June 14 (11)

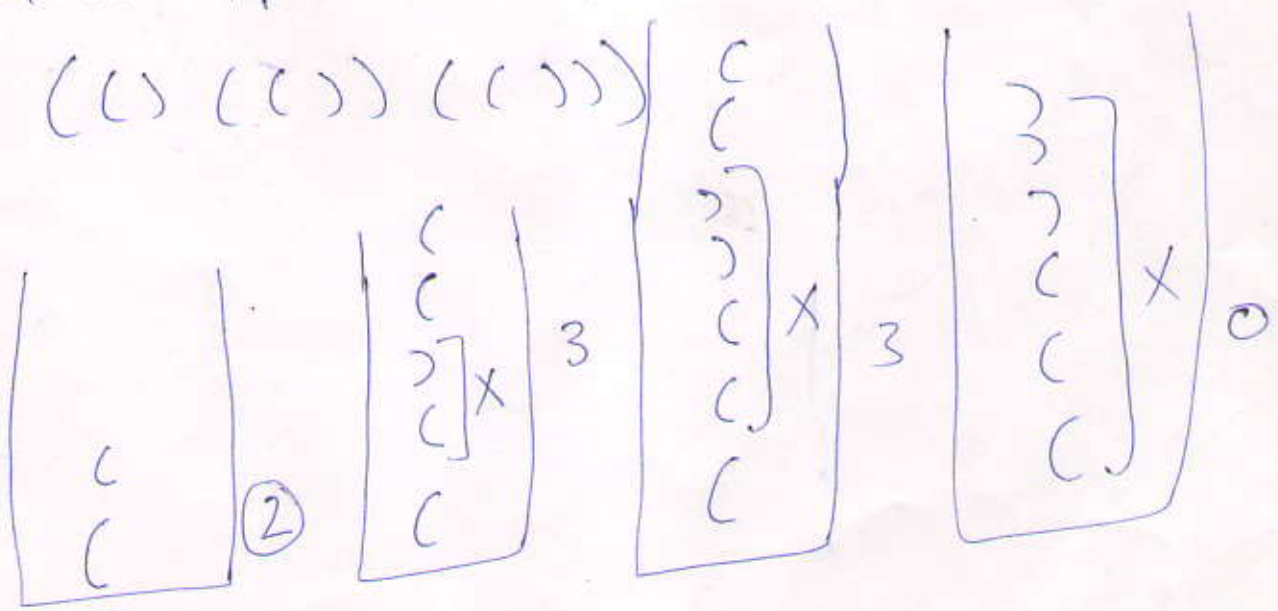
38. Searching for an element in the hash table requires O(1) time for the average time whereas for the direct addressing it holds for the worst case time.

(39) An algorithm is made up of 2 modules M_1 and M_2. If time complexity of modules M_1 and M_2 are $h(n)$ and $g(n)$ respectively, the time complexity of algo. is.

Ans $\max(h(n), g(n))$

(40) What is the max. no. of parenthesis that will appear on the stack at any one time for parenthesis expression given by

(() (()) (()))



Ans 3

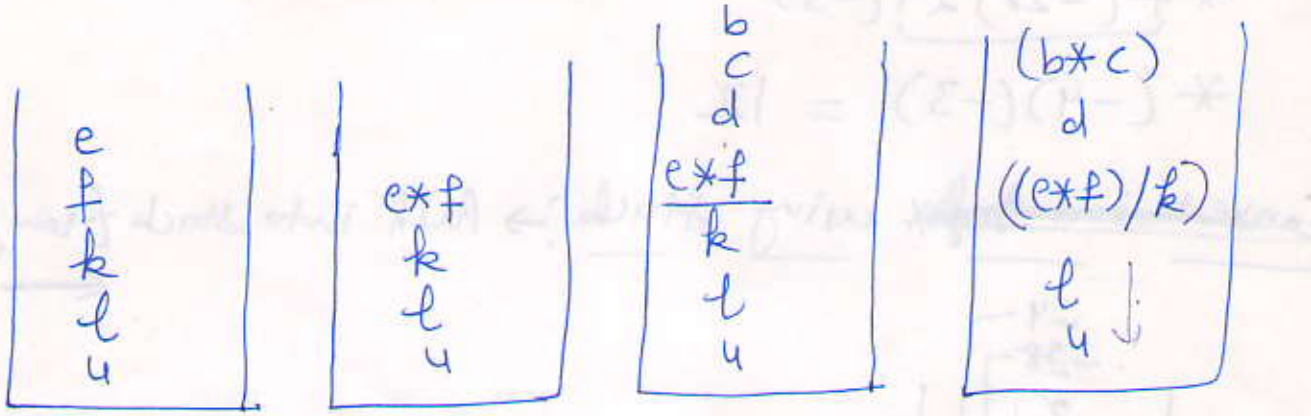
(36) Mergesort make two recursive calls. Which statement is true after these two recursive calls finish, but before the merge step. (34)

