**Problem Statement**

This problem was asked by Etsy.

You are working as a software engineer for a company that handles large amounts of data. One of your tasks is to efficiently store and retrieve data in a way that allows for quick searches and balanced access times. Given a sorted array of unique integers, your task is to convert it into a height-balanced binary search tree (BST). A height-balanced BST ensures that the depth of the two subtrees of every node never ers by more than one, leading to optimized search times.

**Input Format**

* The first line contains an integer n, the size of the array.
* The second line contains n space-separated integers representing the elements of the sorted array.

**Output Format**

* Print the in-order traversal of the BST.

**Constraints**

* 1 <= n <= 1000
* -10^4 <= nums[i] <= 10^4

**Example**

**Example 1**

**Input:**

7

-10 -3 0 5 9

**Output:**

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

**Example 2**

**Input:**

6

1 2 3 4 5 6

**Output:**

[1, 2, 3, 4, 5, 6]

**Solution**

python

class TreeNode:

def \_\_init\_\_(self, val=0, left=None, right=None):

self.val = val

self.left = left

self.right = right

def sortedArrayToBST(nums):

def helper(left, right):

if left > right:

return None

mid = (left + right) // 2

node = TreeNode(nums[mid])

node.left = helper(left, mid - 1)

node.right = helper(mid + 1, right)

return node

return helper(0, len(nums) - 1)

def printInOrder(node):

if not node:

return []

return printInOrder(node.left) + [node.val] + printInOrder(node.right)

# Input Reading

n = int(input())

nums = list(map(int, input().split()))

# Convert the sorted array to a height-balanced BST

root = sortedArrayToBST(nums)

# Print the in-order traversal of the BST

print(printInOrder(root))

**Explanation**

1. **TreeNode Class**: A simple class to represent nodes of the binary tree.
2. **sortedArrayToBST Function**:
   * helper is a recursive function that constructs the BST.
   * It finds the middle element and makes it the root.
   * Recursively builds the left and right subtrees using the subarrays to the left and right of the middle element.
3. **printInOrder Function**:
   * This function is used to print the BST in-order to verify the correctness of the tree.

**Additional Test Cases**

**Test Case 1**

**Input:**

5

-3 -1 2 4 6

**Output:**

[-3, -1, 2, 4, 6]

**Test Case 2**

**Input:**

8

-5 -3 -2 0 1 3 5 7

**Output:**

[-5, -3, -2, 0, 1, 3, 5, 7]

**Test Case 3**

**Input:**

3

1 3 5

**Output:**

[1, 3, 5]