1) Give the sequence of letters for each traversal of this binary tree

(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 postorder traversal

a → c → d → e → n → w → s → r → q

2) It is located below

3) It is located below

4) It is located below

5) It is located below

6) It is located below

7) It is located below

8) B+-tree is to be stored on disk whose block size is 3096 bytes. The data records to be stored are 36 bytes, and their key is 4 bytes. Determine the values for M and L for the B+-tree. Assume pointers are 4 bytes each

The value for M)

4(M-1) → keys

4M pointers

4(M-1) + 4M = 8M-4

So, 8M = 3096 + 4→ 8M = 3100 → M = 3100/8 → M = 387

The value for L)

L = 3096/36 = 86 → L = 86

9) For the problem above, how many levels are needed to store 8,600,000 records?

If each internal node branches at least 193 ways, then the leaves are no deeper than level 4 because

8,600,000 / 193 = 44,599.5855 → 44,559.5855 / 193 = 203.878681 → 203.878681 / 193 = 1.1962626. and + 1 level.

10) If a binary tree has N nodes, how many null child pointers will it have?

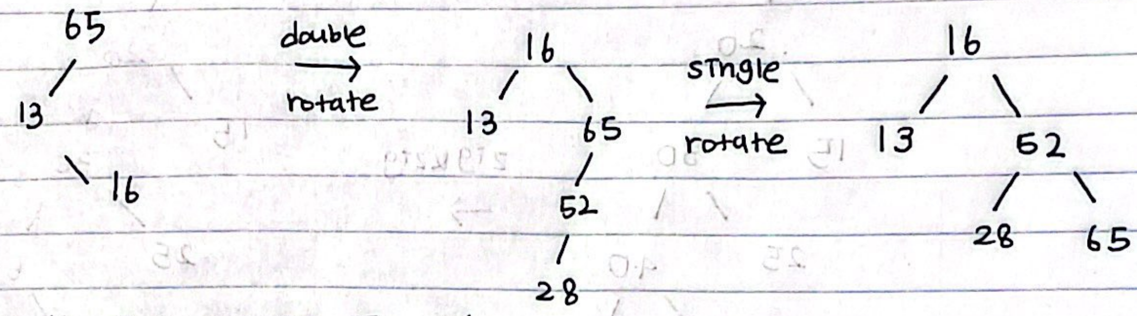
- If a binary tree has N nodes, the total pointers will be 2N because each node has 2 pointers. However, the occupied child pointesrs is N-1 because root doesn't have the parent. So, 2N – (N-1) = N+1

11) In a perfect binary tree (one filled at every level), what does adding another level do to the number of nodes in the tree?

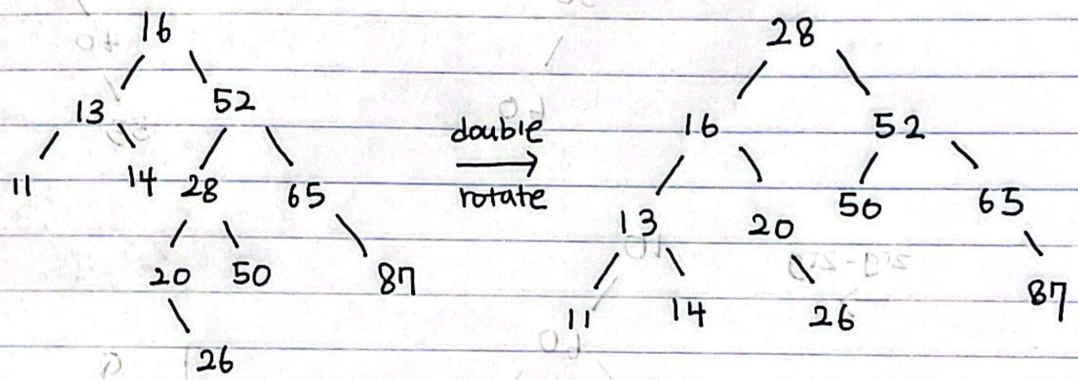
- In a perfect binary tree, 2^k(k is the level) is added to the existed node when adding another level. This is because the perfect binary tree $2^k - 1$ nodes before adding kth level (so k-1 level →$2^k - 1$), but before adding another lever, then the nodes become $2^{k+1} - 1$ → $2^{k+1} - 1 - (2^k - 1) = 2^k(2-1) = 2^k$. So, adding another level makes the node increase as much as 2^k