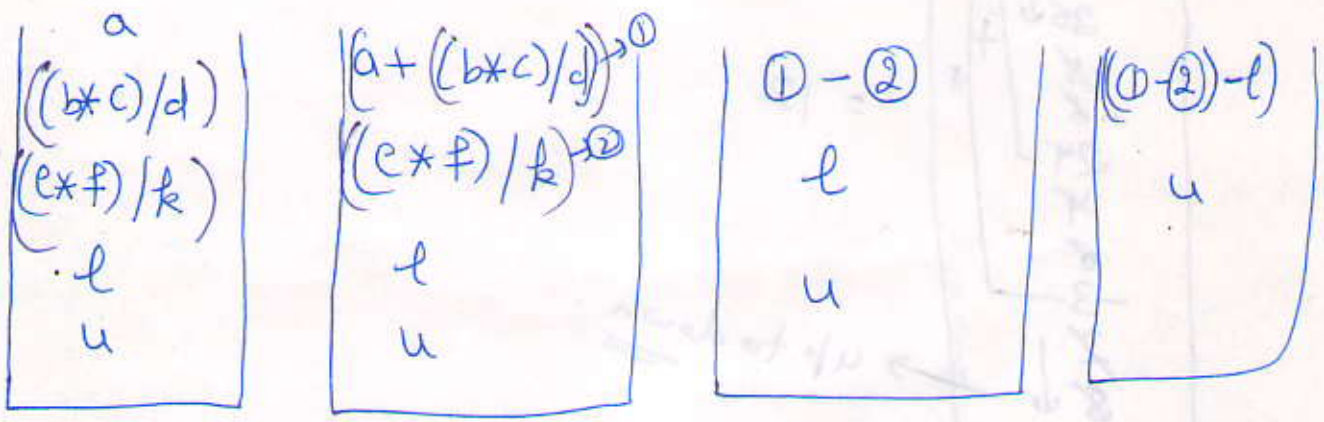
Sol:- Elements in each half of the array are sorted among themselves.

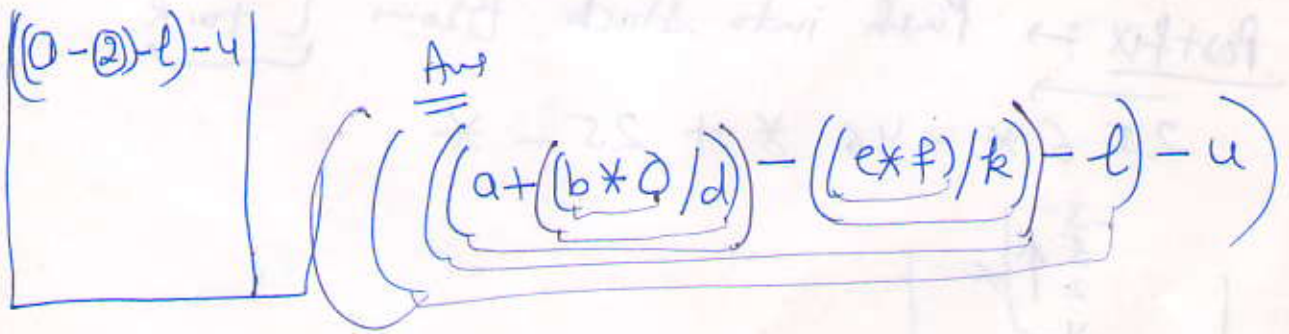
Prefix to Infix using Stack :

(3) (4)

$---+a/*bcd/*efkLu$ ←







Evaluate :→ Prefix

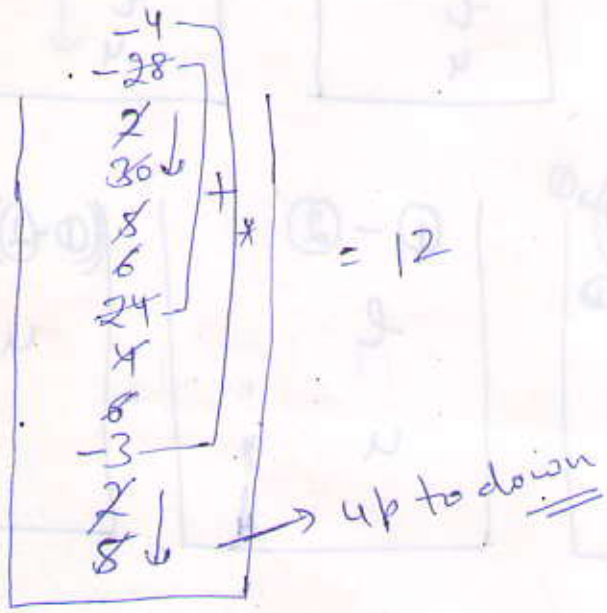
→
* + − 2 * 5 6 * 4 6 − 2 5

* + − 2 30 24 (−3)

