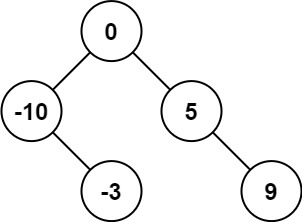
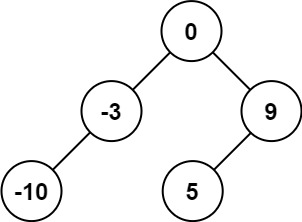
**108. Convert Sorted Array to Binary Search Tree**

Easy

Given an integer array nums where the elements are sorted in **ascending order**, convert *it to a****height-balanced****binary search tree*.

A **height-balanced** binary tree is a binary tree in which the depth of the two subtrees of every node never differs by more than one.

**Example 1:**

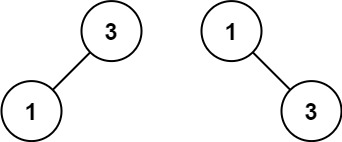


**Input:** nums = [-10,-3,0,5,9]

**Output:** [0,-3,9,-10,null,5]

**Explanation:** [0,-10,5,null,-3,null,9] is also accepted:

**Example 2:**



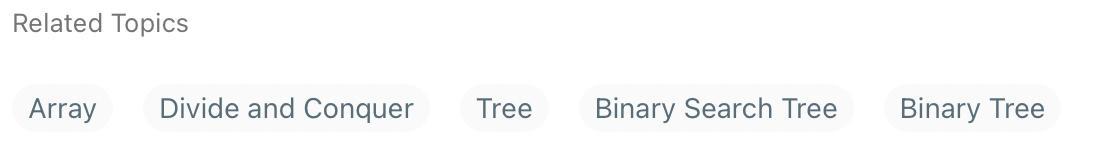
**Input:** nums = [1,3]

**Output:** [3,1]

**Explanation:** [1,null,3] and [3,1] are both height-balanced BSTs.

**Constraints:**

* 1 <= nums.length <= 104
* -104 <= nums[i] <= 104
* nums is sorted in a **strictly increasing** order.



/\*\*

\* Definition for a binary tree node.

\* public class TreeNode {

\* int val;

\* TreeNode left;

\* TreeNode right;

\* TreeNode() {}

\* TreeNode(int val) { this.val = val; }

\* TreeNode(int val, TreeNode left, TreeNode right) {

\* this.val = val;

\* this.left = left;

\* this.right = right;

\* }

\* }

\*/

class Solution {

public TreeNode create(int[] nums, int index, int length) {

TreeNode head;

int lengthl = length / 2;

int lengthr = length - (length / 2 + 1);

if (length == 1) {

head = new TreeNode(nums[index], null, null);

} else if (length == 2) {

head = new TreeNode(nums[index], create(nums, index - (lengthl + 1) / 2, lengthl), null);

} else

head = new TreeNode(nums[index], create(nums, index - (lengthl + 1) / 2, lengthl),

create(nums, index + lengthr / 2 + 1, lengthr));

return head;

}

public TreeNode sortedArrayToBST(int[] nums) {

TreeNode head = create(nums, nums.length / 2, nums.length);

return head;

}

}

* In this approch I used analogy to binary search and recursion

class Solution {  
public TreeNode sortedArrayToBST(int[] nums) {

return Tree(nums,0,nums.length-1);

}

public TreeNode Tree(int[] nums,int l ,int h){

TreeNode ans = null;

if(l<=h){

int m = (l+h)/2;

TreeNode root = new TreeNode(nums[m]);

root.left = Tree(nums,l,m-1);

root.right = Tree(nums,m+1,h);

ans = root;

}

return ans;}