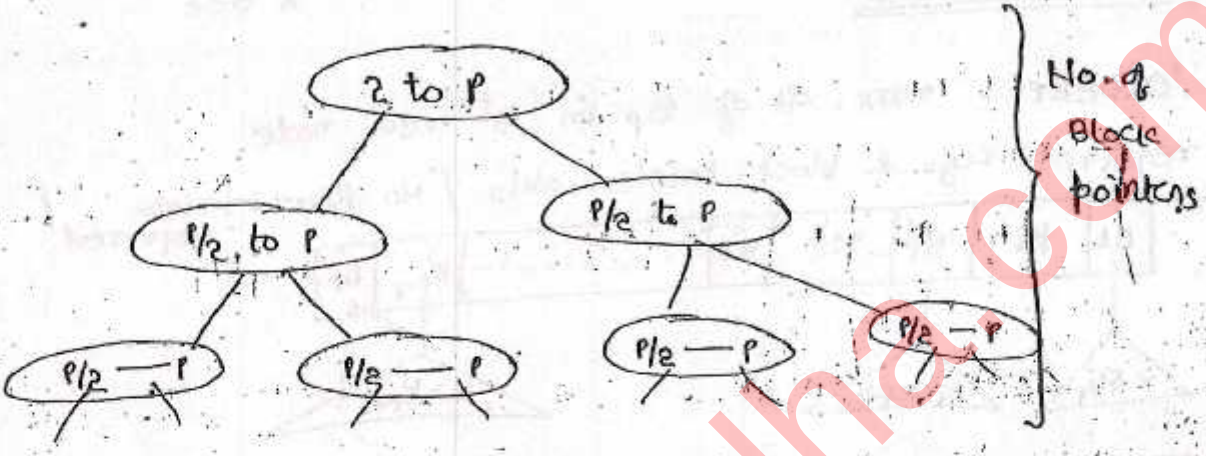
(2) Structure of Leaf Node



(3) Every Internal node except root should be atleast $\lceil p/2 \rceil$ block pointer and atmost p Block pointers.

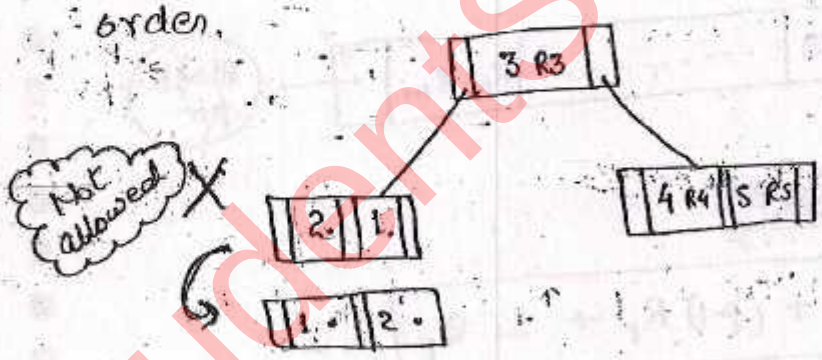
↓ minimum ↓ maximum

④ Root can be (atleast) two block pointers and (atmost) p block pointers.
(root node may not always 50% occupied)



⑤ Every leaf node should be at same level

⑥ Key's within node should be in Ascending order.



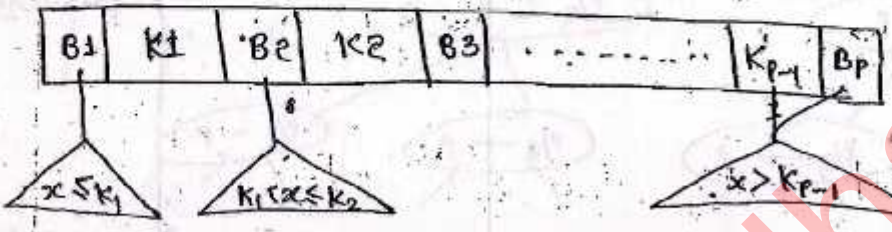
B^+ Tree Definition :-

└ more standard data structure compared to B tree.

(I) Internal Nodes:

'ORDER$_P$' : max. # of B_p in B^+ tree node.

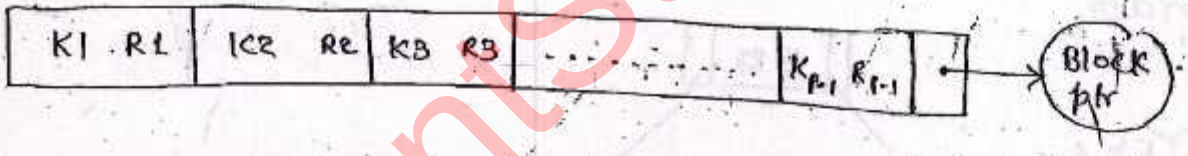
- requires keys & block pointers only / No Record pointers required



or

$x < K_1$ | $K_1 \leq x < K_2$ | $x \geq K_{p-1}$

(II) Structure of Leaf Node:



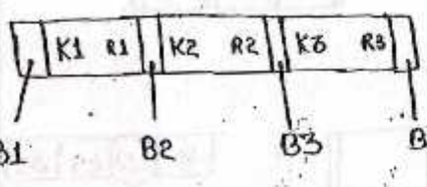
Capacity of B^+ tree:

$$= (p-1) \text{ keys} + (p-1) R_p + 1 B_p$$

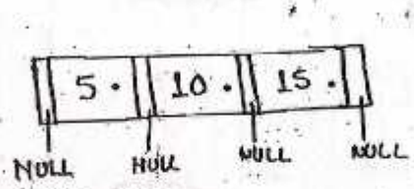
(III) (IV) (V) (VI) Same as B tree

Q:- Consider B-tree, with ORDER P=4, &
Sequence Keys = 5, 10, 15, 1, 23, 34, 13, 8, 18, 37, 19, 40

Sol^{n} :- ORDER P = 4.



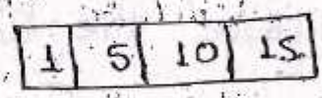
↳ Thus one node can accomodate atmost 3 keys!