* + (−28) 24 (−3)

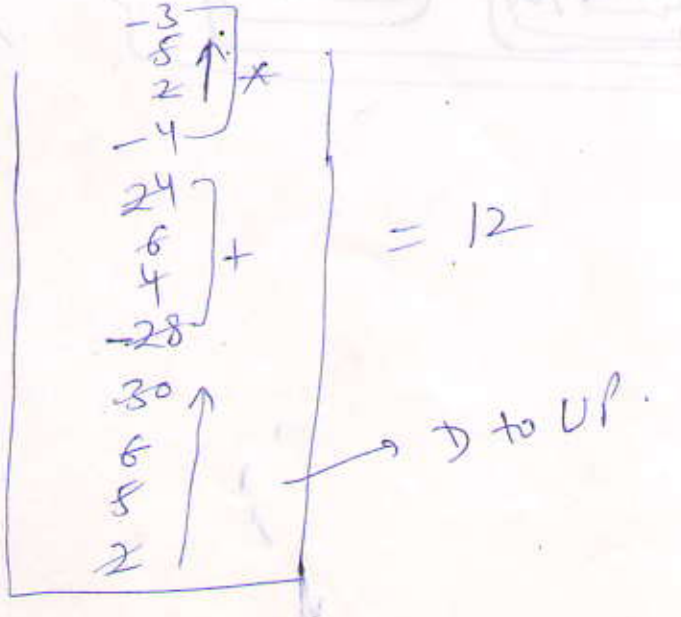
* (−4) (−3) = 12

~~Convert into Infix~~ using stack :→ Push into stack from R to L ←



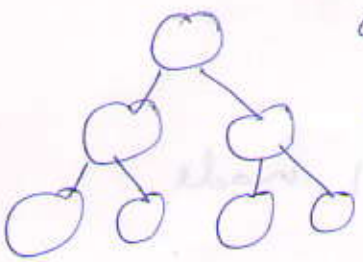
Postfix :→ Push into stack from L to R.

→
2 5 6 * − 4 6 * + 2 5 − *



Binary Tree :→ Let no. of nodes in a binary tree is n then find Min & Max height of a binary tree.

Sol:→ **Min height** means each level has max (ie 2) no. of children.



at level 0 2^0 nodes
at level 1 2^1 n
at n 2 2^2 n
⋮
at n h 2^h nodes.

Let h be the height of this binary tree. and n be the total no. of nodes.

So. $2^0 + 2^1 + 2^2 + \cdots 2^h = n$

G.P. series.

$$ar^0 + ar^1 + \cdots ar^n = \frac{a(r^{n+1}-1)}{r-1}, \quad r \geq 1$$

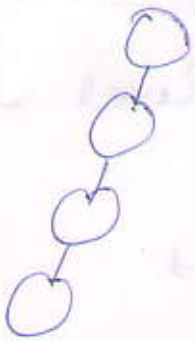
here $a = 1 \quad r = 2$

so $2^{h+1} - 1 = n$

$2^{h+1} = n+1 \Rightarrow h+1 = \log(n+1)$

$h = \log(n+1) - 1$ [Min height]

Max height :→ To get max height, each level will have min no. of nodes



at level 0 ~~0 width~~ 1 node

at level 1 1 node

n 1 2 1 n

!

at n h 1 node

So.

$1 + 1 + \ldots (h+1) \text{ time} = n$

$h+1 = n \Rightarrow h = n-1$ [Max height]

1) Let height of a binary tree is h then find the Min and max. no. of nodes.

Min nodes $n = h+1$

Max nods $n = 2^{h+1} - 1$

Q :→ Let there are n nodes in a binary tree what is the min. and max. no. of leaf and non-leaf nodes in a binary tree respectively.

