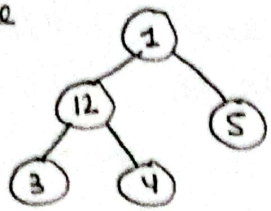
(10) Binary Tree | Binary Search Tree:

Binary tree can have only two child nodes.

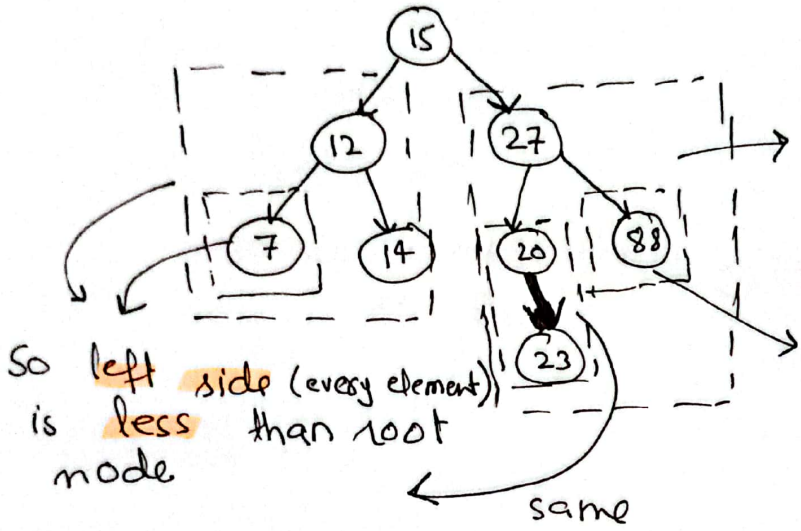
like



→ (3) & (4) are leaf nodes.

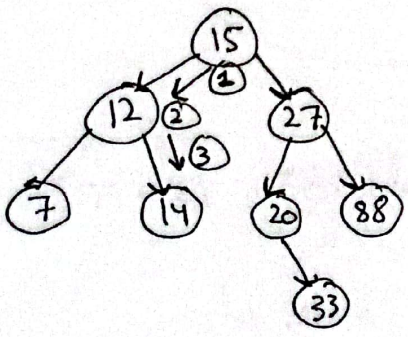
→ Binary Search Tree is the special case of Binary Tree where elements have some kind of order.

e.g



↳ Another property of BST is that element are always unique.

→ So we use BST to implement set() data structure - That is similar to list but does not allows unique elements.

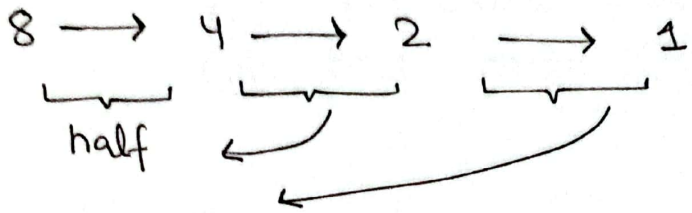


→ If you want to search 14 :

① → start with 15 (root)
② → As 14 < 15 → go left
③ → Reached 12
④ → 14 > 12 go right
⑤ → There you have it

→ So if you store these element in list your complexity will be
O(n) → Linear
(and here you are eliminating half of element.

e.g If n=8 (nodes)

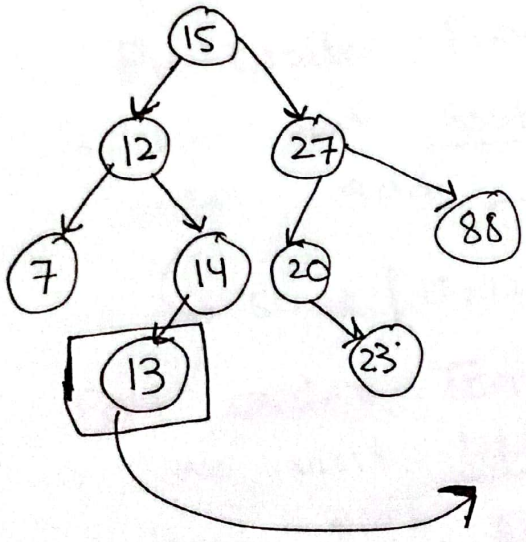


↳ 3 iterations

$\log_2 8 = 3$

search complexity of BST = O(log n)

Inserting :



→ Insert (13)
① Start with root
② 13 < 15 → Go left
③ You are at 12
④ 13 > 12 → go right
⑤ You are at 14
⑥ 13 < 14 → go left
⑦ Empty → insert there

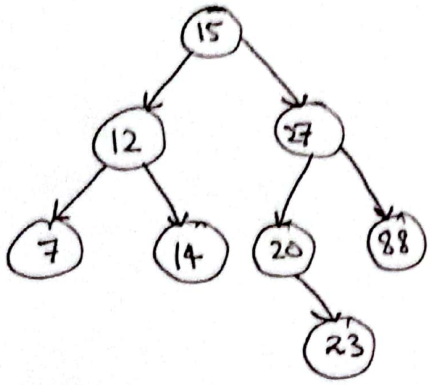
→ ∴
Insert complexity = O(log n)
↳ same 1/2

Traversal Techniques

↳ Techniques used to look for a element in Binary Tree. (moving in BT)

1. Breadth first search
2. Depth first search
   - (i) In order Traversal
   - (ii) Pre order Traversal
   - (iii) Post order Traversal

(i) In order Traversal:



→ so in in this Technique our root Node stay in order. i.e we visit left side first. Then root and then right side.

↳ so ⇒ [7, 12, 14, 15, 20, 23, 27, 88]

(ii) Pre order Traversal:

→ we visit Roots first then left side and In end righ side.

↳ so ⇒ [15, 12, 7, 14, 27, 20, 23, 88]

(iii) Post order Traversal:

→ we visit left side first, then righ side and in end we visit root

↳ so ⇒ [7, 14, 12, 23, 20, 88, 27, 15]