② ~~65~~. ~~13~~. ~~16~~. ~~52~~. ~~28~~. ~~11~~. ~~20~~. ~~14~~. ~~87~~. ~~50~~. ~~26~~

add 65. 13. 16 add 52. 28



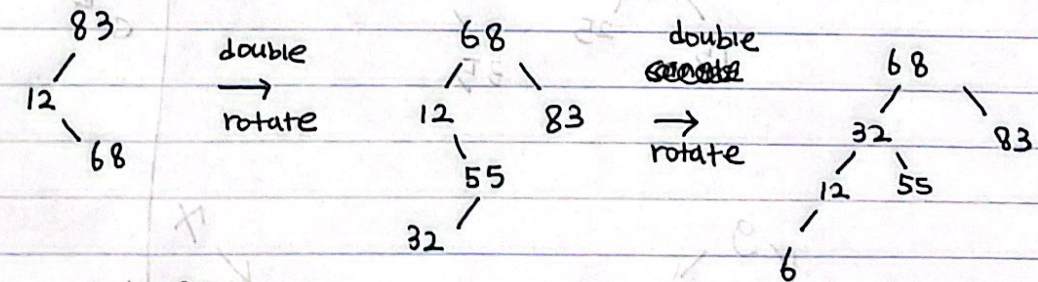
add 11, 20, 14, 87, 50, 26

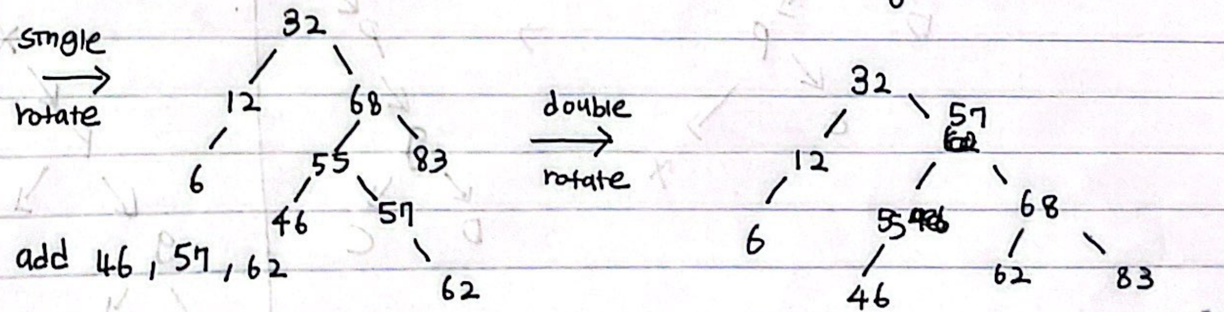


③ ~~83~~. ~~12~~. ~~68~~. ~~55~~. ~~32~~. ~~6~~. ~~46~~. ~~57~~. ~~62~~

add 83. 12. 68 add 55

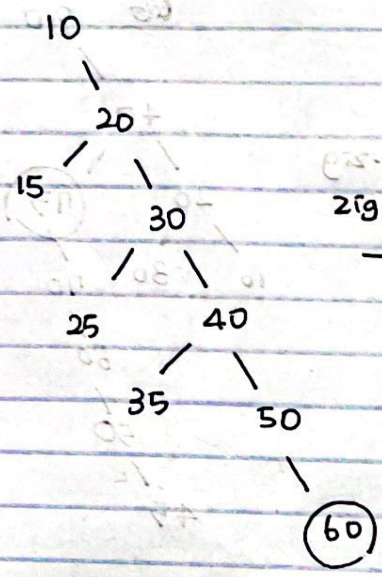
add 6



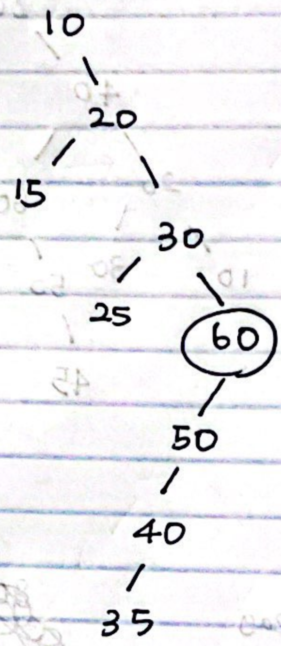


add 46, 57, 62

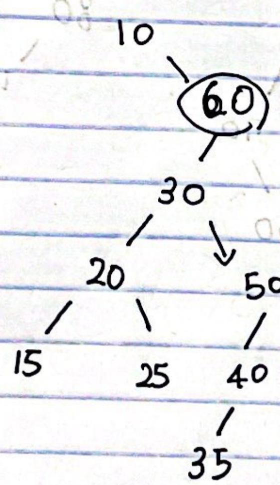
(a) (4)



