

108. Convert Sorted Array to Binary Search Tree

107 Binary Tree Level Order Traversal II

- Depth-first Search + Tree

Description

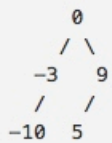
Given an array where elements are sorted in ascending order, convert it to a height balanced BST.

For this problem, a height-balanced binary tree is defined as a binary tree in which the depth of the two subtrees of *every* node never differ by more than 1.

Example:

Given the sorted array: [-10,-3,0,5,9],

One possible answer is: [0,-3,9,-10,null,5], which represents the following height balanced BST:



1. Thought line

2. Depth-first Search + Tree

```
1 /**
2  * Definition for a binary tree node.
3  * struct TreeNode {
4  *     int val;
5  *     TreeNode *left;
6  *     TreeNode *right;
7  *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
8  * };
9  */
10 void arrayRootFind(vector<int>& nums, int st, int ed, TreeNode* node, string str = "toRightChild"){
11     if (st>ed) return;
12
13     int mid = (st+ed)/2;
14
15     if (str == "toRightChild"){
16         node->right = new TreeNode(nums[(st+ed)/2]);
17         arrayRootFind(nums, st, mid-1, node->right, "toLeftChild");
18         arrayRootFind(nums, mid+1, ed, node->right, "toRightChild");
19     }
20     else if (str == "toLeftChild"){
21         node->left = new TreeNode(nums[(st+ed)/2]);
22         arrayRootFind(nums, st, mid-1, node->left, "toLeftChild");
23         arrayRootFind(nums, mid+1, ed, node->left, "toRightChild");
24     }
25 }
26
27 class Solution {
```

```
28 public:
29     TreeNode* sortedArrayToBST(vector<int>& nums) {
30         if (nums.empty()) return nullptr;
31         TreeNode* dummyHead = new TreeNode(INT_MIN);
32         arrayRootFind(nums, 0, nums.size()-1, dummyHead);
33         return dummyHead->right;
34     }
35 };
```