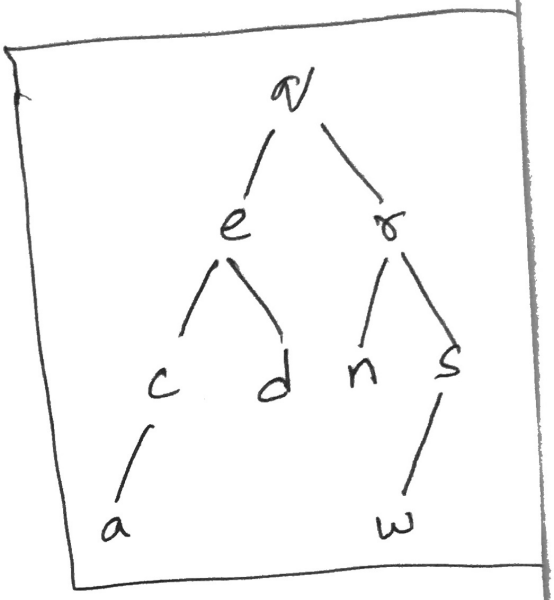
ASSIGNMENT #4

SUBMITTED BY:

NAME : SAI KIRAN TADURI

NET ID : sxt161730

(1) Tree Traversals



(a) An Inorder traversal

a, c, e, d, q, n, r, w, s

(b) A preorder traversal

q, e, c, a, d, r, n, s, w

(c) A post order traversal

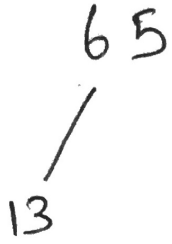
a, c, d, e, n, w, s, r, q

(2) AVL Tree for the following values.

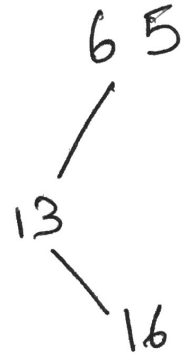
65 13 16 52 28 11 20 14 87 50 26

(i) 65

(ii)

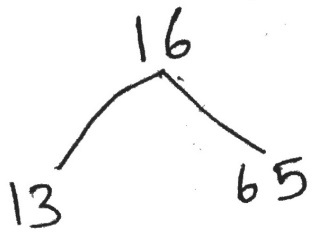


(iii)

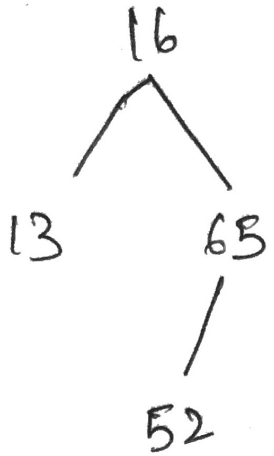


65 is unbalanced. To fix it, we do a double rotation, making 16 the new root. (inside)

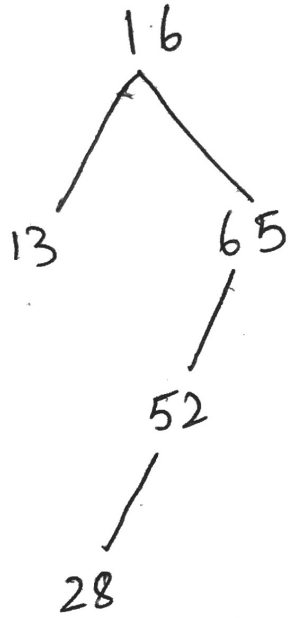
(iv)



(v)

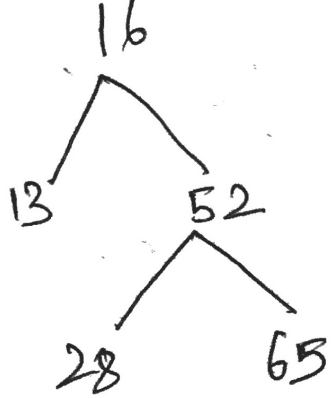


(vi)

