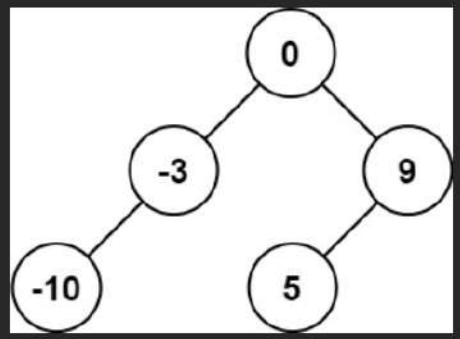
Description | Editorial | Solutions (4.6K) | Submissions

108. Convert Sorted Array to Binary Search Tree

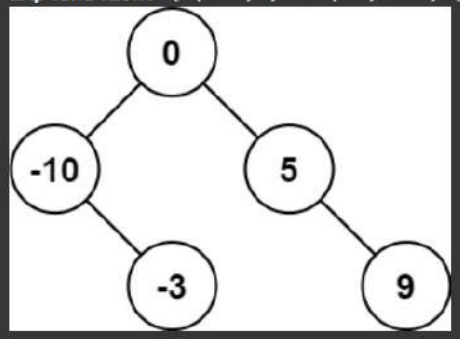
Easy | 9.7K | 482 | Companies

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

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:
```



Java | Auto

```java
}

class Solution {
    public TreeNode sortedArrayToBST(int[] arr) {
        if(arr.length==0)return null;
        return bst(arr,0,arr.length-1);
    }

    public TreeNode bst(int arr[], int i, int j){
        if(i>j)return null;

        int mid = i+(j-i)/2;
        TreeNode node = new TreeNode(arr[mid]);
        node.left = bst(arr, i, mid-1);
        node.right = bst(arr, mid+1, j);
        return node;
    }
}
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

nums =
[-10,-3,0,5,9]

Output

[0,-10,5,null,-3,null,9]

Expected

[0,-3,9,-10,null,5]

Console | Run | Submit

Description | Editorial | Solutions (2K) | Submissions

- Time complexity: $O(n)$
- Space complexity: $O(n)$ on the stack

Code

```
class Solution {
  int prev = -100000, min = Integer.MAX_VALUE;

  public int getMinimumDifference(TreeNode node) {
    if (node == null) return min;

    getMinimumDifference(node.left);
    min = Math.min(min, node.val - prev);
    prev = node.val;
    getMinimumDifference(node.right);

    return min;
  }
}
```

If you like my solution, please upvote it!

Previous: Beats 100% Easy + Video | Java C++ P...

Next: Easy Java Solution | #javith_sadham_h...

Comments (0) Sort by: Best

Type comment here... (Markdown supported)

Upvote 4 | Comments 0 | Favorite | Share

Java | Auto

```
12            this.right = right;
13  *     }
14  * }
15  */
16 class Solution {
17   int prev = -100000, min = Integer.MAX_VALUE;
18
19   public int getMinimumDifference(TreeNode node) {
20     if (node == null) return min;
21
22     getMinimumDifference(node.left);
23     min = Math.min(min, node.val - prev);
24     prev = node.val;
25     getMinimumDifference(node.right);
26
27     return min;
28   }
29 }
```

Testcase | Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

root =
[4,2,6,1,3]

Output

1

Expected

1

Console | Run | Submit