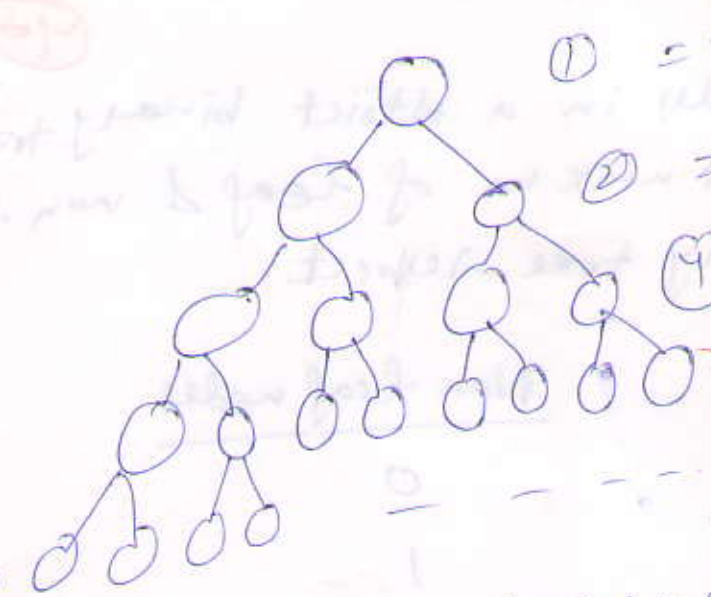
Sol :→ No. of nodes = No. of leaf nodes + No. of non leaf nodes

[Minimum leaf node = 1
Max. non leaf node = h-1

Max leaf node = $\lceil \frac{h}{2} \rceil$

Min leaf node = n − (max leaf node)

Non $= \lfloor \frac{h}{2} \rfloor$



[as Max leaf nodes. [so everything shd be full]

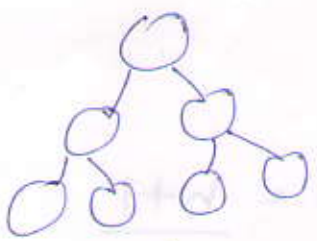
Total nodes = 1+2+4+8+16
= 31

leaf nodes = $\left\lceil \frac{31}{2} \right\rceil$ = 15.5 = 16

So. non leaf nods = 31 − 16 = 15

Q Let there are n (odd) nodes in a strict binary tree then what is the height of strict binary tree.

Sol:→ Min height [Consume max node at each level] ie. (2) for each node



2^0 at level 0
2^1 " " 1
2^2 " " 2
⋮
2^h " " h.

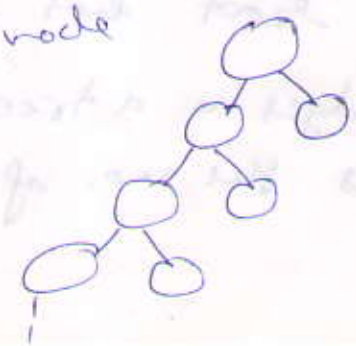
$2^0 + 2^1 + 2^2 + \cdots 2^h = n$

$2^{h+1} - 1 = n$ $\quad h = \log(n+1) - 1$

Max height :→ Consume min node at each level. ie 2.

$1 + (2+2+ \cdots h \text{ time}) = n$

$1 + 2h = n \Rightarrow h = \frac{n-1}{2}$



2^0 = 1 node
2 node
2 node
2 node

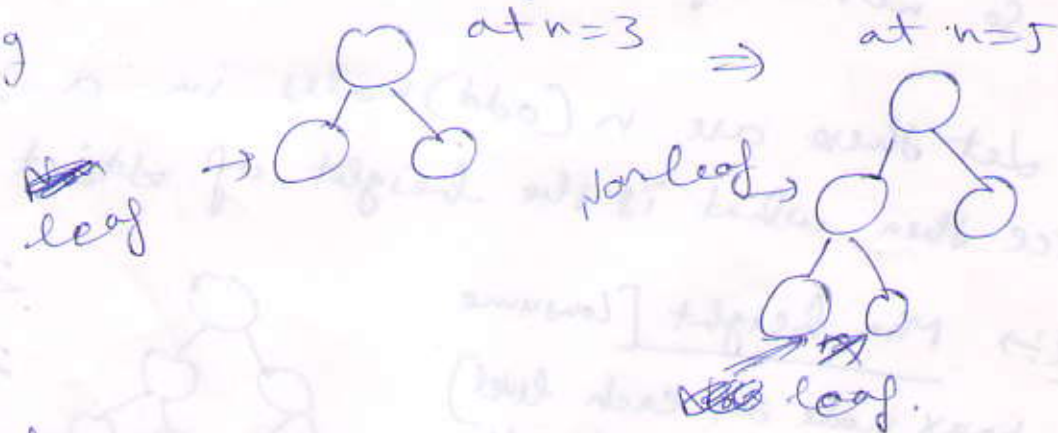
Q:→ Let there are n nodes in a strict binary tree then what is the min. & max no. of leaf & non leaf nodes in a strict binary tree respect.

Sol:

n.	leaf nodes	Non leaf nodes
1	1	0
3	2	1
5	2-1+2=3	1+1=2
7	3-1+2=4	2+1=3

At every step, one leaf node would be converted to non leaf and 2 non leaf would be created.

e.g



So. leaf node = $\frac{n+1}{2}$ Non-leaf = $\frac{n-1}{2}$

It is fixed.. No Minimum or Maximum.