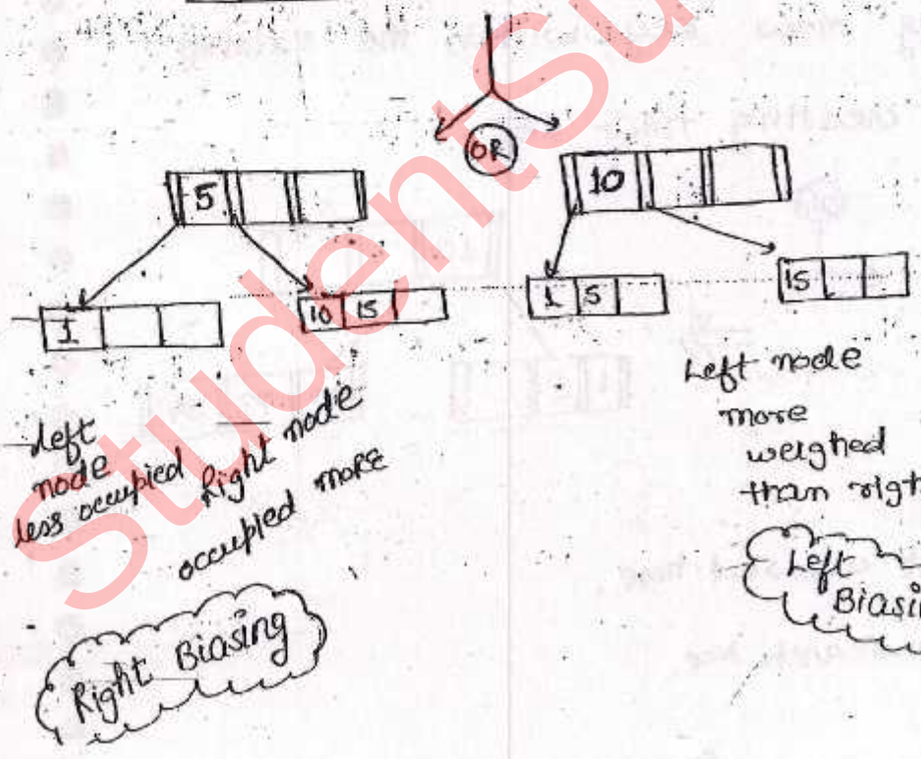


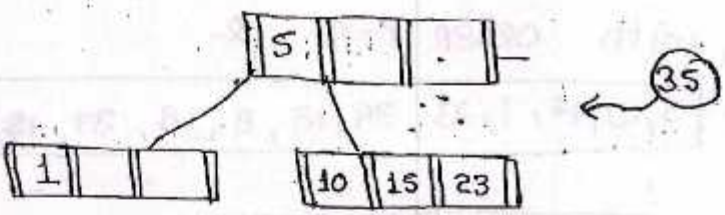
- Since capacity full so to insert (1) we go for

Node splitting

↳ write all keys including current one in ascending order and take middle one as root.

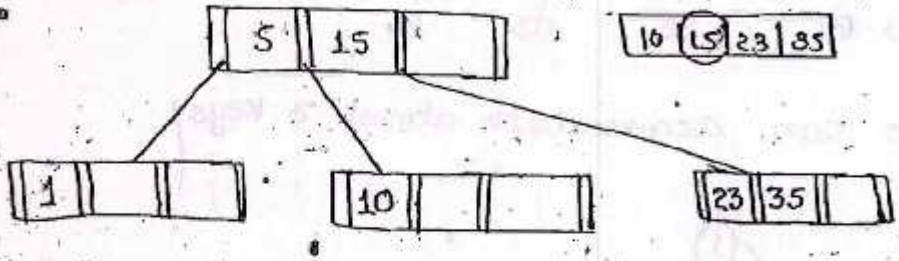






Always Insertion should be done from leaf node.

↳ If we insert 23 in root then it violate B-tree definition.



13

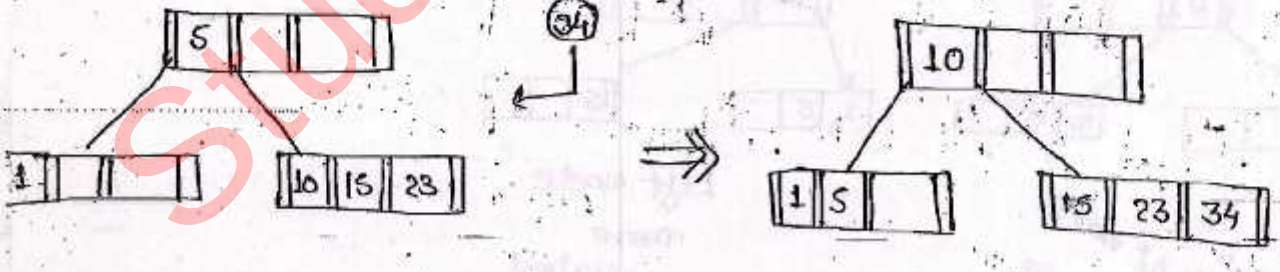
Atcrnative for Node Splitting

Redistribution of keys

left biasing or right biasing

more efficient than

if any leaf node contains free space to accomodate new key, then distribute the keys, i.e accomodating new key within the existing nodes without creating new key.

