Documentation Convert Sorted Array to Binary Search Tree

Problem Statement

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

Example 1

Input:

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

Output:

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

Explanation:

The following tree structures are all height-balanced BSTs for the input array [-10, -3, 0, 5, 9]:

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

Example 2

Input:

nums = [1, 3]

Output:

[3, 1]

Explanation:

Both of the following are height-balanced BSTs for the input array [1, 3]:

- [1, null, 3]
- [3, 1]

Constraints

- 1 <= nums.length <= 10^4
- -10^4 <= nums[i] <= 10^4
- nums is sorted in a strictly increasing order.

Solution

The goal is to convert the sorted array nums into a height-balanced binary search tree. A height-balanced BST is defined as a binary tree in which the depth of the two subtrees of every node never differs by more than one.

Approach

The key to achieving a height-balanced BST is to always pick the middle element of the current subarray to be the root. This way, we ensure that the left and right subtrees are of approximately equal size, maintaining the balance of the tree.

Algorithm

1. **Base Case:** If the input array is empty, return None.
2. **Recursive Case:** Use a helper function to build the BST:
   - Select the middle element of the current subarray as the root node.
   - Recursively build the left subtree using the left half of the current subarray.
   - Recursively build the right subtree using the right half of the current subarray.

Explanation of the Code

- **TreeNode Class:** Defines the structure of a binary tree node with val, left, and right attributes.
- **Solution Class:** Contains the method sortedArrayToBST which converts the sorted array into a BST.
  - *Base Case:* Checks if the nums array is empty. If true, returns None.
  - *helper Function:* A recursive function that constructs the BST.
    - ✓ Determines the middle index of the current subarray.
    - ✓ Creates a TreeNode with the middle element.
    - ✓ Recursively constructs the left and right subtrees using the left and right halves of the subarray, respectively.
    - ✓ Returns the constructed node, which serves as the root for the current subarray.
  - *Return Statement:* Initiates the recursive process by calling helper with the full range of the input array.

This approach ensures that the BST is height-balanced by always choosing the middle element as the root for each subarray, thus evenly distributing the nodes.