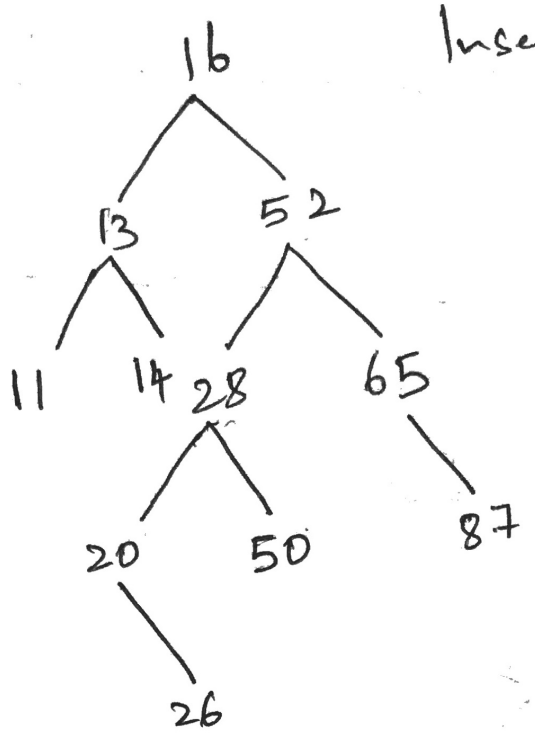


65 is unbalanced. To fix it, we do a single rotation. (outside)

(vii)

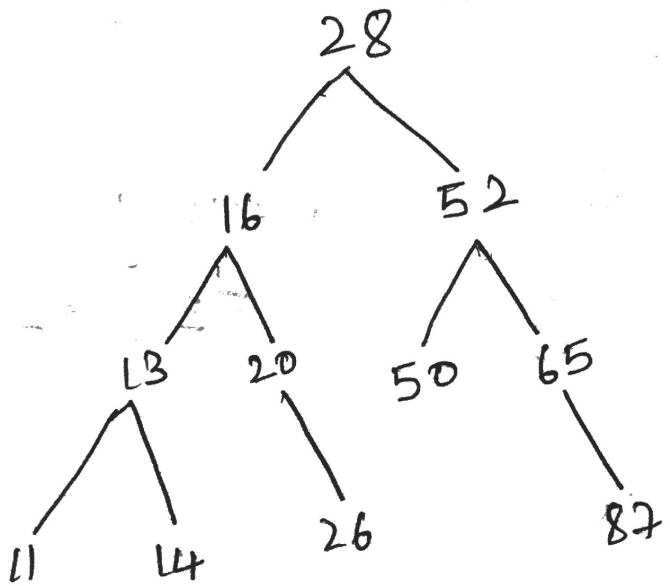


(viii)



Inserting 11, 20, 14, 87, 50 and 26

16 is unbalanced. To fix it, we do a double rotation. (inside)



(3) Draw an AVL tree for the following values

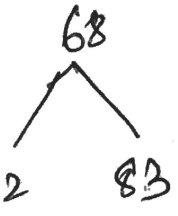
83 12 68 55 32 6 46 57 62

(i)

Inserted 83, 12, 68

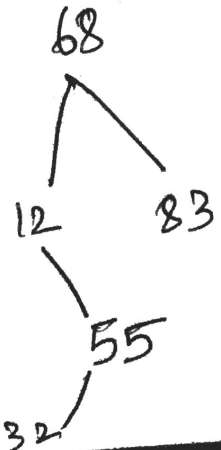
83 is unbalanced. To fix it we do a double rotation, making 68 as the new root. (inside)

(ii)



