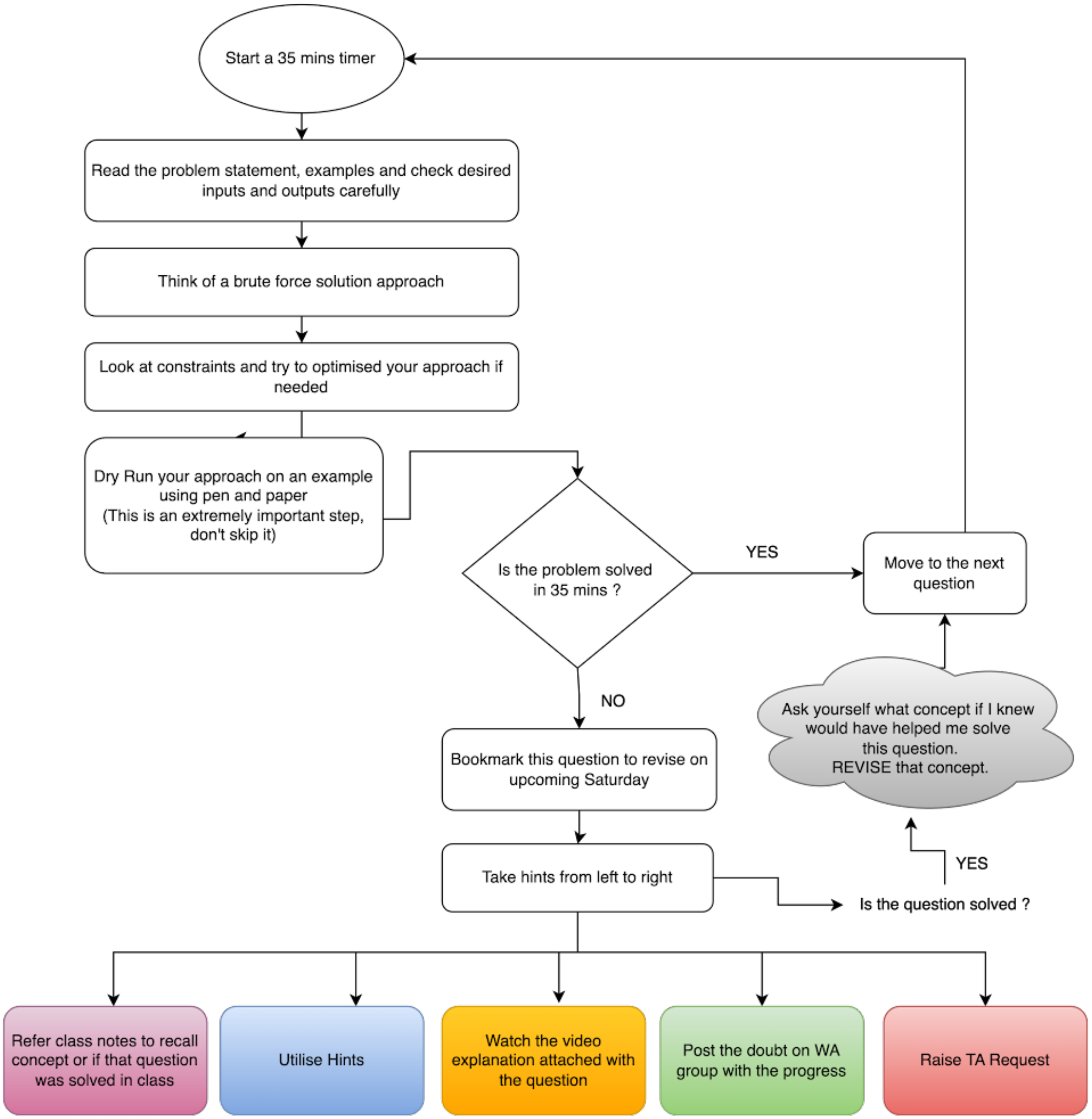
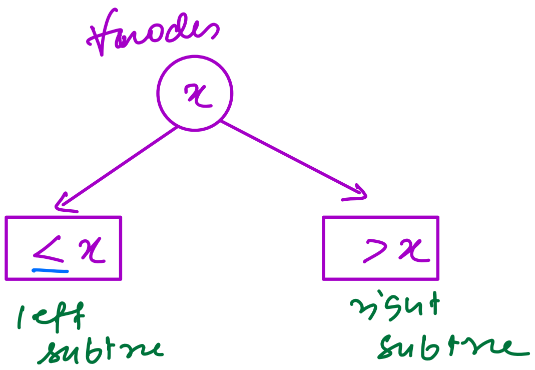
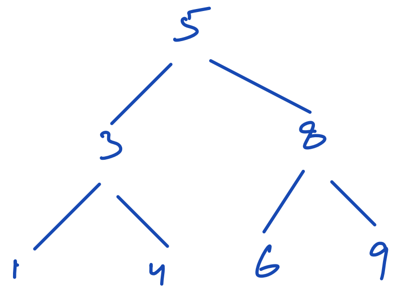
Trees 3 : BST

