Q1. Given two strings text1 and text2, return the length of their longest common subsequence. If there is no common subsequence, return 0.
A subsequence of a string is a new string generated from the original string with some characters (can be none) deleted without changing the relative order of the remaining characters.

For example, "ace" is a subsequence of "abcde".
A common subsequence of two strings is a subsequence that is common to both strings.

Example 1:
Input: text1 = "abcde", text2 = "ace"
Output: 3
Explanation: The longest common subsequence is "ace" and its length is 3.
Example 2:

Input: text1 = "abc", text2 = "abc"
Output: 3
Explanation: The longest common subsequence is "abc" and its length is 3.
Example 3:

Input: text1 = "abc", text2 = "def"
Output: 0
Explanation: There is no such common subsequence, so the result is 0.

Constraints:
1 <= text1.length, text2.length <= 1000
text1 and text2 consist of only lowercase English characters.

Q2. Given an unsorted array of integers nums, return the length of the longest consecutive elements sequence.
You must write an algorithm that runs in O(n) time.

Example 1:
Input: nums = [100,4,200,1,3,2]
Output: 4
Explanation: The longest consecutive elements sequence is [1, 2, 3, 4]. Therefore its length is 4.

Example 2:
Input: nums = [0,3,7,2,5,8,4,6,0,1]
Output: 9

Example 3:
Input: nums = [1,0,1,2]
Output: 3

Constraints:
0 <= nums.length <= 10^5
-109 <= nums[i] <= 10^9

Q3. Given an integer array nums, return the length of the longest strictly increasing subsequence.

Example 1:
Input: nums = [10,9,2,5,3,7,101,18]
Output: 4
Explanation: The longest increasing subsequence is [2,3,7,101], therefore the length is 4.

Example 2:
Input: nums = [0,1,0,3,2,3]
Output: 4

Example 3:
Input: nums = [7,7,7,7,7,7,7]
Output: 1

Constraints:
1 <= nums.length <= 2500
-10^4 <= nums[i] <= 10^4

Follow up: Can you come up with an algorithm that runs in O(n log(n)) time complexity?

Q4.Check if each internal node of a BST has exactly one child

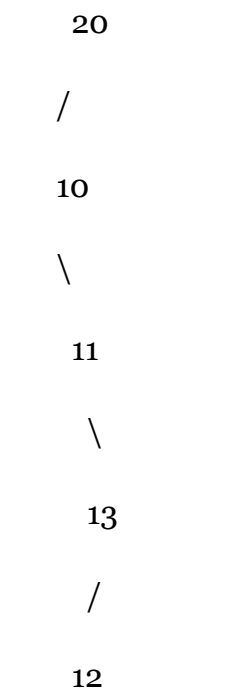
Given Preorder traversal of a BST, check if each non-leaf node has only one child. Assume that the BST contains unique entries.

Test case 1:

Input:= [20, 10, 11, 13, 12]

Output: Yes

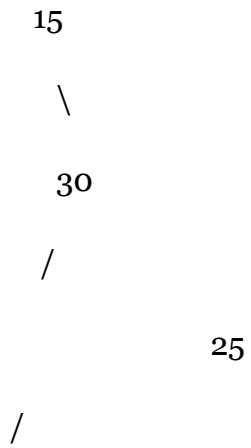
The given array represents the following BST. In the following BST, every internal node has exactly 1 child. Therefore, the output is true.



Test case 2:

Input: [15, 30, 25, 18, 20]

Output: Yes



18

\

20

Q5. Find the smallest missing element from a sorted array

Given a sorted array of non-negative distinct integers, find the smallest missing non-negative element in it.

For example,

Test case 1:

Input: nums[] = [0, 1, 2, 6, 9, 11, 15]

Output: The smallest missing element is 3

Test case 2:

Input: nums[] = [1, 2, 3, 4, 6, 9, 11, 15]

Output: The smallest missing element is 0

Q6. Given two strings word1 and word2, return the minimum number of operations required to convert word1 to word2.

You have the following three operations permitted on a word:

Insert a character
Delete a character
Replace a character

Example 1:
Input: word1 = "horse", word2 = "ros"
Output: 3

Explanation:
horse -> rorse (replace 'h' with 'r')
rorse -> rose (remove 'r')

rose -> ros (remove 'e')

Example 2:
Input: word1 = "intention", word2 = "execution"
Output: 5

Explanation:
intention -> inention (remove 't')
inention -> enention (replace 'i' with 'e')
enention -> exention (replace 'n' with 'x')
exention -> exection (replace 'n' with 'c')
exection -> execution (insert 'u')

Constraints:
0 <= word1.length, word2.length <= 500
word1 and word2 consist of lowercase English letters.