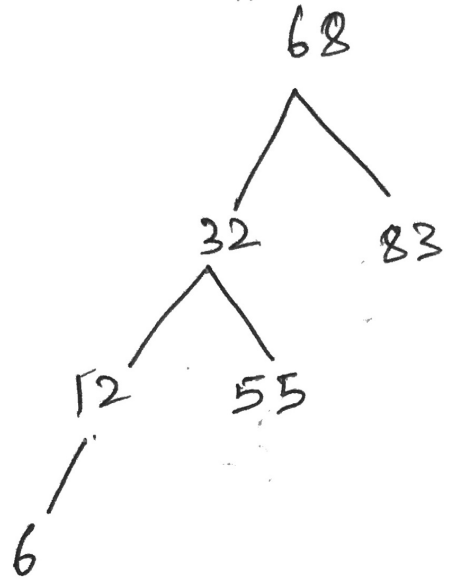
Inserting 55, 32

(iii)



12 is unbalanced. To fix it we do a double rotation. (inside)

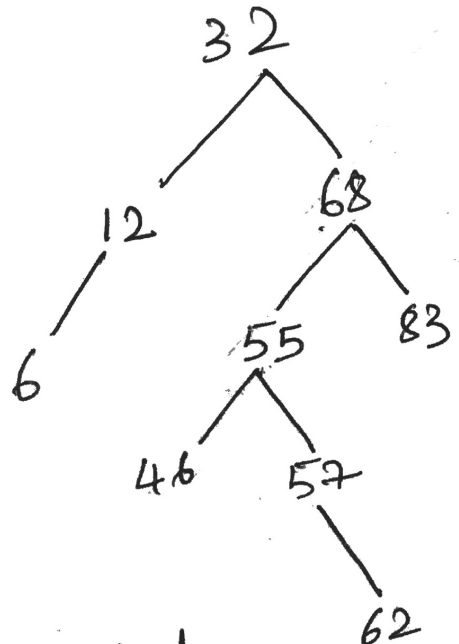
(iv)



Inserting 6

68 is unbalanced. To fix it we do a single rotation (outside)

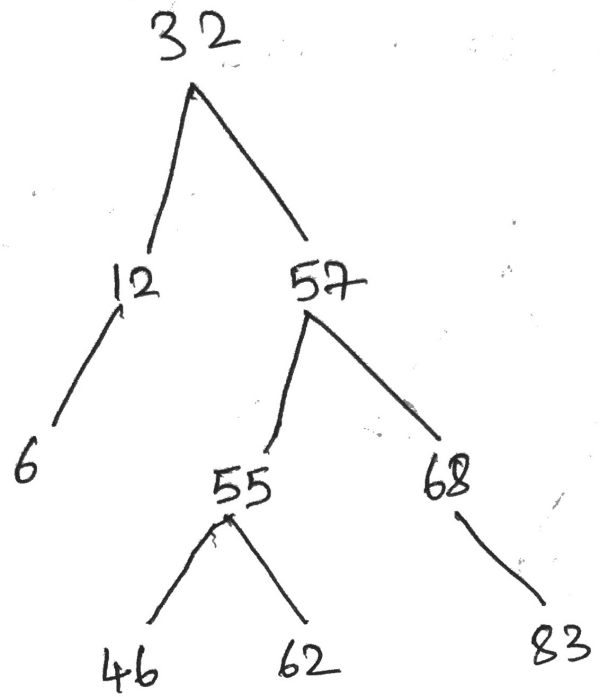
(v)