How to solve a problem ?


Start a 35 mins timer
Read the problem statement, examples and check desired inputs and outputs carefully
Think of a brute force solution approach
Look at constraints and try to optimised your approach if needed
Dry Run your approach on an example using pen and paper (This is an extremely important step, don't skip it)
Is the problem solved in 35 mins ?
YES
Move to the next question
NO
Bookmark this question to revise on upcoming Saturday
Ask yourself what concept if I knew would have helped me solve this question. REVISE that concept.
YES
Take hints from left to right
Is the question solved ?
Refer class notes to recall concept or if that question was solved in class
Utilise Hints
Watch the video explanation attached with the question
Post the doubt on WA group with the progress
Raise TA Request

Binary Search Tree

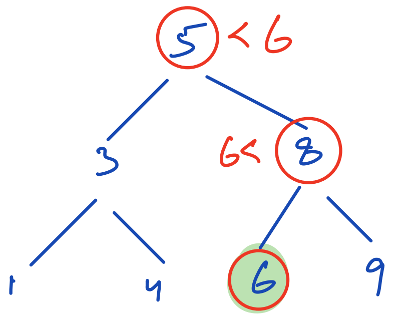




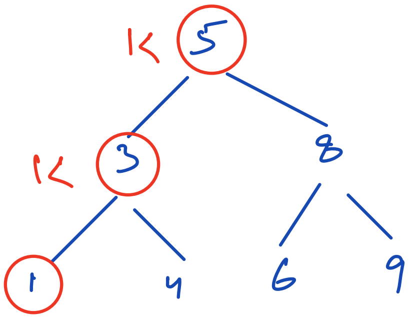
all data in left subtree $\leq x$

all data in right subtree $> x$

Searching



find (6) → true



Total no. of nodes visited to find (1) ?

ans = 3

Code

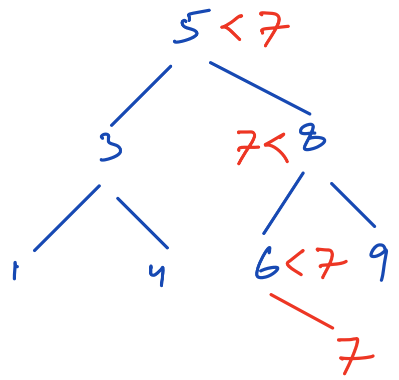
```
Node  search( root, target) {
    if( root == NULL)   return NULL

    if( root.data == target)
        return root

    if( target < root.data) {
        return search ( root.left, target)
    }
    return search( root.right, target)
}
```

TC = O(H)

SC = O(H)

Insertion → search + insert

insert(7)