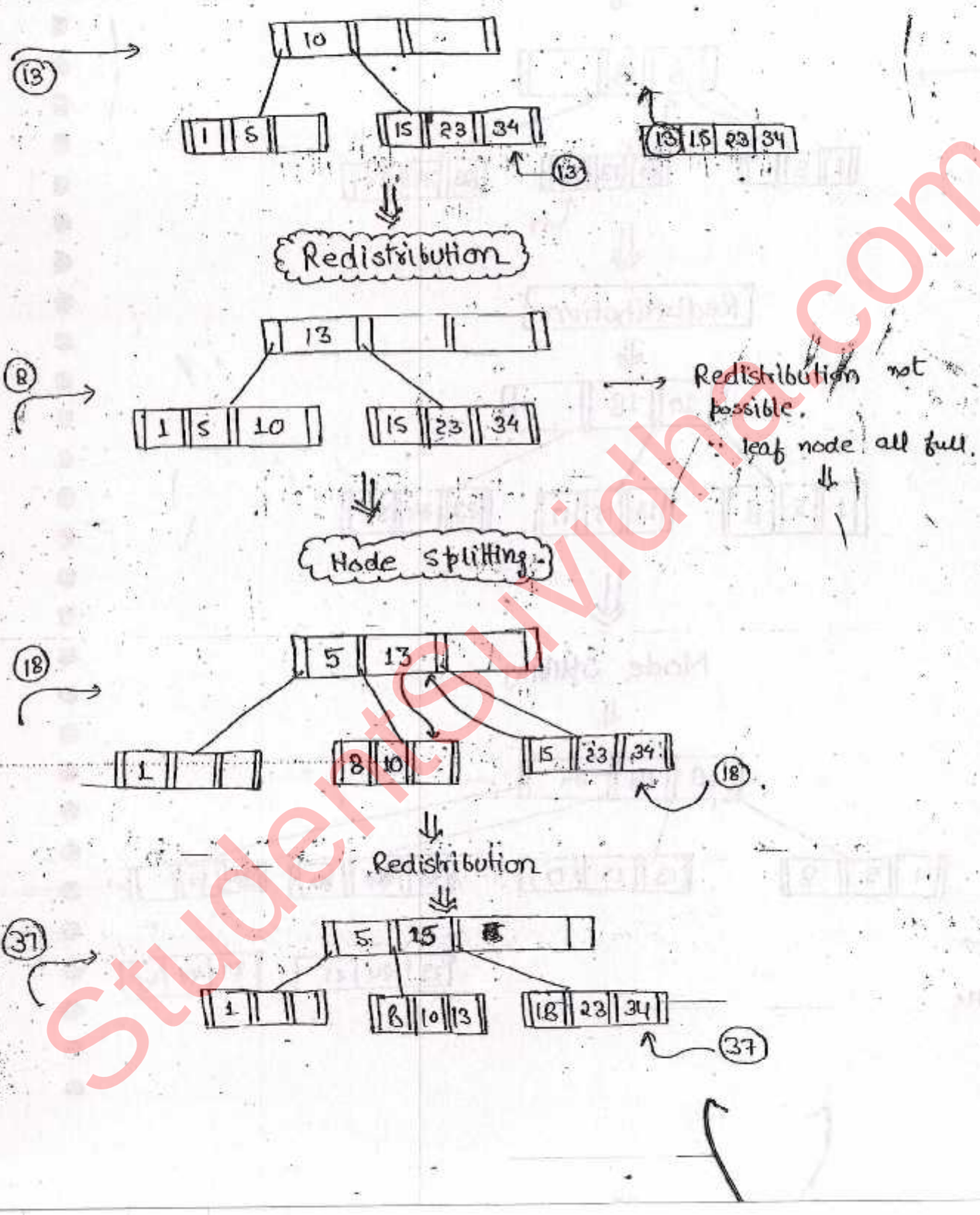


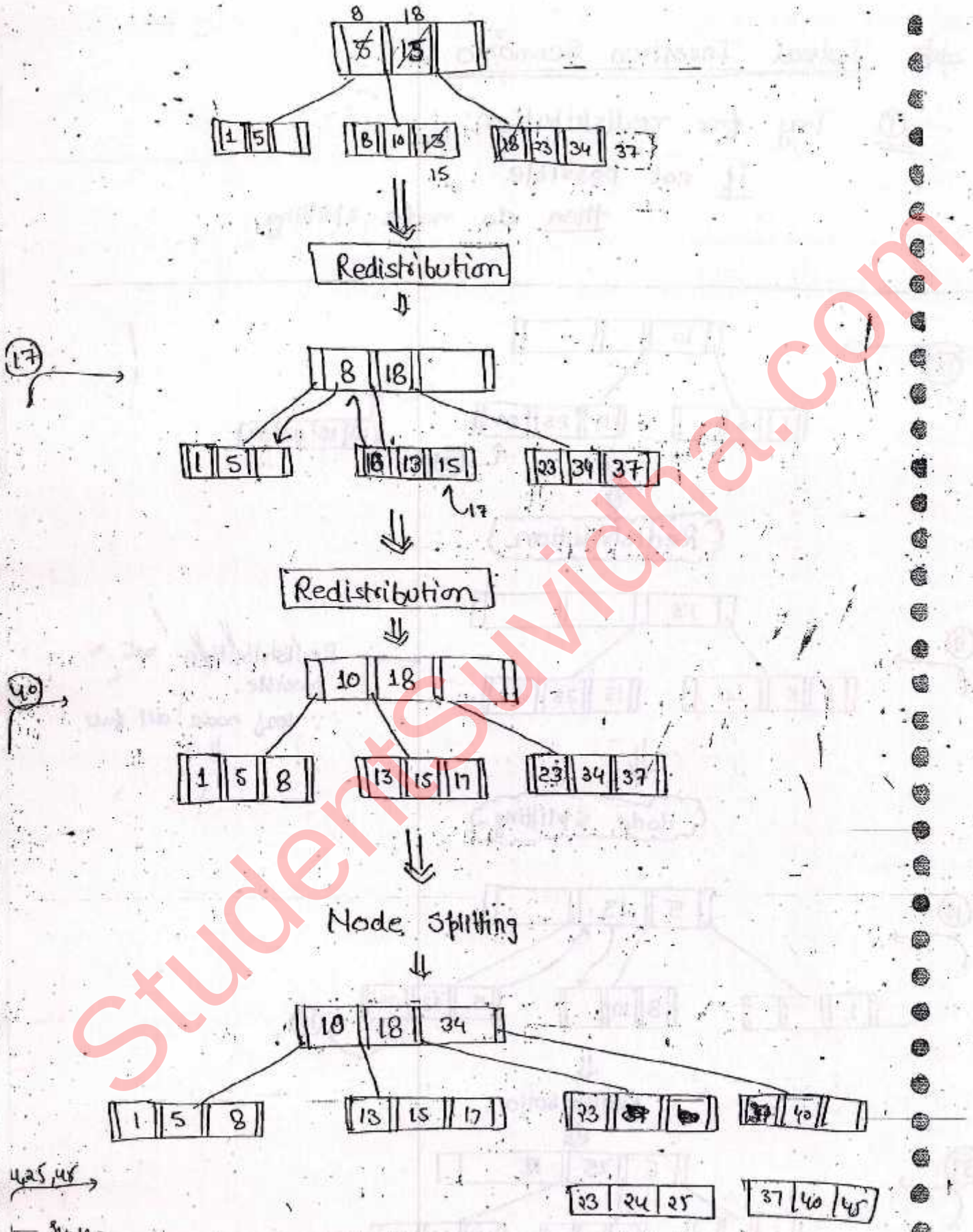
Duplicate keys not allowed here.

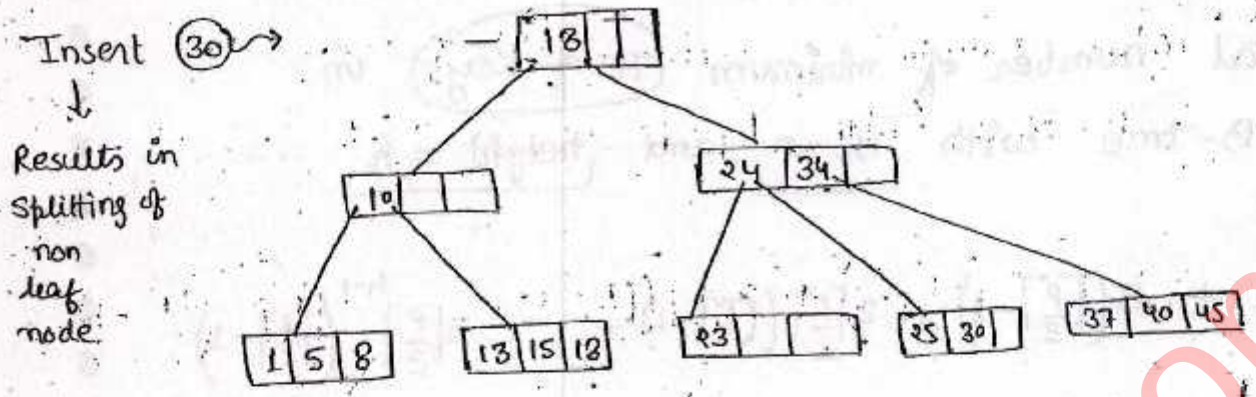
↳ ∵ it is search tree.

Ideal Insertion Scenario :-

(1) Try for redistribution.
if not possible
then do node splitting.







ORDER p : Maximum no. of Block Pointers ≡ No. of children

Height / LEVEL	min # of nodes	min # of Block Pointers	min # of keys
0 / 1	1	2	1
1 / 2	2	$2\times\lceil\frac{p}{2}\rceil$	$2(\lceil\frac{p}{2}\rceil-1)$
2 / 3	$2\times\lceil\frac{p}{2}\rceil$	$2\times\lceil\frac{p}{2}\rceil^2$	$2\lceil\frac{p}{2}\rceil(\lceil\frac{p}{2}\rceil-1)$
3 / 4	$2\times\lceil\frac{p}{2}\rceil^2$	$2\times\lceil\frac{p}{2}\rceil^3$	$2\lceil\frac{p}{2}\rceil^2(\lceil\frac{p}{2}\rceil-1)$
h / h+1	$2\times\lceil\frac{p}{2}\rceil^{h-1}$	$2\lceil\frac{p}{2}\rceil^{h}$	$2\lceil\frac{p}{2}\rceil^{h}(\lceil\frac{p}{2}\rceil-1)$

equals

(2 to p)

$\lceil p/2 \rceil$ $\lceil p/2 \rceil$

$2\lceil\frac{p}{2}\rceil$

For Height (1)

mini no. of null block ptrs pointers.