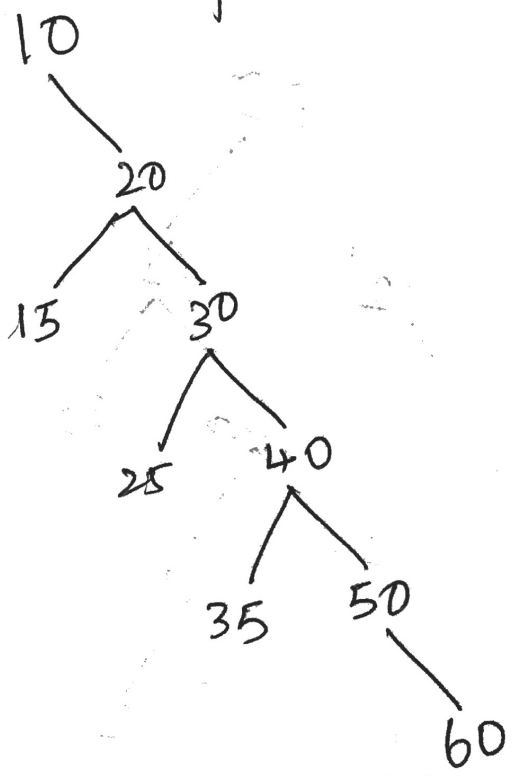
Inserting 46, 57, 62

68 is unbalanced. To fix it we do a double rotation (inside)

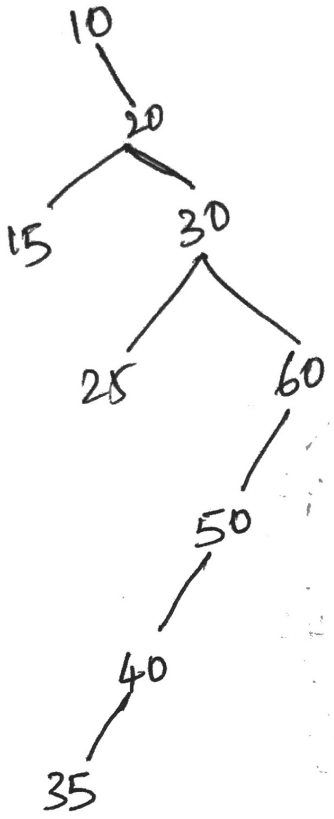
(d)



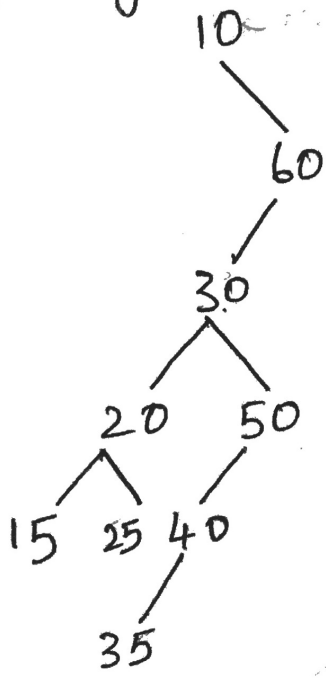
(4) For a splay tree shown below, an access of node 60 is performed.



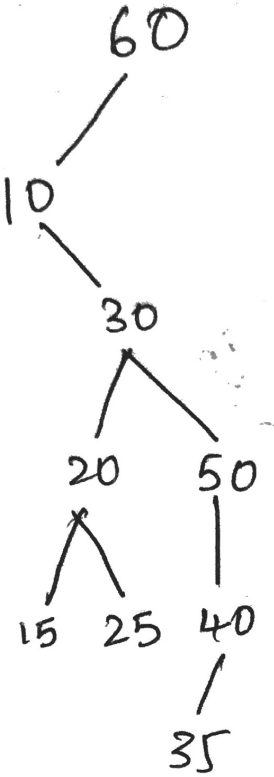
(i) Zig-zig, outside



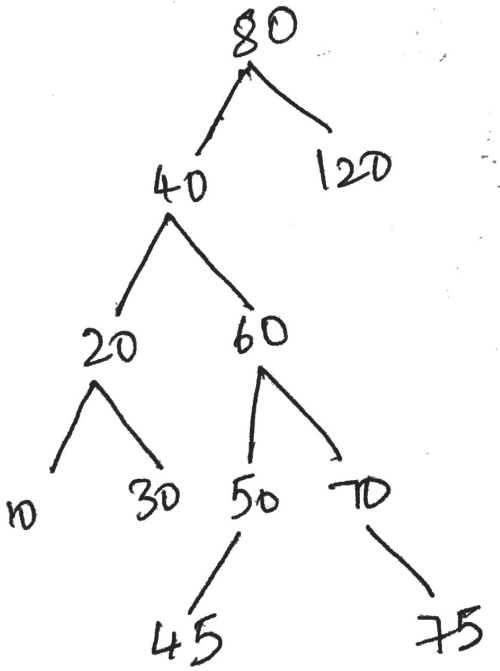
(ii) zig-zig, outside.