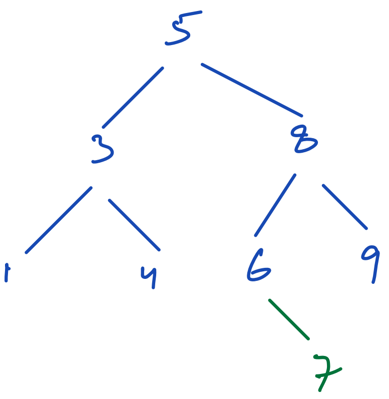


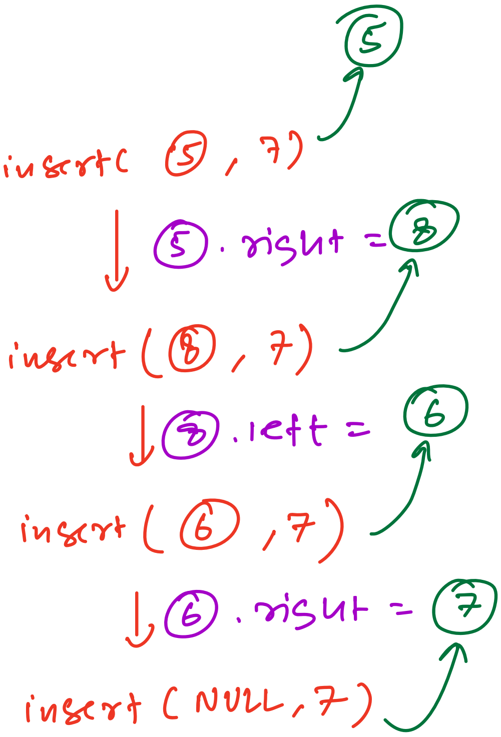
Code

```
Node insert(root, value){
    if (root == NULL)
        return new Node(value)

    if( value <= root.data){
        root.left = insert(root.left, value)

    }
    else{
        root.right = insert(root.right, value)

    }
    return root

}
```

TC = O(H)
SC = O(H)

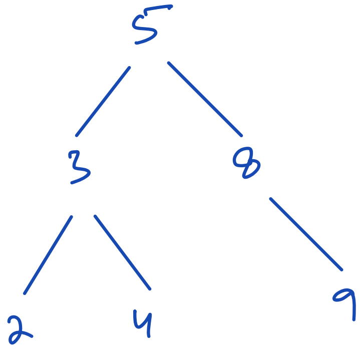
Dry run





Ques find smallest element in BST ?

left most node in BST



code

```
if (root == NULL) return -1    invalid value
temp = root
while( temp.left != NULL) {
    temp = temp.left
}
return temp.data
```

TC = O(H)

SC = O(1)

Ques → find the largest element in BST ?

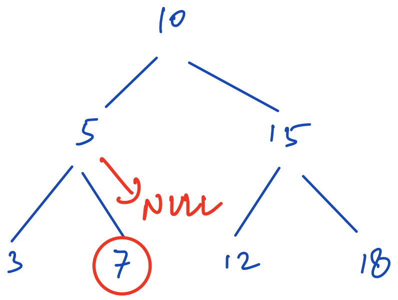
right most node in BST

TODO → code

Deletion in BST

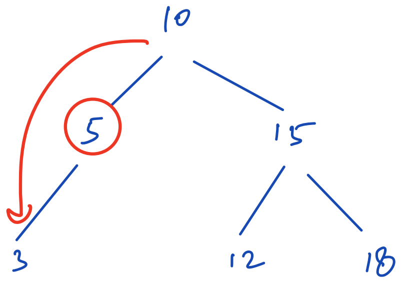
Case 1 : delete leaf node
(Node with 0 child)

update the link of parent
to NULL

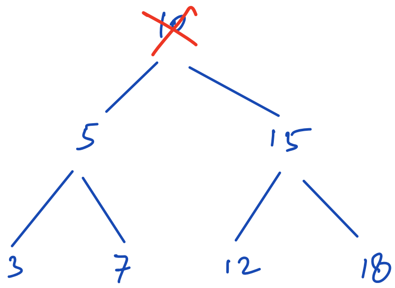


Case 2 : delete node with
1 child

update the link of parent
to the only child



Case 3 : delete node with
2 children



1. replace the node with
its inorder predecessor or successor.

predecessor ↓ largest element in left subtree

successor ↓ smallest element in right subtree

2. delete the replaced node as case 1 or case 2.