= Total number of minimum (# of keys) in B-tree with $B_p = p$ and height $= h$.

$$= 1 + 2\left(\left\lceil\frac{p}{2}\right\rceil - 1\right) + 2\left\lceil\frac{p}{2}\right\rceil\left(\left\lceil\frac{p}{2}\right\rceil - 1\right) + \cdots\cdots 2\left\lceil\frac{p}{2}\right\rceil^{h-1}\left(\left\lceil\frac{p}{2}\right\rceil - 1\right)$$

$$= 1 + 2\left(\left\lceil\frac{p}{2}\right\rceil - 1\right)\left\{1 + \left\lceil\frac{p}{2}\right\rceil + \left\lceil\frac{p}{2}\right\rceil^2 + \cdots\cdots \left\lceil\frac{p}{2}\right\rceil^{h-1}\right\}$$

$$= 1 + 2\left(\left\lceil\frac{p}{2}\right\rceil - 1\right)\left\{\frac{\left\lceil\frac{p}{2}\right\rceil^h - 1}{\left\lceil\frac{p}{2}\right\rceil - 1}\right\}$$

$$= \boxed{1 + 2\left\{\left\lceil\frac{p}{2}\right\rceil^h - 1\right\}} \#$$

$\Rightarrow$ **Level l, height $= l-1$**

Min # keys in B-tree with $B_p = p$ & Level l

$$\boxed{1 + 2\left\{\left\lceil\frac{p}{2}\right\rceil^{l-1} - 1\right\}} \#$$

F No. of Record Pointers = No. of keys.

Every key is associated with each ~~key~~ record pointer.

Height	Max # of Nodes	Max # of B. Ptrs	Max # of keys
0	1	p	p-1
1	p	p^2	$p(p-1)$
2	p^2	p^3	$p^2(p-1)$
3	p^3	p^4	$p^3(p-1)$
⋮	⋮	⋮	⋮
h	p^h	p^{h+1} (ptrs pointing to null)	$p^h(p-1)$

⇒ Max. # of keys in B-tree with order p and height h

$$(p-1) + p(p-1) + p^2(p-1) + \cdots + p^h(p-1)$$

$$(p-1)\{1 + p + p^2 + \cdots p^h\}$$

$$(p-1)\left\{\frac{p^{h+1}-1}{p-1}\right\}$$

$$= \boxed{p^{h+1}-1} \quad \#$$

⇒ level h ; order p.

$$= \boxed{p^h - 1} \quad \#$$

Q. Height of B-tree = ?

ORDER = p

keys = N. (can be anything from minimum to maximum)

Case I :-

We know that,

$$N = 1 + 2\left\{\left[\frac{p}{2}\right]^{h} - 1\right\}$$

$$\# \quad h = \log_{\lceil p/2 \rceil}\left\{\frac{(N-1)}{2} + 1\right\}$$

min. # of keys/node

max height atmost

Case II $N = p^{h+1} - 1$

$$h = \log_{p}(N+1) - 1$$

atleast min. height

we consider max # keys/node

Height of B-Tree

$$\equiv \Theta(\log_p N) \#$$

ORDER p

min	max
p/2 ←	p
p →	2p

max = 200% (min)

or

$$\frac{\text{min}}{p} = \frac{50\%}{\text{max } p}$$

#

Identify minimum number of nodes and keys in B-tree with order $p=4$ and level $l=5$.

(i) and assume. order p = max. no. of block pointers in B tree node.

(ii) " " p = " " " keys in " "

Soln Mini. no. of nodes with order p and level l is:

$$1+2+2\times\left\lceil\frac{p}{2}\right\rceil+2\times\left\lceil\frac{p}{2}\right\rceil^2+2\times\left\lceil\frac{p}{2}\right\rceil^3+\cdots\cdots 2\times\left\lceil\frac{p}{2}\right\rceil^{l-1}$$

$$=1+2\left\{1+\left\lceil\frac{p}{2}\right\rceil+\left\lceil\frac{p}{2}\right\rceil^2+\cdots\cdots\left\lceil\frac{p}{2}\right\rceil^{l-2}\right\}$$

$$=1+2\left\{1+\left\lceil\frac{p}{2}\right\rceil^2+\left\lceil\frac{p}{2}\right\rceil^3\right\}$$

$$=1+2\left\{1+\left\lceil\frac{4}{2}\right\rceil^2+\left\lceil\frac{4}{2}\right\rceil^3\right\}$$

$$=1+2\left\{1+4+8\right\}$$

$$=27$$

Level	min # of nodes	min # of B p'	min # of keys
1	1	$\lceil\frac{p}{2}\rceil=2$	$\lceil\frac{p}{2}\rceil-1=1$
2	2	$2\lceil\frac{p}{2}\rceil=4$	$2\times\lceil\frac{p}{2}\rceil-1=2\times1$
3	4	$4\lceil\frac{p}{2}\rceil=8$	$4(\lceil\frac{p}{2}\rceil-1)=4\times1$
4	8	$8\lceil\frac{p}{2}\rceil=16$	$8(\lceil\frac{p}{2}\rceil-1)=8\times1$
5	16	$16\lceil\frac{p}{2}\rceil=32$	$16(\lceil\frac{p}{2}\rceil-1)=16$

Capacity Based on definition of ORDER P:

(i) ORDER P: Max # of B_p in B tree node

$$P\,B_p + (P-1)\,key + (P-1)\,R_p$$

→ Default

(ii) ORDER P: Max # of keys in B-tree node

$$(P+1)\,B_p + P \text{ keys} + P.R_p$$

(iii) ORDER P: Min # of B_p in B-tree node

$$2p.B_p + (2p-1)\,keys + (2p-1)\,R_p$$

(iv) ORDER P: Min # of keys in B Tree node

	min	max
keys	P	2P
B_p	P+1	(2p+1)

$$(2p+1)\,B_p + 2p \text{ keys} + 2p\,R_p$$

→ Capacity OR BLOCK SIZE

max keys = 4
Max B_p = 5

(II) $(P+1)\,B_p + P \text{ keys} + P\,R_p$

Level	min # of nodes	min no of B_p	min # of keys
1	1	2	1
2	2	$2\left\lceil\frac{p}{2}\right\rceil = 2*3$	$2\left(\left\lceil\frac{p}{2}\right\rceil - 1\right)$ $2\left(\left\lceil\frac{5}{2}\right\rceil - 1\right)$ $2*2 = 4$
3	6	$6\times\left\lceil\frac{5}{2}\right\rceil = 6\times3$	$6*\left(\left\lceil\frac{p}{2}\right\rceil - 1\right)$ $6*\left(\left\lceil\frac{5}{2}\right\rceil - 1\right) = 12$
4	18	$18\times\left\lceil\frac{5}{2}\right\rceil = 54$	$18*\left(\left\lceil\frac{5}{2}\right\rceil - 1\right)$ $= 18*2 = 36$
5	54	$54\times3 = 162$	$54*2 = 108$

1+4+12+36+108

(III) Max. = ? P = 4

Level	Max. # of nodes	Max. # of B_p	Max # of keys
1	1	4	3
2	4	4×4	4×3
3	16	16*4	16*3

48+12+3

(21)

(IV) $p = 4$

$h = 3$

p keys + $(p+1)B_p + p\ R_p$

$B_p = 5$

level	max # nodes	max. # Bp	max. key
1	1	$(p+1) = 5$	$p_i = 4$
2	5	$5*5 = 25$	$5*4 = 20$
3	25	$25*5 = 125$	$25*4 = 100$

31 #

4+20+100

= 124 #

Pg 55
Q 55
12

2/11/11

Q12/(55)

P denotes max # of tree pointers in B tree node

Key = 10 Bytes

Block size = 512 Bytes

Data ptr = 8 Bytes

Block Pointer 5 Bytes

what is the max value of P.