In Strict binary tree / Full binary tree :→ every leaf node has degree = 1
every non leaf has degree = 3
& root has degree = 2

Total no. of edges in a tree = n-1
where n is the no. of vertices

Every edge contributes for degree 2 [1 for 1 vertex]

$$2(n-1) = (\ell \times 1) + (n-\ell-1)\times 3 + 2\times 1$$

(ℓ → leaf, $n-\ell-1$ → Non leaf, 2 → root)

$$2n-2 = \ell + 3n - 3\ell - 3 + 2$$

$$-n = -2\ell + 1$$

$$2\ell - 1 = n \Rightarrow \boxed{\ell = \frac{n+1}{2}}$$

Knary tree :→ Every node must have K or 0 children.

Q :→ Find no. of leaf nodes where no. of nodes in Knary tree is n.

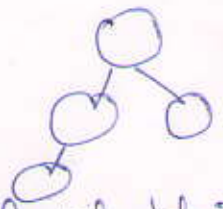
$$2(n-1) = \ell \times 1 + (n-\ell-1)(K+1) + K$$

(K+1 : child, parent)

$$\ell = \frac{(K-1)n+1}{K}$$

If $K=2$ $\ell = \frac{n+1}{2}$

Complete binary tree :→

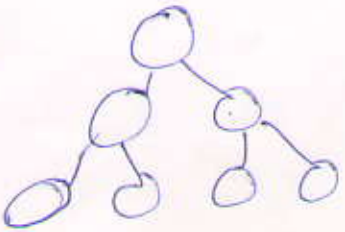


① Binary tree.

② All levels are full except last level that contains node from L to R.

Full binary tree :→ ① Strict binary tree

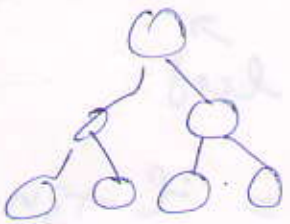
② Every level shd be full



Q:→ Let height of a complete binary tree is h. What is the minimum & max no. of nodes in a complete binary tree.

Sol:→ Max nodes.

$$n = 2^{h+1} - 1$$



height h

Min nodes.

$$2^0 + (2^1 + 2^2 + \cdots 2^{h-1}) + 1$$

$$(2^{(h-1)+1} - 1) + 1 = 2^h$$

height h.

Tree

- Rooted (has parent/child relationship)
- Unrooted (Spanning tree) (does not have parent/child relationship)

Reconstruction of Binary Tree :→

Given ① Inorder

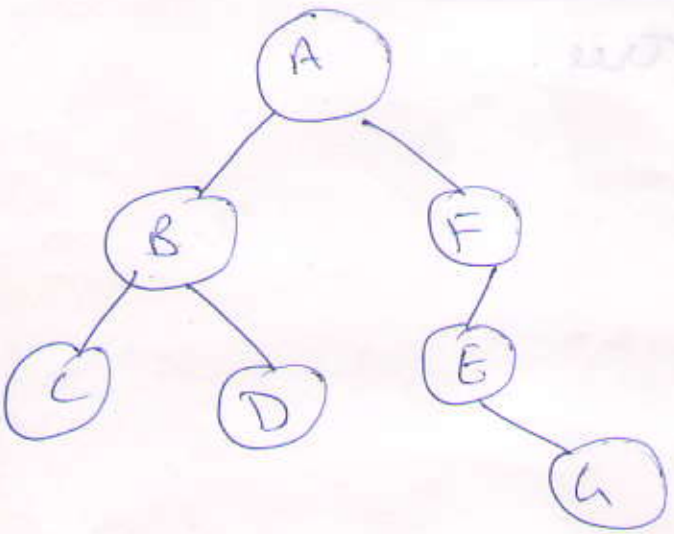
② Preorder / Postorder.

Tree Traversal :→

Inorder :→ left Root Right

Preorder → Root left Right

Postorder → left Right Root.