Code

```
Node delete( root, K) {
    if( root == NULL)    return NULL
    if( K < root.data) {
        root.left = delete( root.left ,K)
    }
    else if ( K > root.data) {
        root.right = delete ( root.right ,K)

    }
    else {    // perform deletion
        if( root.left == NULL && root.right == NULL) { //Case 1
            return NULL
        }
        else if( root.left == NULL || root.right == NULL ) { // Case 2
            if( root.left != NULL)
                return root.left
            return root.right
        }
        else {    // Case 3
            Node temp = root.left;
            while( temp. right != NULL) {
                temp= temp.right
            }
            root. data = temp. data
            root. left = delete ( root. left, temp. data)
        }
    }
    return root
}
```

$TC = O(H) + \text{either } \underbrace{O(1)}_{\text{case 1,2}} \text{ or } \underbrace{O(H)}_{\text{case3}}$

$TC = O(H)$

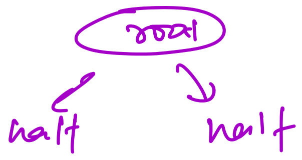
$SC = O(H)$

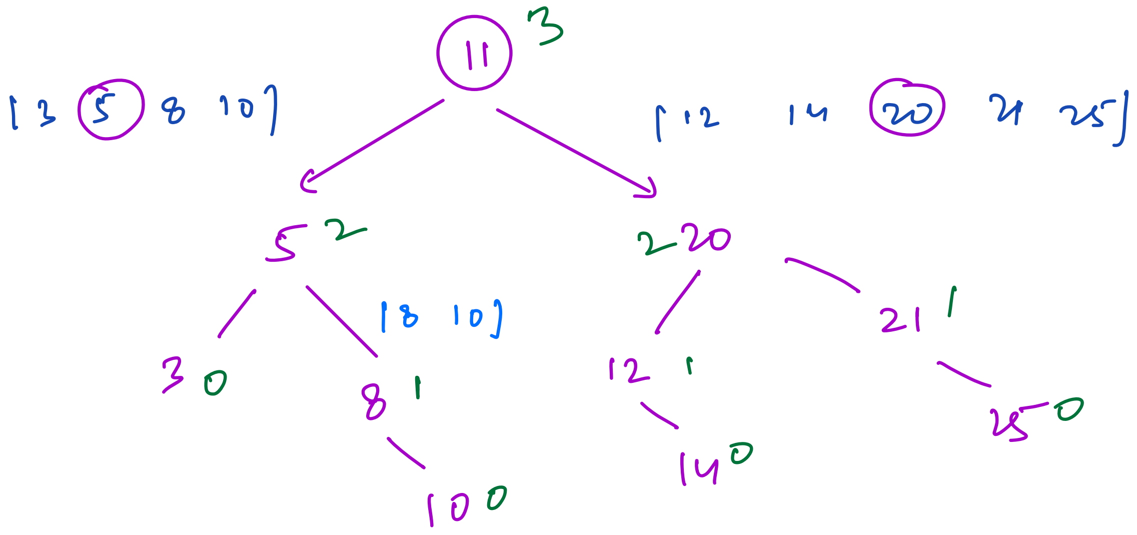
Question

|weight of left - weight of right| <=1

Construct a weight balanced BST from sorted array

eg $A = \left[\begin{array}{cccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 3 & 5 & 8 & 10 & \boxed{11} & 12 & 14 & 20 & 21 & 25 \end{array}\right]$





Code

```
                     0    n-1
Node build ( A, l, r) {
    if ( l > r) return NULL
    mid = (l+r)/2      or   l+(r-l)/2
    root = new Node( A[mid])
    root.left = build( A, l, mid-1)
    root.right = build (A, mid+1, r)

    return root
}
```

