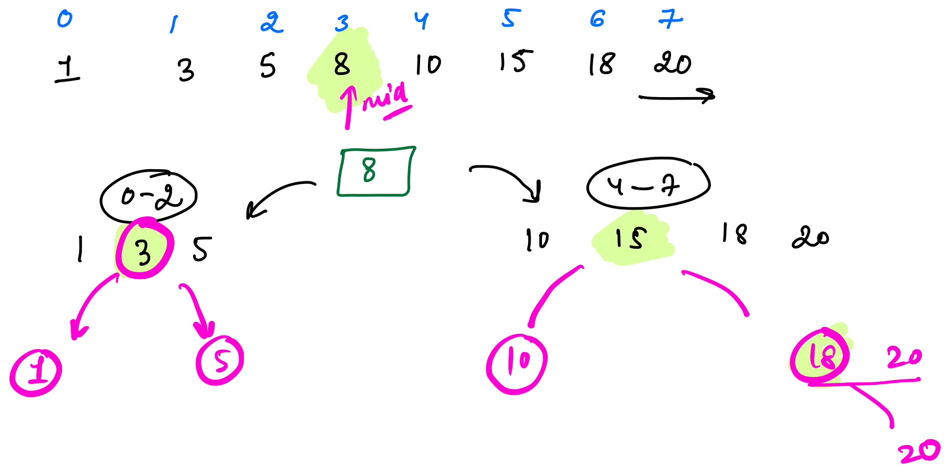
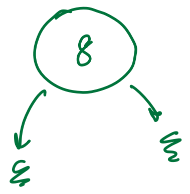
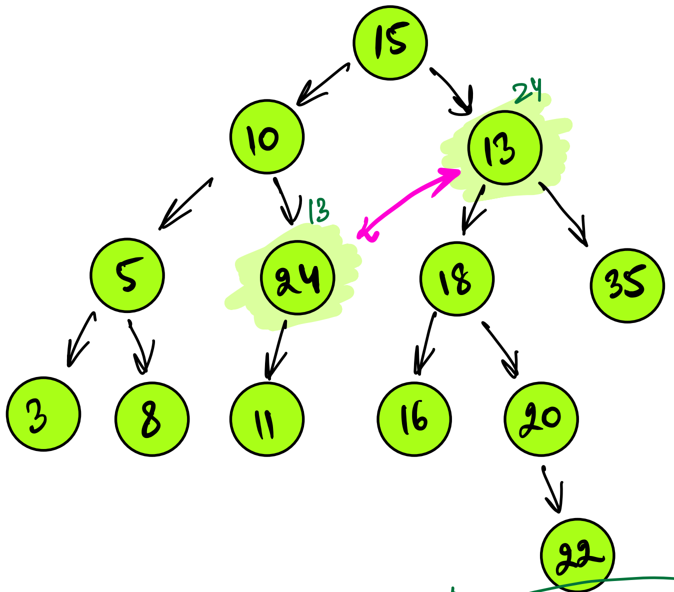
- Given sorted array. Convert it into balanced BST.