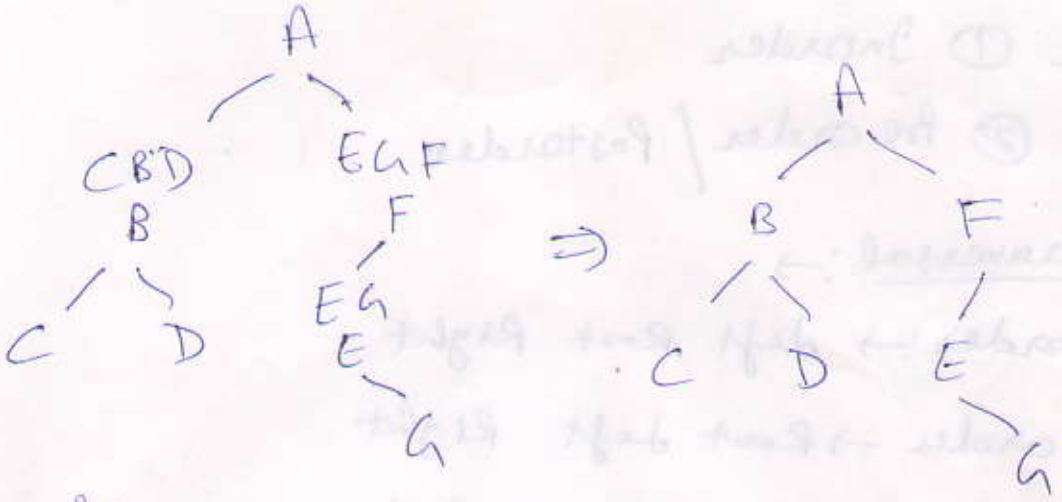
Inorder :→ C B D A E G F [To find left/Right]
(left, Right)

Preorder :→ A B C D F E G [To find root]
(root → A, root → B)

Postorder :→ C D B G E F A ← root

Reconstructing the tree using Inorder and Preorder :→

Reconstructing the tree using Inorder and Postorder :→



Preorder & Post order Given : Cannot determine unique Binary Tree.

A strictly Binary Tree with n leaf nodes always have exactly $2n-1$ nodes.

Main Diff. b/w B & B+ tree

(1) In B tree Root node is in Main memory and rest of the tree is in secondary storage device

(2) In B^+ tree all the indexes are in Main memory and data is in secondary storage device.

(2) In B tree data is stored along with key (Row id/physical add.). Means when you access a key, you get the data immediately.

(2) In B^+ tree is data is stored only on bottom leaves. You have to go to at end to get the data. Once you are at bottom leaves you can access the data sequentially as leaves are connected via doubly linked list. So B^+ is more efficient for range queries [like $5 \le A \le 10$]. As all the data is in one block and can be access at once; once you are at bottom leaf.

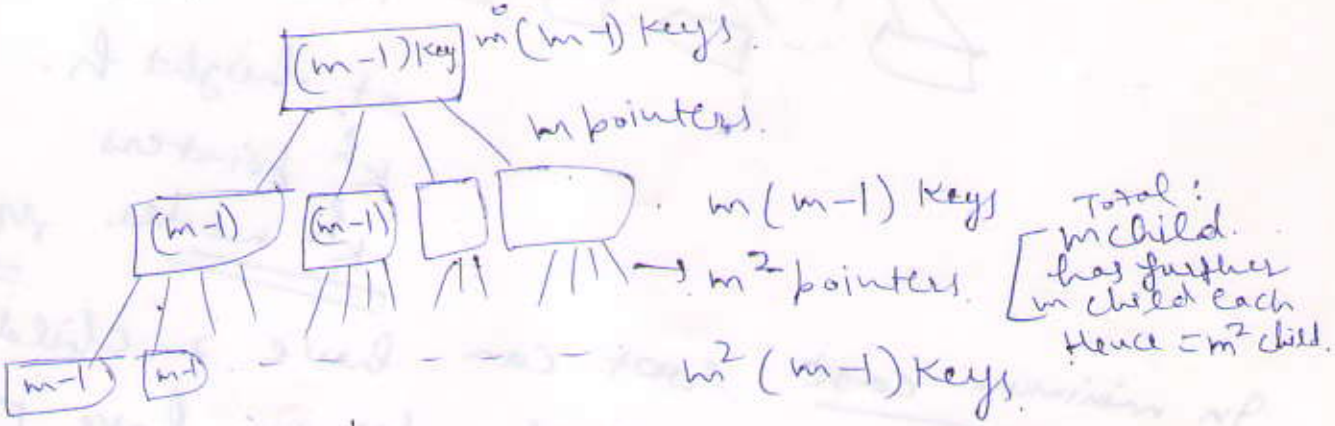
(3) In B tree, each level [pointers + key/data] are stored in one block of secondary storage.

(3) In B^+ tree, only bottom leaves (only data) are stored in one block of secondary storage.

DEC 12 (12)

46 (12)

34) The max. no. of keys stored in a B-tree of order m and depth d is



Max Keys $= m^0(m-1) + m(m-1) + m^2(m-1) + \cdots m^d(m-1)$.