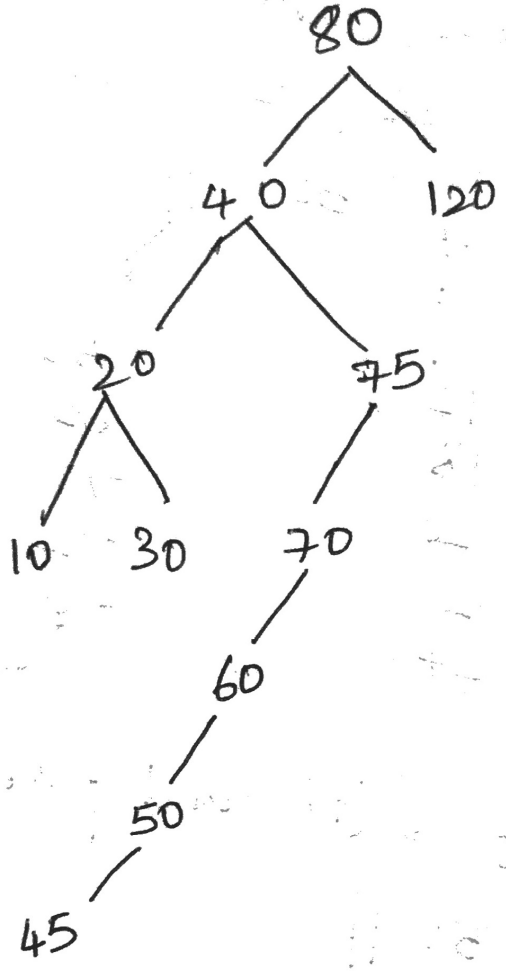
(iii) Child of root, rotate child with root



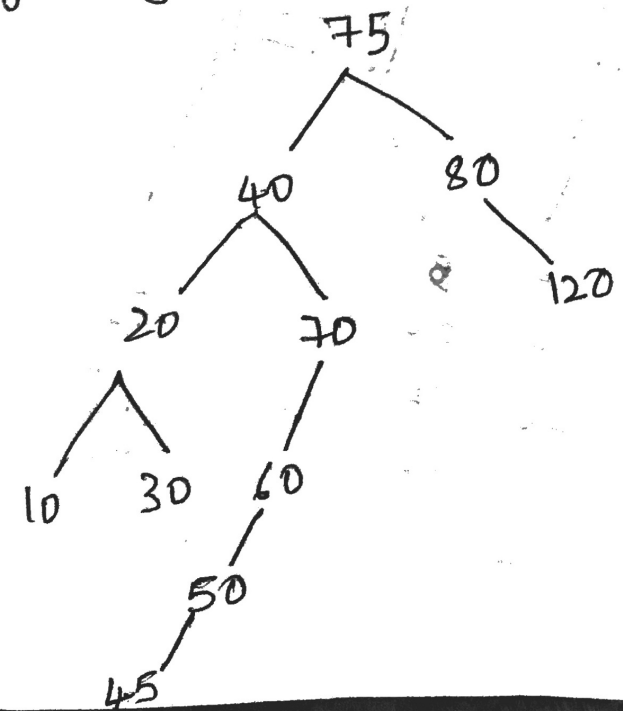
(5) For the splay tree shown below, access of node 75 is performed.