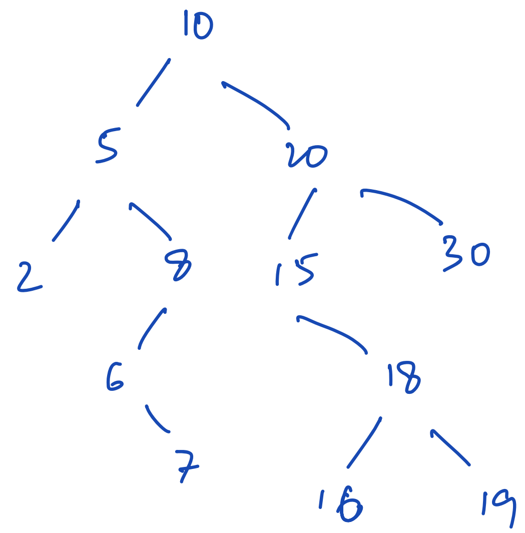
$TC = O(N)$

$SC = O(\log N)$

$H = \log N$

Ques

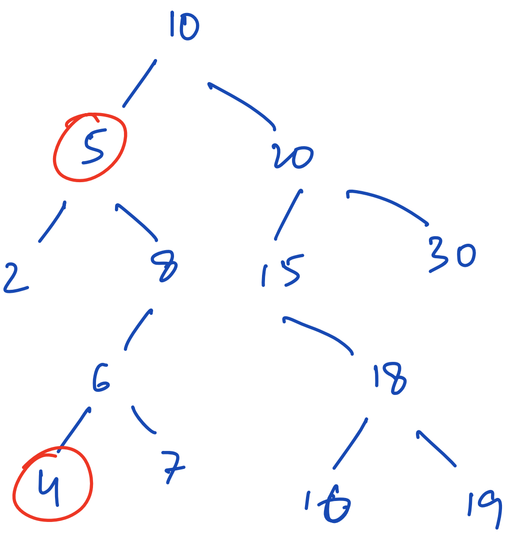
Check if the given binary tree is BST ?



ans = true

inorder

2 5 6 7 8 10 15 16 18 19 20 30

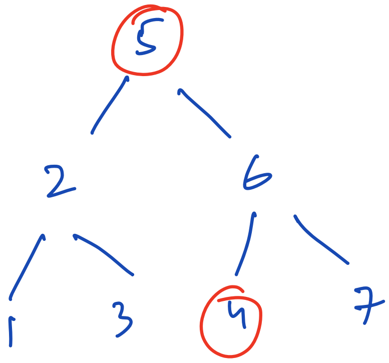


inorder

2 5 4 6 7 8 10 15

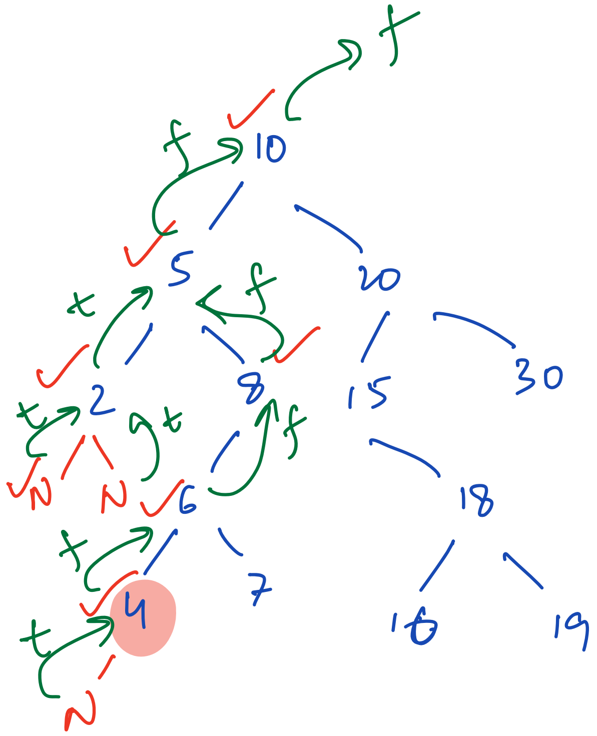
16 18 19 20 30

ans = false



Code

```
    int prev = -INF      // min. value possible

  bool isBST( root ) {

      if ( root == NULL)    return true

      if( isBST( root.left) ==false)      Left
          return false

      if ( root.data < prev) {        Node
            return false
      }

      prev = root.data

      return  isBST( root.right)      Right

  }
```



prev = - ~~INF~~

~~2~~ 5