$(m-1)(m^0 + m^1 + m^2 + \cdots m^d)$

$(m-1)\dfrac{(m^{d+1}-1)}{m-1}$ Ans

Acc. to G.P. $\Rightarrow ar^0 + ar^1 + \cdots + ar^n = \dfrac{a(r^{n+1}-1)}{r-1}$, $r \geq 1$

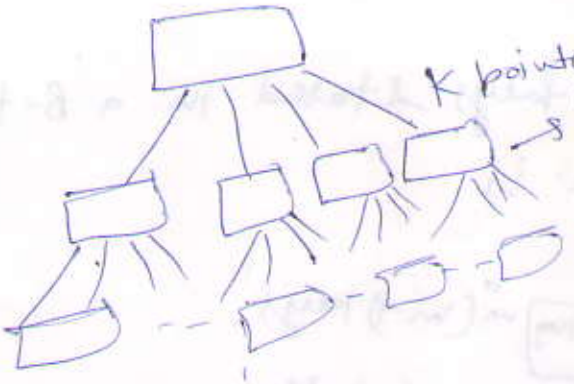
June 14 (11)

(22). The upper bound and lower bound for the no. of leaves in a B-tree of degree k with height h is given by.

upper bound : k^h

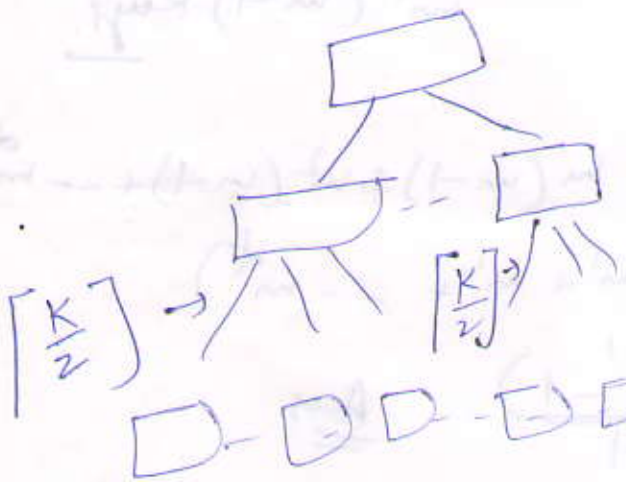
lower bound : $2\lceil k/2 \rceil^{h-1}$ (A)

Upper bound



at height h.
K^h pointers
K^h nodes. Max.

In minimum case root can have 2 children
Non leaf nodes can have $\lceil \frac{K}{2} \rceil$ children



2 nodes [at h=1] $= 2\left(\lceil \frac{K}{2} \rceil\right)^0$

$2\lceil \frac{K}{2} \rceil$ nodes [at h=2]

$2\lceil \frac{K}{2} \rceil \times \lceil \frac{K}{2} \rceil = 2\lceil \frac{K}{2} \rceil^2$

Total nodes → each node can have min. children → at h=3

at height h

$2\lceil \frac{K}{2} \rceil^{h-1}$

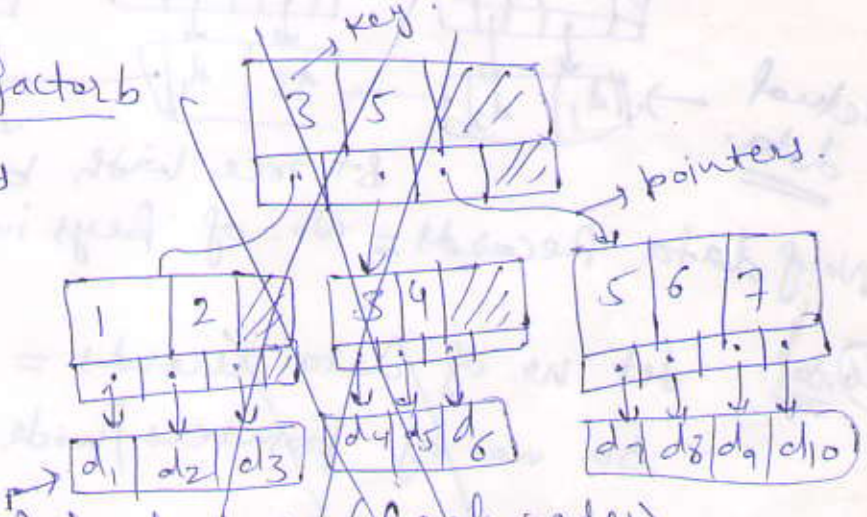
Min bound

June/12 (11)

(26) A B^+ tree index is to be built on the name att. of the relation student. Assume that all student names are of length 8 bytes, disk blocks are of size 512 bytes and index pointers are of size 4 bytes. Given this scenario what would be the best choice of the degree (i.e. no. of pointers/node) of the B^+ tree.