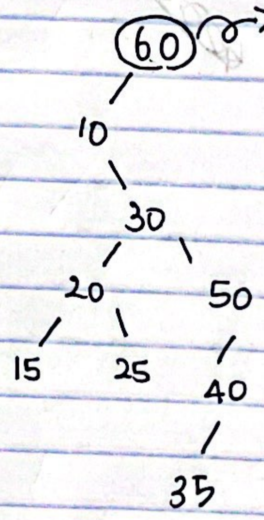
zig-zag →



zig-zig →

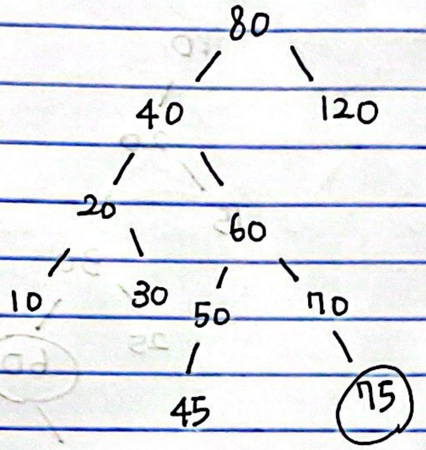


→

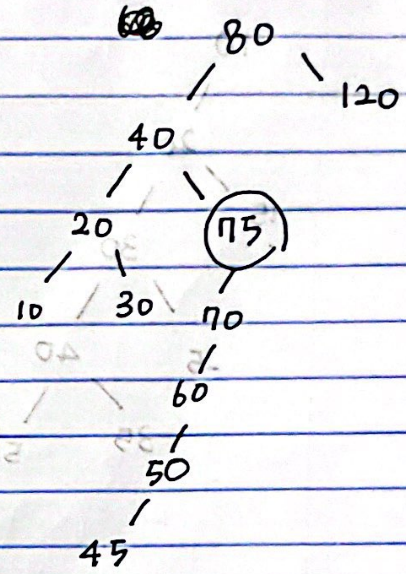


→ 60 became root node

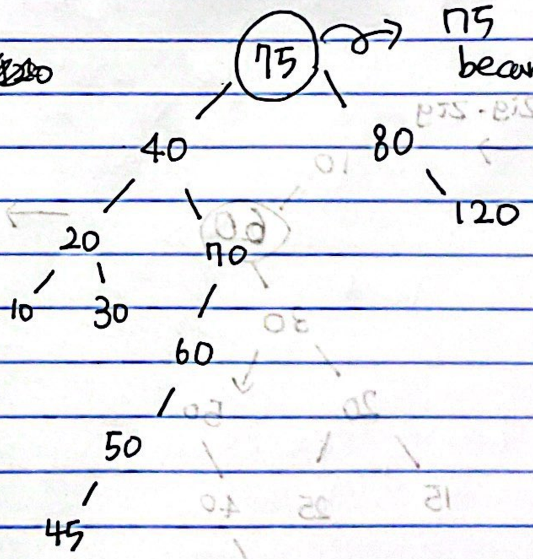