Soln | P = max # B_p in B-tree node

$$P * B_p + (P-1)\,Keys + (P-1)\,R_p \leq Block\_size$$

$$P * 5 + (P-1)10 + (P-1)8 \leq Block\_size$$

$$23P - 18 \leq 512$$

$$23P \leq 530$$

$$P = \frac{530}{23} = 23.0434$$

P = 23

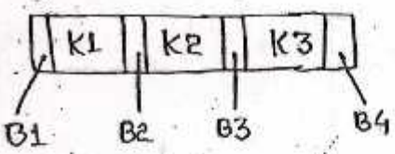
Can't take 24 ∵ then it exceeds 512.

9) ORDER P: max # of keys in B-Tree

$$(P+1)B_p + P * Keys + P R_p \leq Block\ size$$

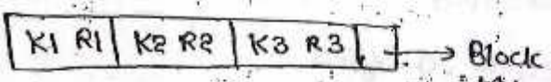
B^+ Tree

Internal Node



→ V_4 B_p

Leaf Node



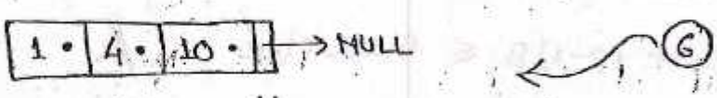
3 − R_p
1 − B_p

Q Construct B^+ Tree with order P=4 for following sequence of keys:

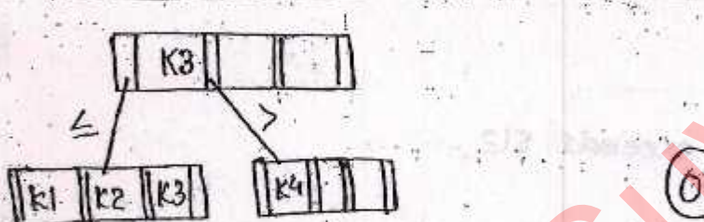
4, 10, 1, 6, 12, 25, 13, 7, 45.

⇒ Order = 4

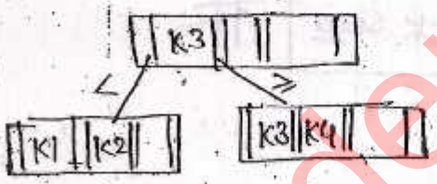
Keys in Block = 3



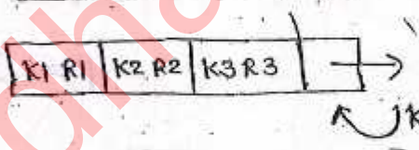
⇓ Node splitting



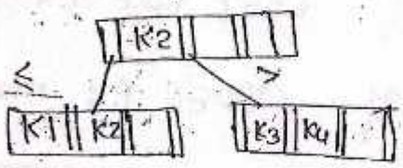
OR



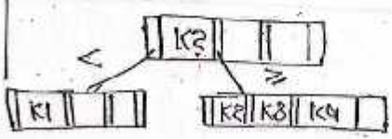
Leaf Node splitting



K1 < K2 < K3 < K4



OR



→ If order even ——— 4 ways of splitting

B^+ Tree

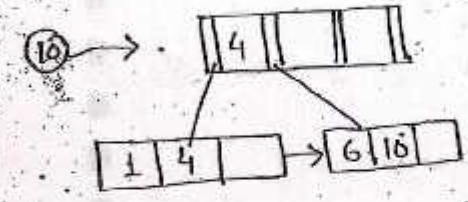
↳ If order = even, — 4 ways of splitting

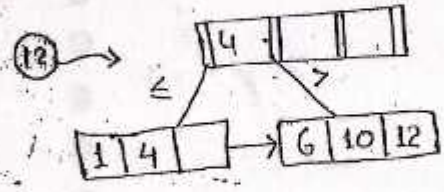
" " = odd, — 2 ways of splitting.

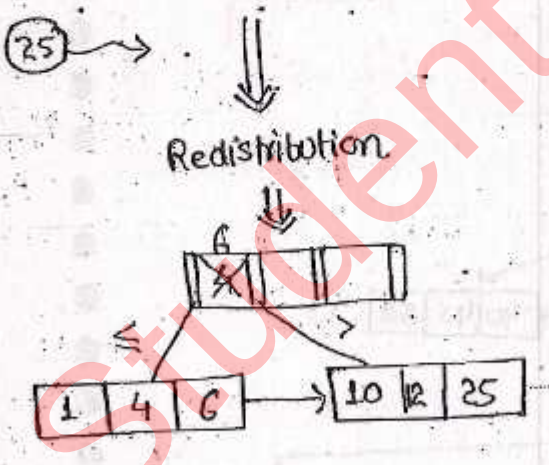
B Tree

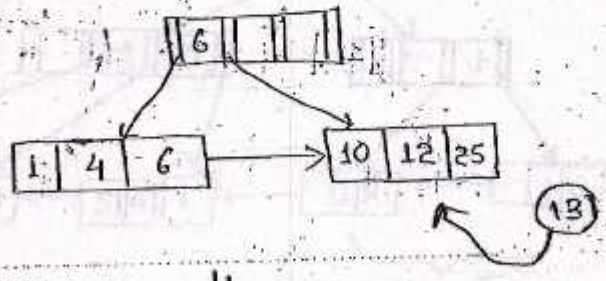
↳ If order = even, — 2 ways of splitting

" " = odd, — 1 " " "

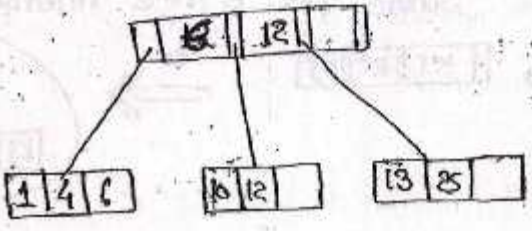




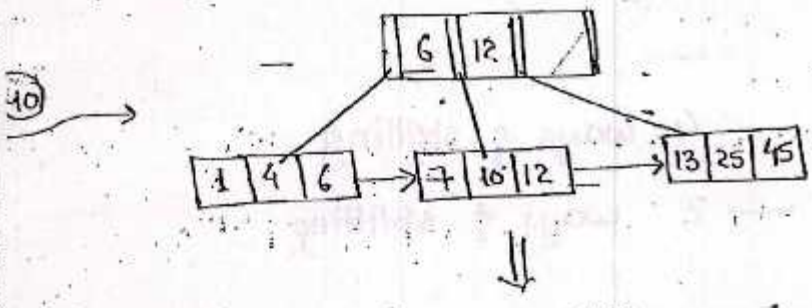




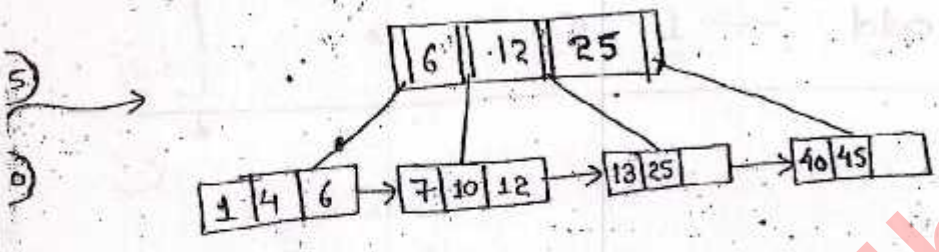
Node splitting

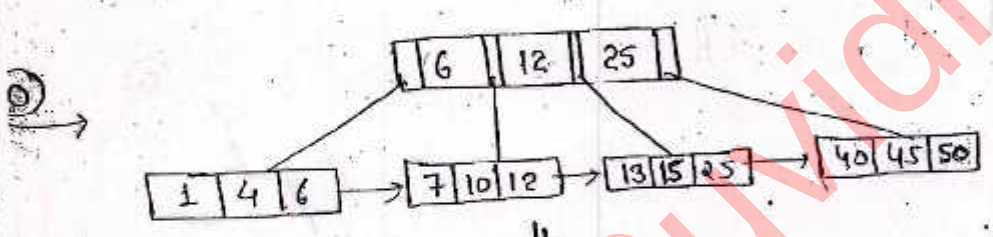


(7)