B+ tree → is an n-ary tree having large no. of children per node. A B^+ tree can be viewed as a B-tree in which each node contains only keys (not key-value pairs) and to which an additional level is added at the bottom with linked leaves.

The **order** or **branching factor b** of a B+ tree measures the capacity of nodes (i.e. the no. of children nodes) for internal nodes in the tree.



Node Type	Children Type	Min children	Max children	e.g. $b = 7$
Root Node (only node in tree)	Records.	1	$b-1$	1 – 6
Root node	Internal/Leaf nodes	2	b	2 – 7
Internal node	Internal/Leaf nodes	$\lceil b/2 \rceil$	b	4 – 7
Leaf nodes	Records	$\lceil b/2 \rceil$	$b-1$	4 – 6

No. of pointers (children) = No. of keys + 1

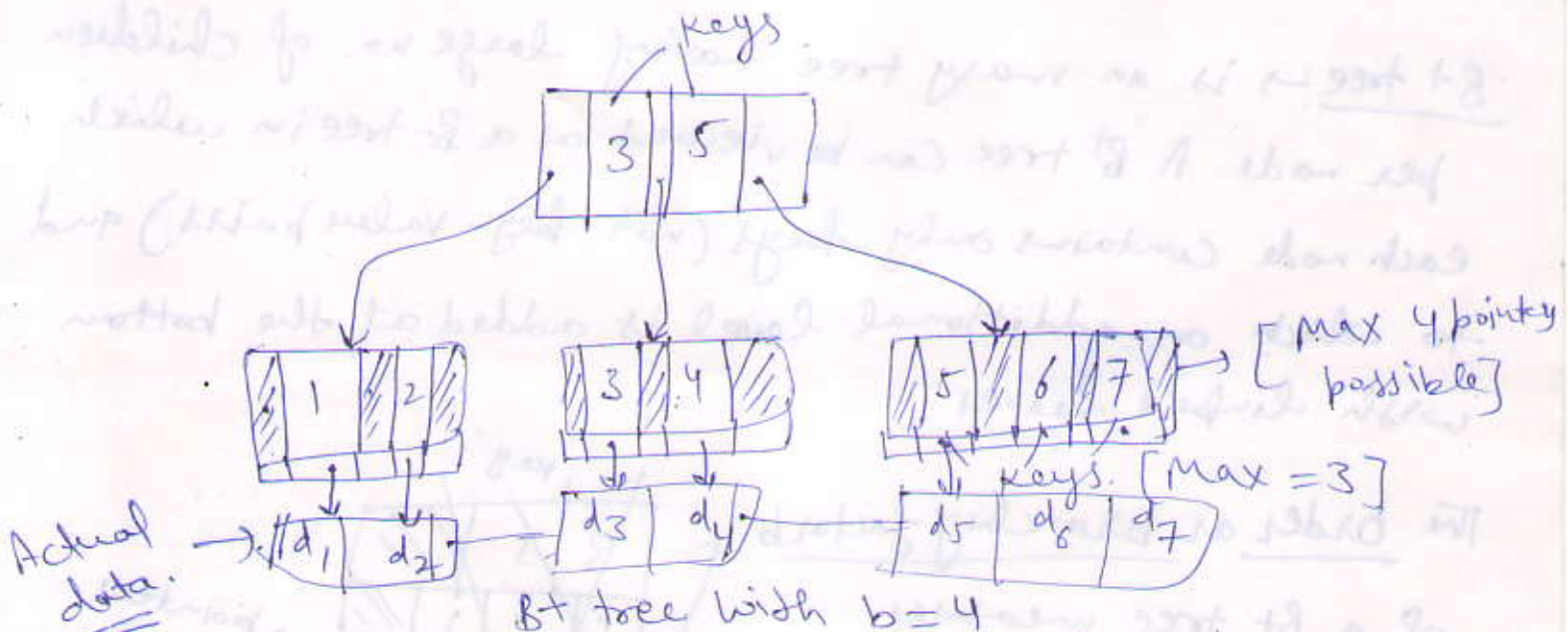
At leaf nodes, keys hold the data.

As. no. of pmax pointers at internal node = b

∴ no. of n keys " " " = b-1

So. no. of " " leaf " = b-1

∴ " " Records " " " = b-1



B+ tree with b = 4

No. of data Records = No. of keys in leaf nodes.

(26) Sol. Let no. of data records = x

So. no. of pointers / node = x+1

$8x + 4(x+1) \leq 512$

$12x = 511 \Rightarrow x = \frac{511}{12} = 42.6 \simeq 43$ Ans

Let b the the degree of b+ tree.

each record needs 8 bytes.

each pointer needs 4 bytes.

~~we have (b-1) records, and (b pointers.~~

$8(b-1) + 4(b) \leq 512 \Rightarrow$

$= 43$