5



Zig-zig →

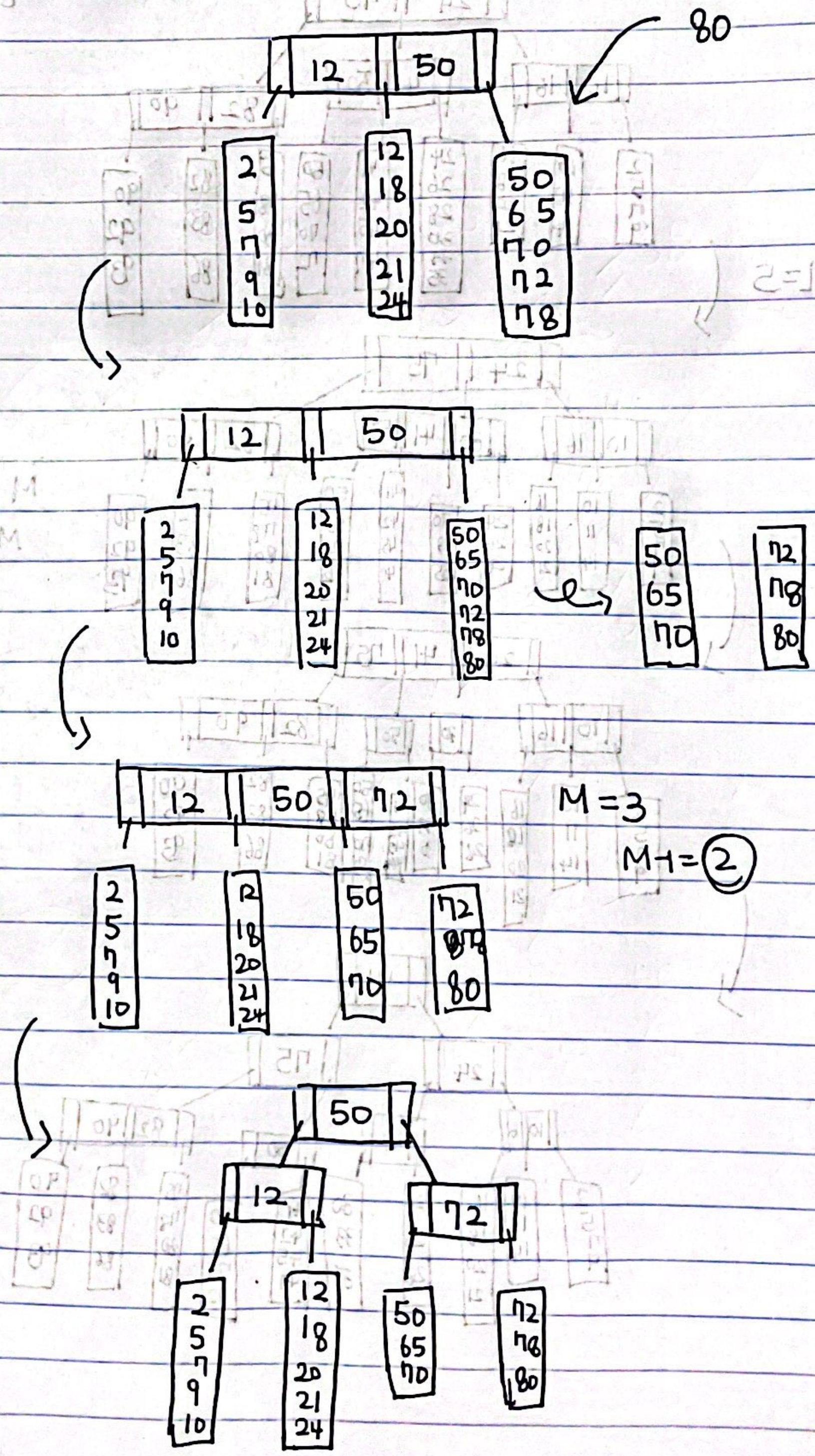


Zig-Zag →



75 became root node

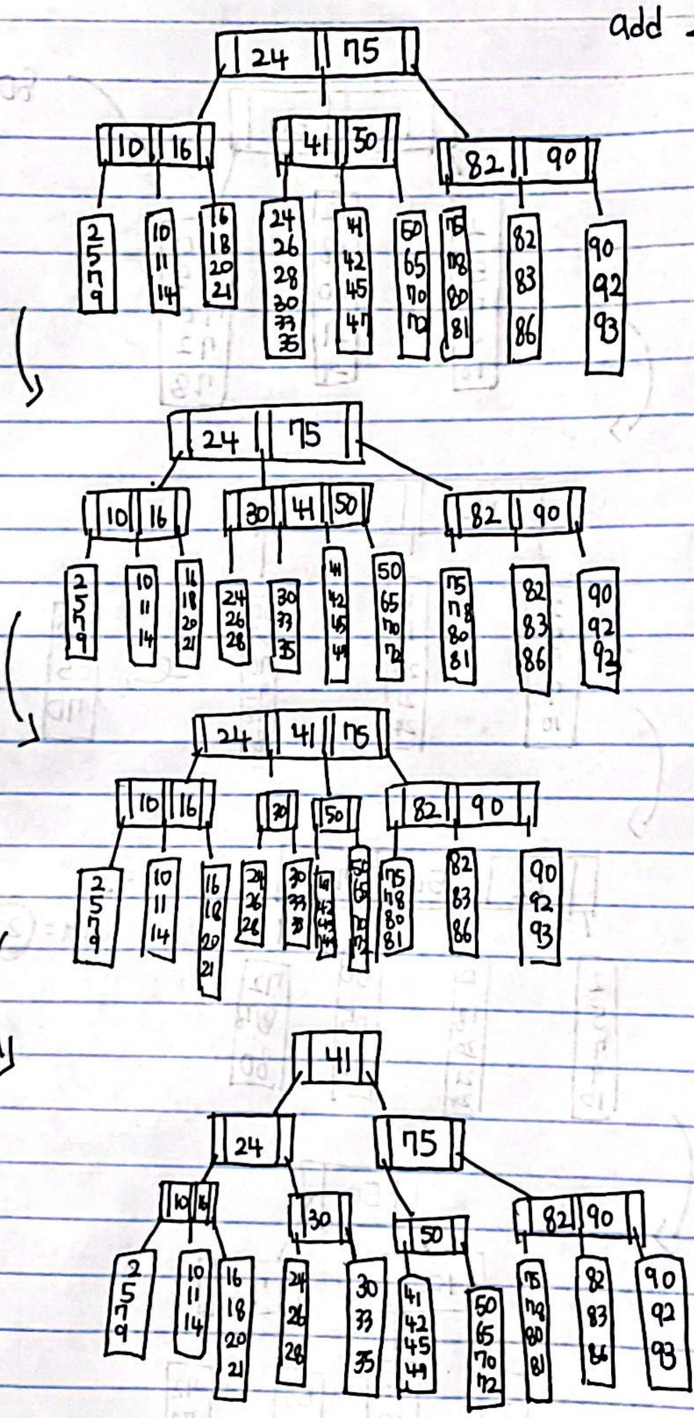
6


80
12 | 50
2 5 7 9 10
12 18 20 21 24
50 65 70 72 78
12 | 50
2 5 7 9 10
12 18 20 21 24
50 65 70 72 78 80
50 65 70
72 78 80
12 | 50 | 72
M=3
M-1=2
2 5 7 9 10
12 18 20 21 24
50 65 70
72 78 80
50
12
72
2 5 7 9 10
12 18 20 21 24
50 65 70
72 78 80

7

add 28

L=5



M=3

M-1=2