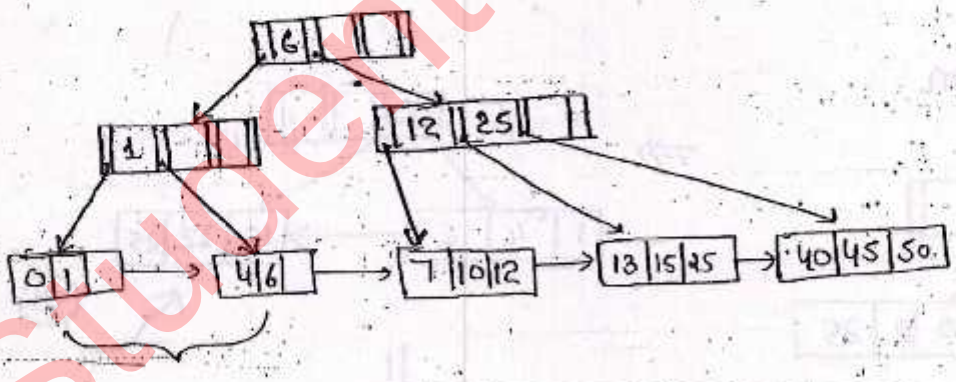


Node Splitting



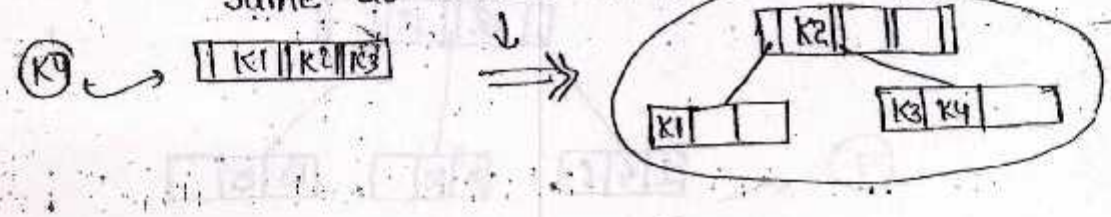


Node Splitting



⇒ Non-leaf Node Splitting :-

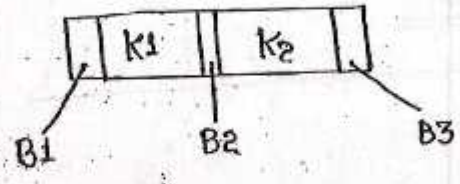
Same as B tree internal node splitting.



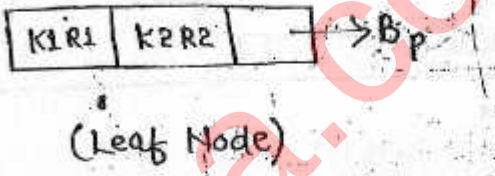
Pg 56
Q 16

Order Internal Node = 3
" Leaf Node = 2

Max no. of Tree pointers = ORDER 3 ≡ B_p
" data items = 2



(Internal Node)



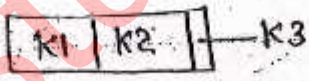
(Leaf Node)

⇒ max # of times leaf node splitted ???

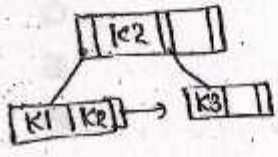
10, 3, 6, 8, 4, 2, 1.

⇒ When max. no of splitting required, ~~go for~~ don't consider redistribution.

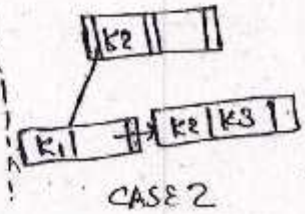
⇒ When mini. no of splitting required always consider redistribution.

⇒ 

K1 < K2 < K3



CASE 1



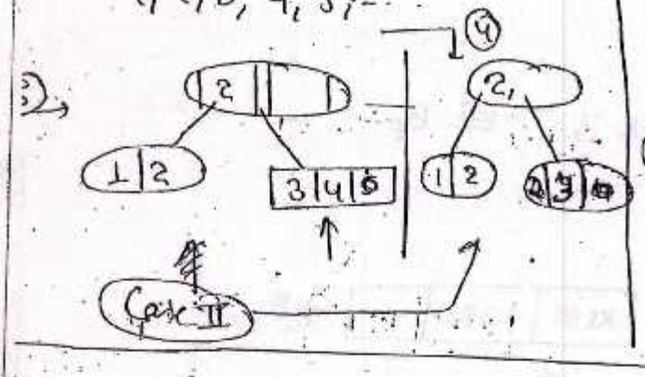
CASE 2

In both the alternatives No. of splittings may differ.

→ Ascending order of keys — CASE (2) result more splits.

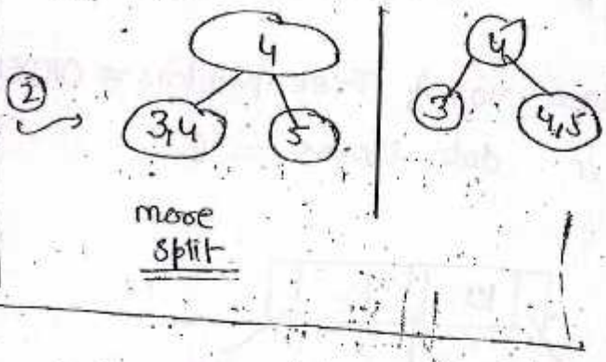
Ascending

1, 2, 3, 4, 5, ...