∀ nodes $|\text{height of left child} - \text{height of right child}| <= 1$

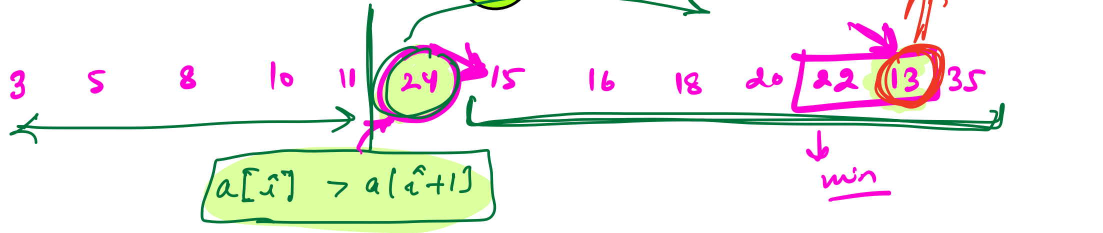


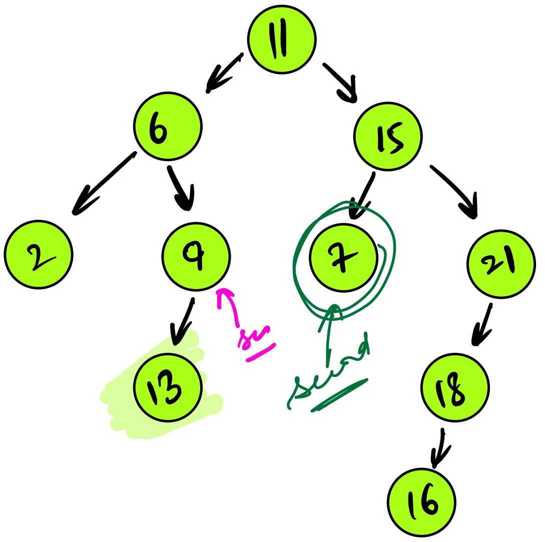
```
Node build ( arr[], start, end )
{
        if ( start > end ) return NULL;

        int mid = (start + end)/2;

    Node temp = new Node (arr[mid]);

   temp.left = build( arr, start, mid-1);
   temp.right = build( arr, mid+1, end);

      return temp;
}
```

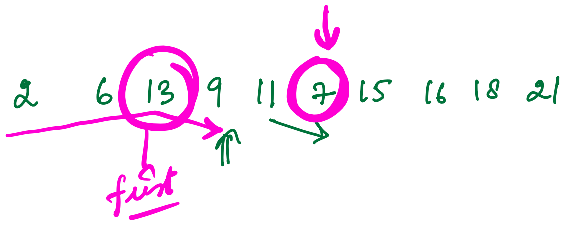


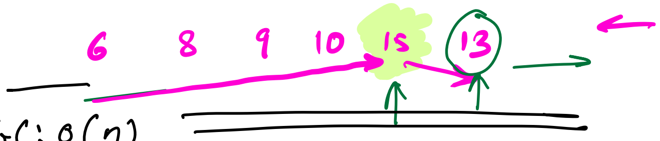


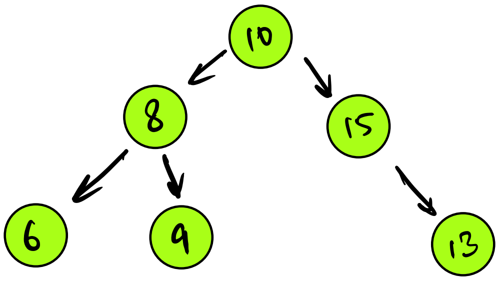
tree (almost BST)
only two nodes
are swapped.
Identify two
swapped nodes
⇓
Swap the data &
return the proper BST











T.C : O(n)

S.C : O(n)

O(~~logn~~)

```
prev=null, first=null, second=null;
void findswap( root )
{
        if(root== null) return;
        findswap( root.left);

        if( prev != null  && prev.data > root.data)
        {
            if( first==null) first=prev;
              second = root;
        }
        prev= root;
         findswap( root.right);
}
```

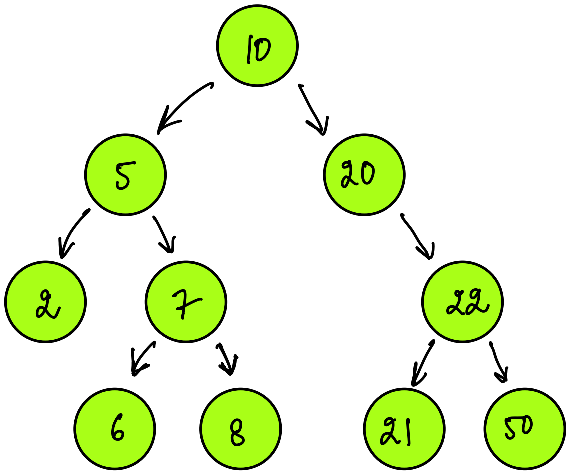
T.C: O(n)

S.C: O(n)

O(logn) _BBST

```
int   temp= first.data;
first.data  = second.data;
second.data = temp;
```

3. K^{th} element in BST



K	
3	6
5	8
10	50

$n + q$

T.C: O(n)

S.C: O(n)

A1: store the inorder & arr[K-1]

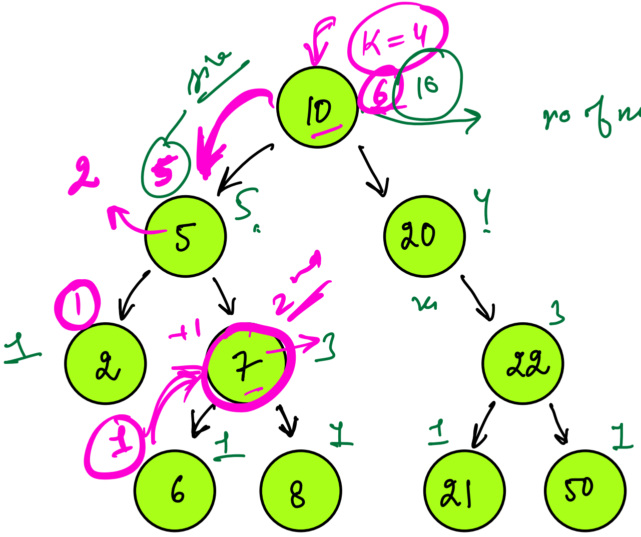
store inorder

while peering the inorder — K^{th} element

T.C: O(n)