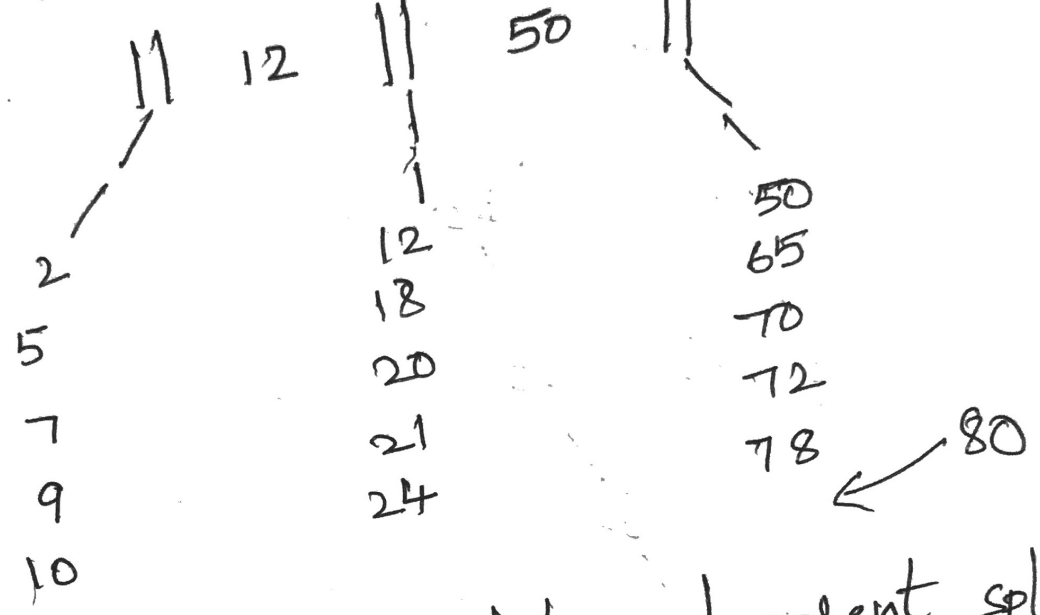
(i) Zig-Zig rotation, outside



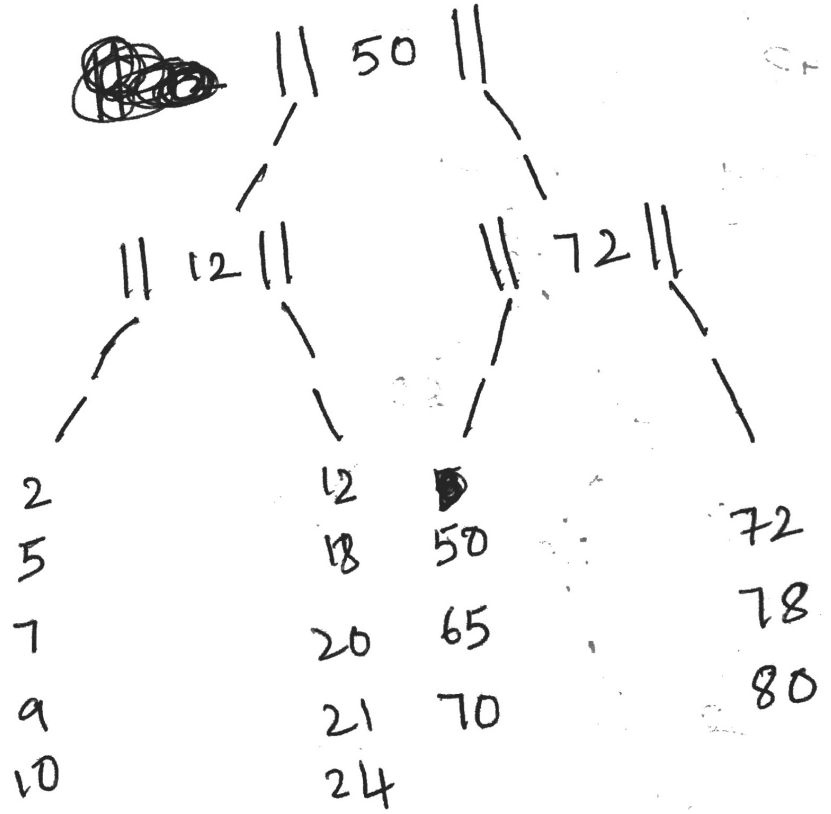
(ii) Zig-Zag, inside



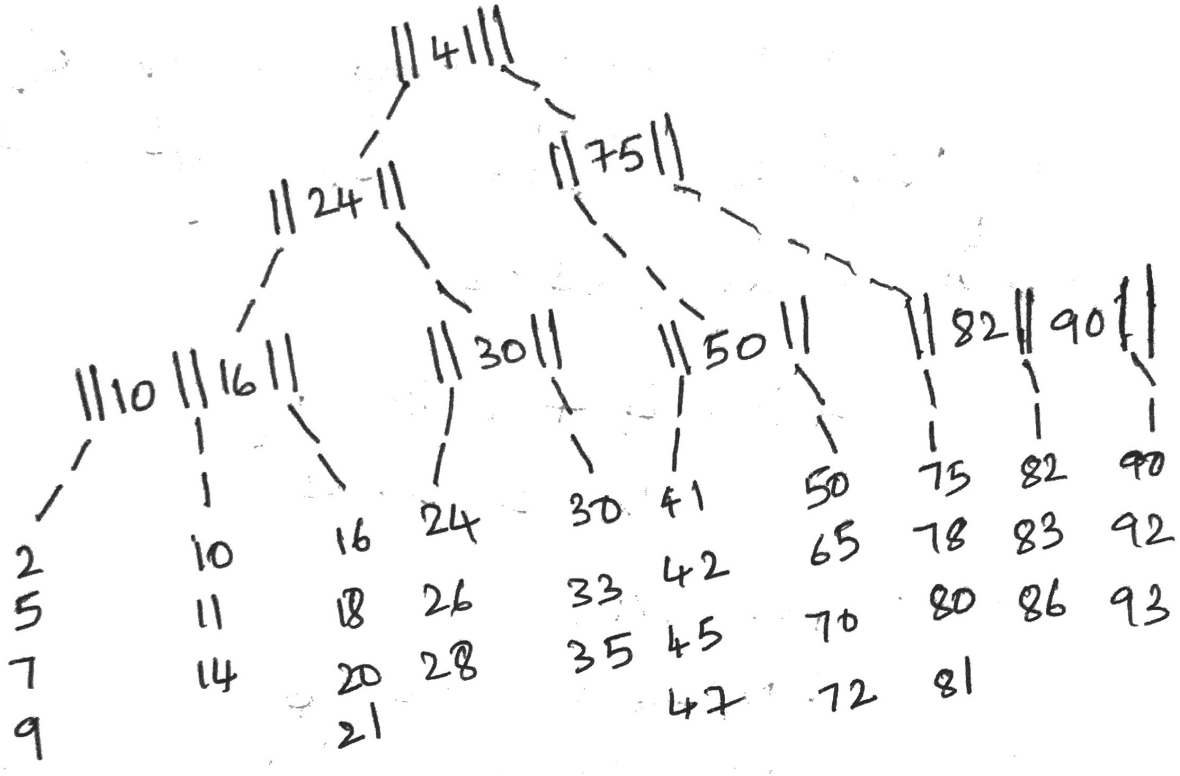
(6) For the B+-tree where M=3 and L=5 shown below, show how an insert value of 80 is handled.



Causes leaf split and parent split



(7) Insert 28 causes leaf and parent split



(8) Block size = 3096 bytes

Data records to be stored are - 36 bytes. and their Key is 4 bytes.

Pointers = 4 bytes.