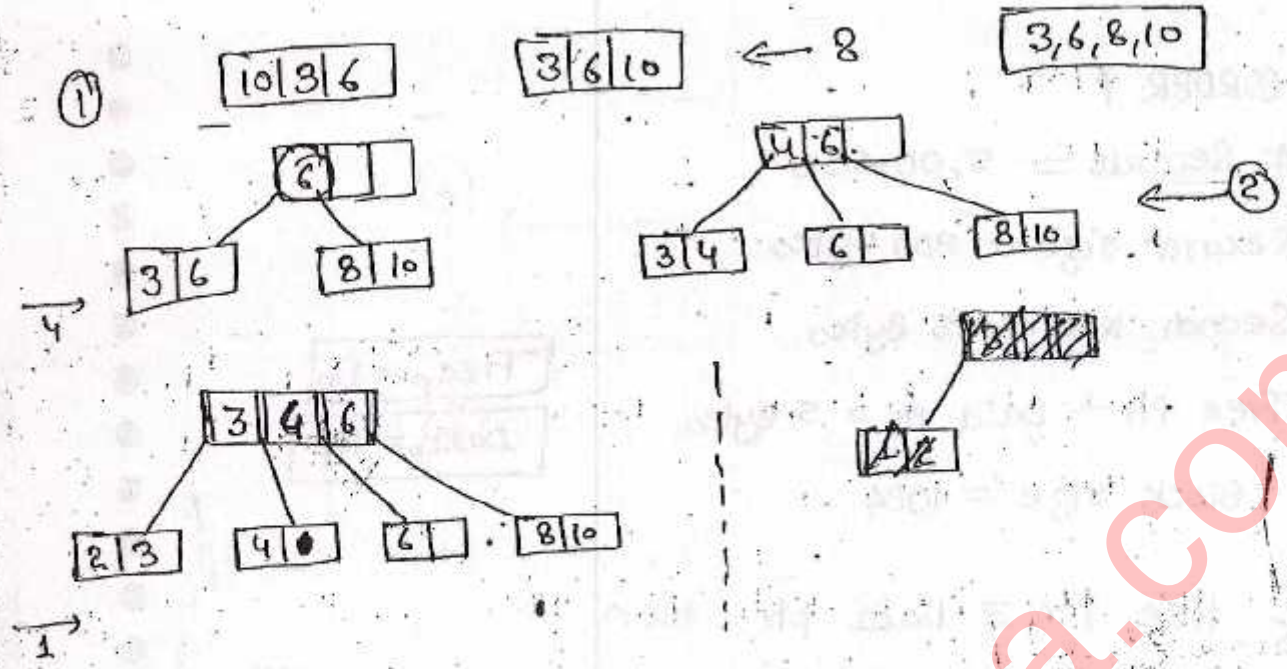


Desendg

5, 4, 3, 2, 1



mode split



---

(5) Key = 8 Byte
Block = 512
Index Ptr = 4 Bytes
Best choice of degree (no. of ptr per node) in B+ tree ???

Sol^n Internal node

Q6. ORDER p

Records = 5,00,000

Record size = 200 Bytes

Search Key = 15 Bytes

Tree Ptr = Data Ptr = 5 Bytes.

Block size = 1024

$Tree_p = B_p$

$Data_p = R_p$

If tree ptr ≡ Data ptr then

Order of Internal node ≡ order of leaf node #

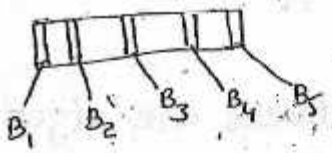
Internal Node

$p * B_p + (p-1)$ keys $\leq$ Block size

$(p-1)$ keys $+ (p-1) R_p + B_p \leq$ Block size

No. of pointers in both internal/external (leaf) node equals "p" (SAME) for both.

(8) (1) ORDER p = max # of child it can have
↳ B_p



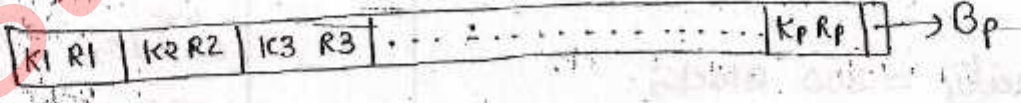
$$p * B_p + (p-1)\,\text{Keys} \leq \text{Block\_size}$$ #

(9) Leaf Node Order = Max # of (Value, R_p) pairs

Capacity p (value, R_p) pair

p keys + p R_p + 1 B_p

$$p * \text{Keys} + p * R_p + 1 * B_p \leq \text{Block Size}$$



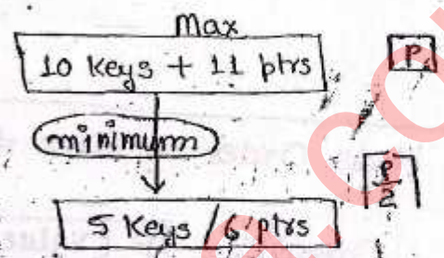
Q:13 ORDER=??

Q: Suppose blocks are sized such that they can either hold 5 records of relation R or be used as B^+ tree node with 10 keys and 11 pointers.
If relation consist 1000 records, what is the smallest no. of blocks that could be used to store relation and sparse B^+ tree index.

a) 210 b) 221 c) 223 d) 245.

⇒ Sparse B^+ tree index =???
with mini/smallest no. of blocks.

└ B^+ tree node can accomodate



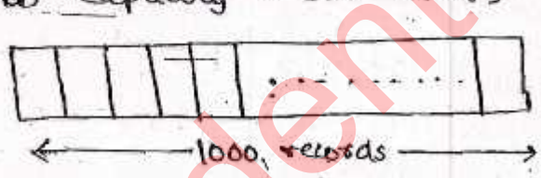
⇒ minimum number of blocks required so, to achieve that, we consider

maximum occupancy per node

10 Key + 11 pointers

If he ask for ~~minimum~~ maximum #of blocks then we consider minimum occupancy per node

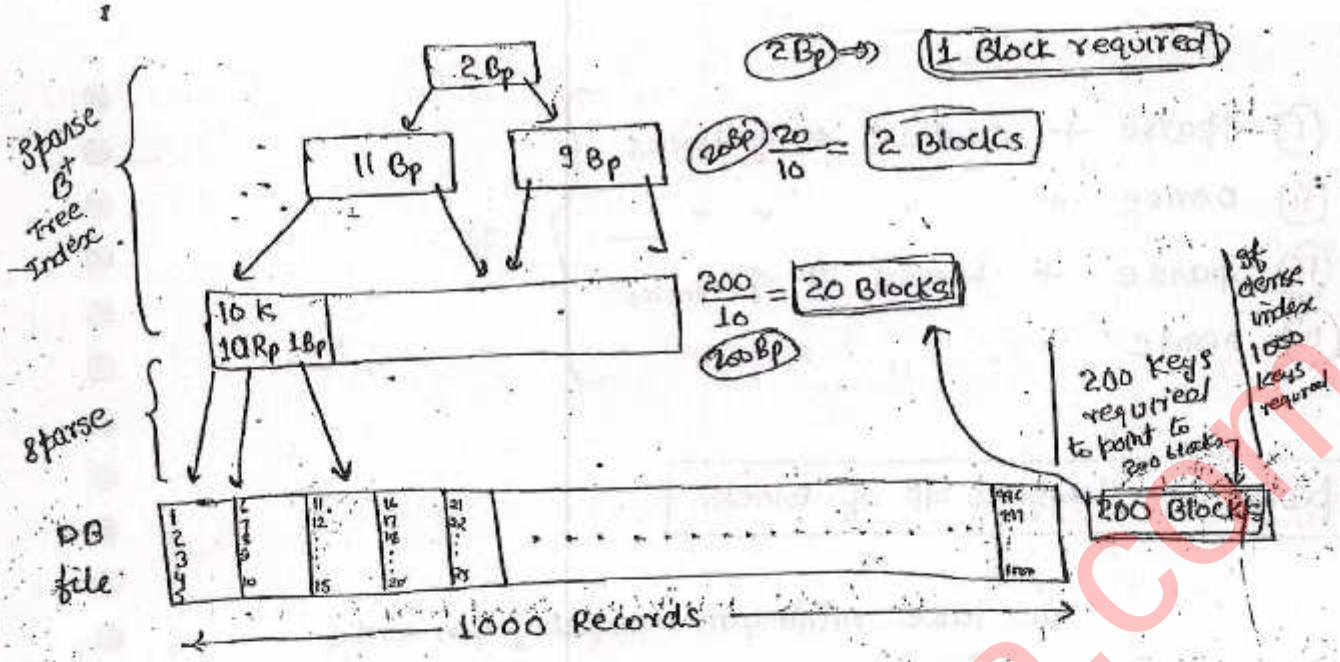
⇒ DB capacity = 200 Blocks



$\frac{1000}{5}$ = 200 Block

∴ Each block contain 5 records.