S.C: O(H)

$q*n$



no of nodes in LST +1

1

size at every node

T.C: O(H)

$q*H$

S.C: O(n)

size calculate
(
precomputation

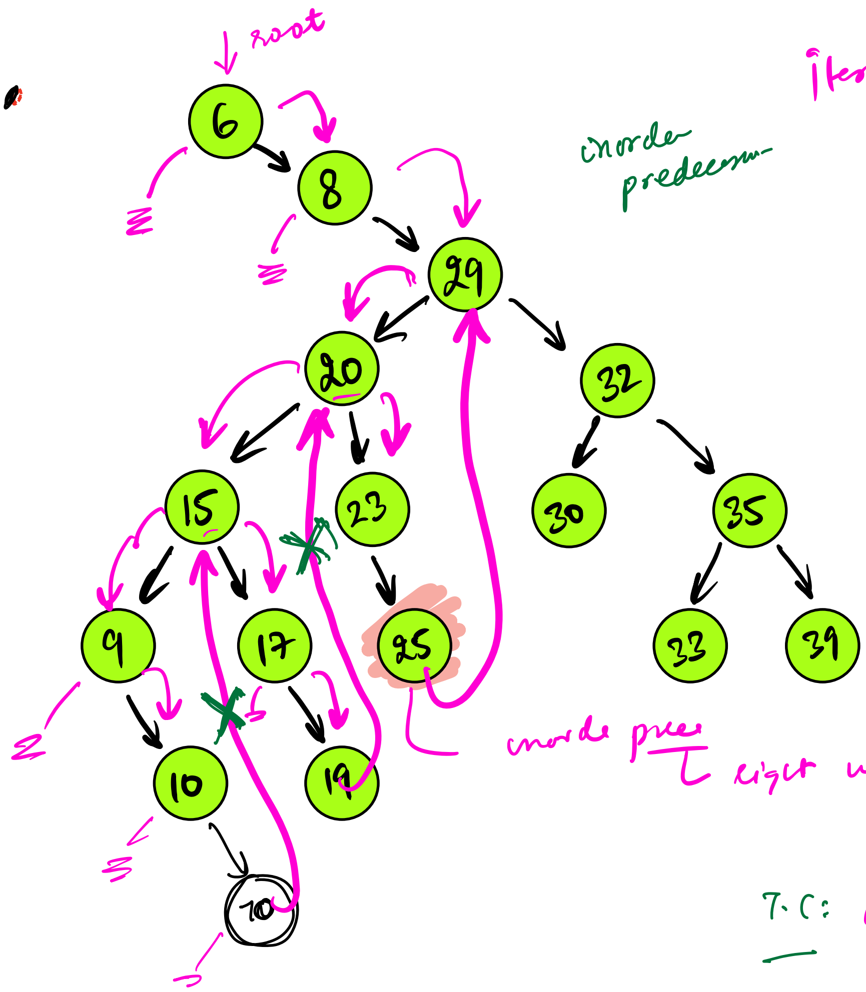
size of LST

Trees - I

store the size in HM

```
if( sizel == K-1)
        return root;
else if( sizel > K-1)
{
        root = root.left;
}
else
{
        K = K-1 - sizel;
        root = root.right;
}
```

$O(n + q*H)$



inorder pred
⊏ right will also null

inorder predecessor

Iterative / recursive

S·C: O(~~H~~)

O(1)

T·C: O(N)

left, right

6 8 9 10 70 15
17 19 20

T·C: O(3n) = O(n)

Morris inorder

```
        curr = root;

    while ( curr != null)
 {       if ( curr. left == null) { print (curr. data); curr = curr. right)}
     else
   { temp = curr. left;                1 ——                    2nd li
     while( temp. right != null && temp. right != curr)
            temp = temp. right;


         if( temp. right == NULL)
         {
                  temp. right = curr;
                   curr = curr. left;

          }
         else  {                                        2nd ti
                     temp. right = null;
                       print( curr. data);
                        curr = curr. right;

                }
       }
 }
```