If each data record is 36 bytes, we could fit 86 records in a block. Thus $\underline{L = 86}$

$$L = \frac{3096}{36} = 86$$

If each key is 4 bytes, there are (M-1) keys and each of the M pointers takes 4 bytes, then each node uses;

$4M - 4 + 4M$ bytes.

$= 8M - 4$ bytes.

Solving for M,

$8M - 4 = 3096$

$M = (3096 + 4) / 8 = 387$

$\underline{M = 387}$

(9) Since each leaf could be half full, 8,600,000 records could take, $\underline{200,000}$ leaves. (8,600,000 / 43).

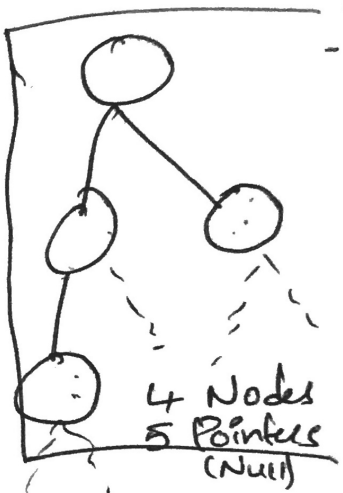
If each internal node branches at least 193 ways, then

$200,000 / 193 = 1036$.

$1036 / 193 = 5 < 193$

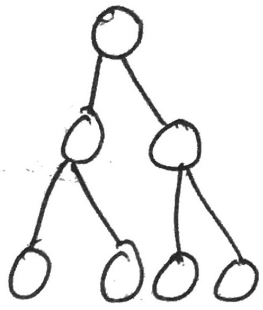
Therefore, the leaves are no deeper than level 4.

(10) Every node in a binary tree has at most 2 pointers. Therefore there are 2N pointers for N nodes. Each node, except the root node, has one incoming pointer from its parent. This accounts for N-1 pointers. Therefore, the remaining are the NULL pointers.

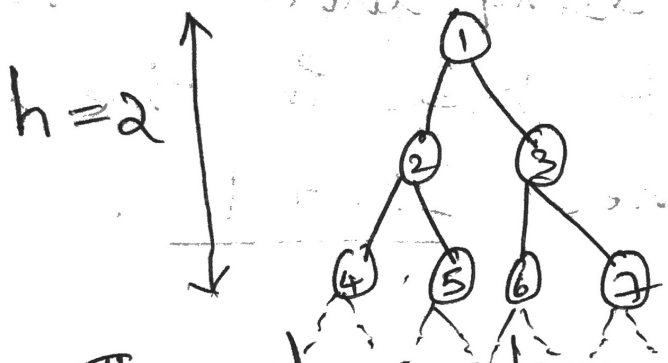


$$2N - (N-1) = \underline{N+1} \text{ null child pointers.}$$

(11) A perfect binary tree is a binary tree where all leaf nodes have the same depth and all other [redacted]. Internal nodes are full nodes.



Consider the following tree,



Number of nodes
$= 2^{h+1} - 1$
$= 2^3 - 1 = 7$

The above tree is a tree with 7 nodes.

In a perfect binary tree (one filled at every level), adding another level increases the height of the tree by 1. and number of nodes in the tree becomes $\underline{2N+1}$

In the above example of height 2, there are 7 nodes. Adding another level, number of nodes in the tree becomes $2(7)+1 = 15$

Therefore, the number of nodes in the tree increases by a factor of 2 i.e., doubles almost.

Therefore, if N is the number of nodes in a tree (perfect binary tree), then by adding another level the number of nodes in the tree becomes $2N + 1$.

Also, the number of nodes in a tree, if another level is added, it increases by 2^{h+1} Nodes where h is the height of the perfect binary tree.

Let 'h' be the height of the perfect binary tree. By adding another level, the number of nodes increases by 2^{h+1}. Therefore, total nodes becomes,

$$N + 2^{h+1}.$$