DB Size = 200 Blocks

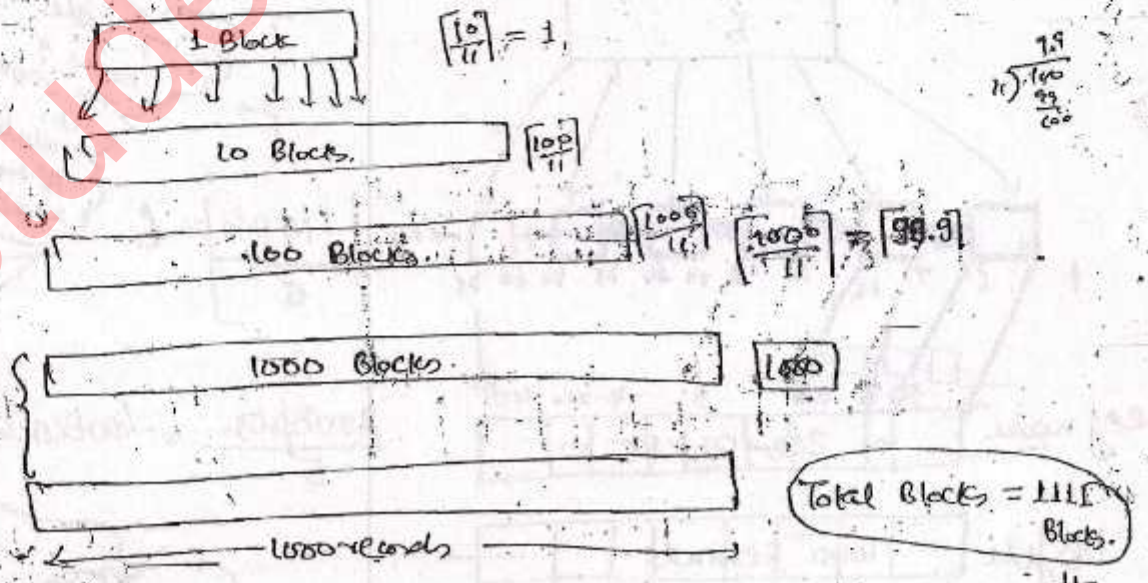


Total no. of blocks required are:-

$200 + 20 + 2 + 1 = 223$ Ans.

(ii) Dense + Smallest # of blocks.

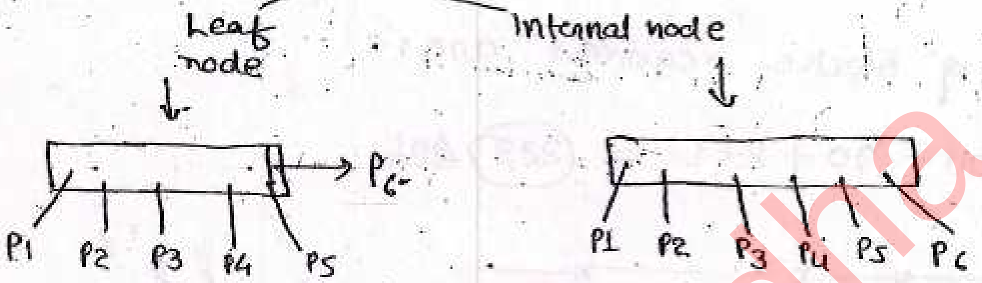
↳ For dense index, every entry has an index.



(i) sparse + smallest # of Blocks
(ii) Dense + " " "
(iii) Sparse + Largest # of Blocks
(iv) Dense + " " "

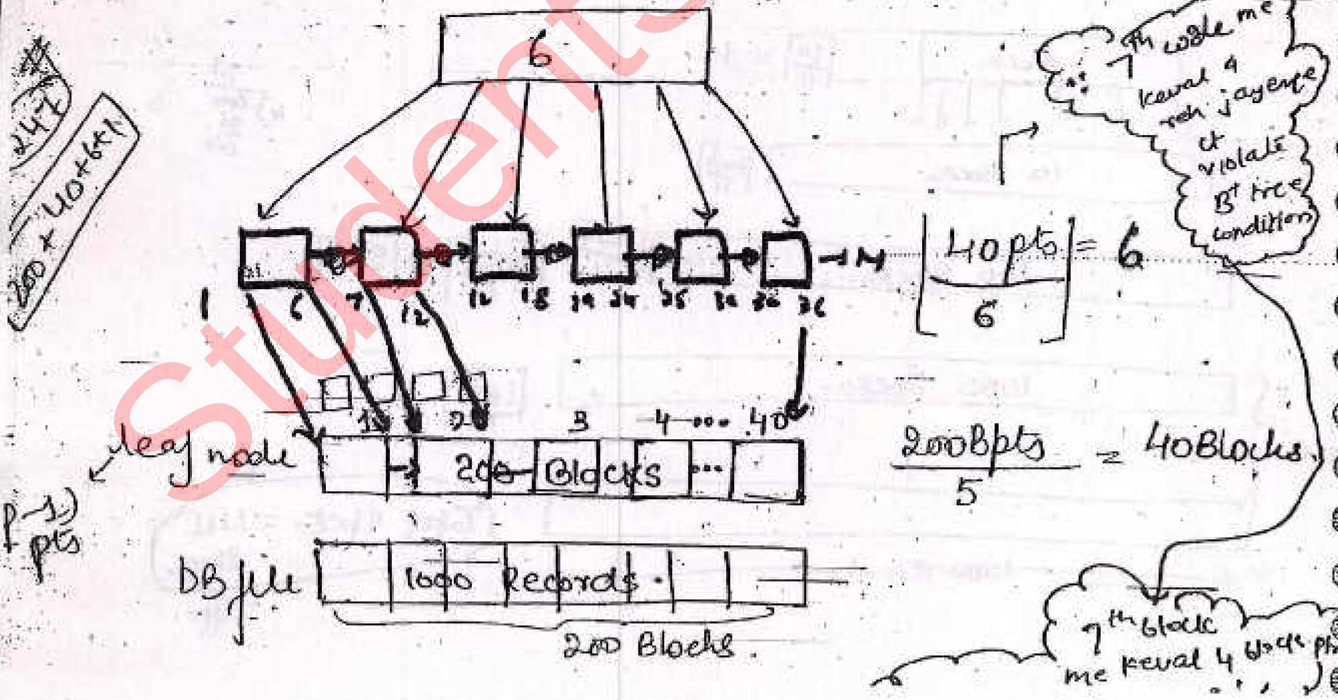
\# Sparse + Largest # of Blocks.

we take minimum occupancy per node



$Keys = \lceil \frac{10}{2} \rceil$ $\lceil \frac{11}{2} \rceil = B_p$ $B_p = \lceil \frac{11}{2} \rceil$

largest: 5 Keys + 6 pts



$\left\lceil \frac{40 pts}{6} \right\rceil = 6$

$\frac{200 Bpts}{5} = 40 Blocks$

\# 247 = 200 + 40 + 6 + 1