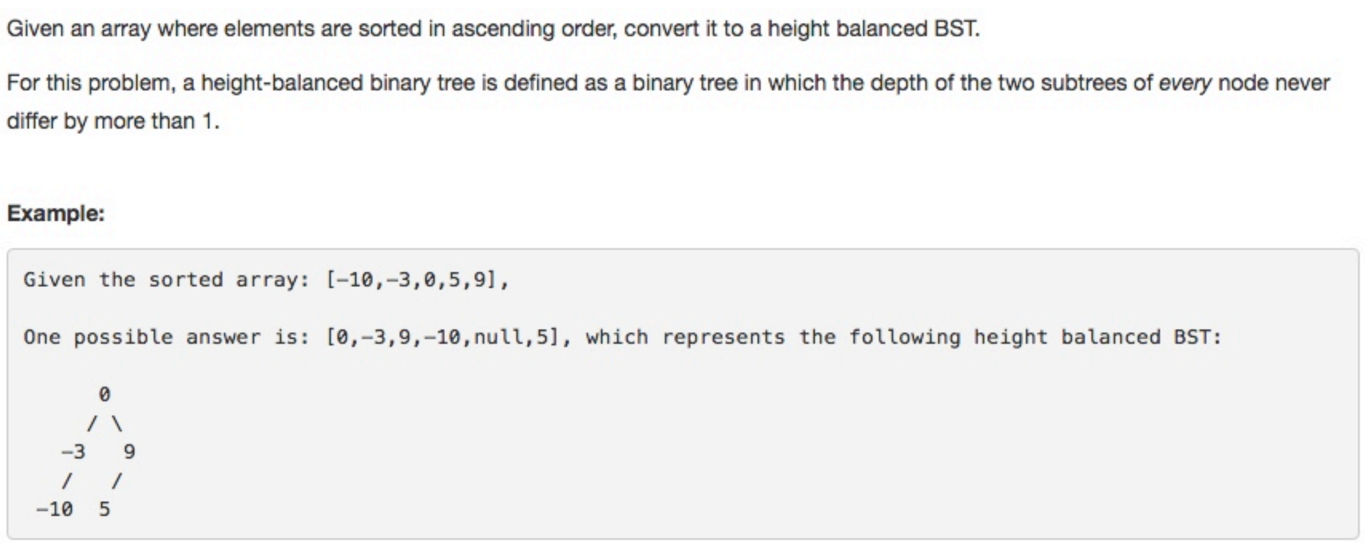
108. Convert Sorted Array to Binary Search Tree

108 Convert Sorted Array to Binary Search Tree

- **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:

      0
     / \
   -3   9
   /   /
 -10  5
```

1. Thought line

- Height-balanced BST

2. Depth-first Search + Tree

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode(int x) : val(x), left(NULL), right(NULL) {}
 * };
 */
void arrayRootFind(vector<int>& nums, int st, int ed, TreeNode* node, string str = "toRightChild"){
    if (st>ed) return;

    int mid = (st+ed)/2;

    if (str == "toRightChild"){
        node->right = new TreeNode(nums[(st+ed)/2]);
        arrayRootFind(nums, st, mid-1, node->right, "toLeftChild");
        arrayRootFind(nums, mid+1, ed, node->right, "toRightChild");
    }
    else if (str == "toLeftChild"){
        node->left = new TreeNode(nums[(st+ed)/2]);
        arrayRootFind(nums, st, mid-1, node->left, "toLeftChild");
        arrayRootFind(nums, mid+1, ed, node->left, "toRightChild");
    }
}
```

```cpp

class Solution {
public:
    TreeNode* sortedArrayToBST(vector<int>& nums) {
        if (nums.empty()) return nullptr;
        TreeNode* dummyHead = new TreeNode(INT_MIN);
        arrayRootFind(nums, 0, nums.size()-1, dummyHead);
        return dummyHead->right;
    }
};
```