Problem Statement

A binary search tree (BST) is defined as a binary tree in which each node satisfies the property such that its value is larger than the value of every node in its left subtree, and less than or equal to the value of every node in its right subtree. The distance between two values in a binary search tree is the minimum number of edges traversed to reach from one value to the other.

Given a list of *n* unique integers, construct a BST by inserting each integer in the given order without rebalancing the tree. Then, find the distance between the two given nodes, *node1* and *node2*, of the BST. In case, either *node1* or *node2* is not present in the tree, return -1.

Input
The input to the function/method consists of four arguments - *values*, representing a list of integers; *n*, an integer representing the number of elements in the list; *node1*, an integer representing the first node and *node2*, an integer representing the second node.

Output
Return an integer representing the distance between *node1* and *node2*, else return -1, if either *node1* or *node2* is not present in the tree.

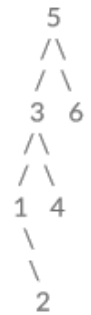
Constraints
$0 < n < 2^{31}$
$0 \leq values[i] < 2^{31}$
$0 \leq i < n$

Example
Input:

values = [5, 6, 3, 1, 2, 4], *n* = 6, *node1* = 2 and *node2* = 4

Output:
3

Explanation:
For the list *values* = [5, 6, 3, 1, 2, 4], the tree is given as below:



The path traversed from *node1* = 2 to *node2* = 4 is: 2, 1, 3, and 4, so output is 3.

Problem Statement

Many popular text editors include a feature to detect unbalanced brackets. A pair of brackets is balanced if each opening bracket is accompanied by a closing bracket. Furthermore, the order is important. An enclosed bracket is only considered balanced if it's corresponding open or close bracket is enclosed by the same set of balanced bracket pairs.

For a given string consisting of the sequence of characters, write an algorithm to check whether the brackets in the string are balanced or not. If the brackets in the string are balanced then the output is 1 otherwise 0.

Input
The input to the function/method consists of an argument - *str*, a string representing the sequence of brackets.

Output
Return 1 if the brackets in the given string are balanced, else return 0.

Constraint
Brackets in string *str* are '(', ')','{', '}', '[', ']', '<', '>' only.

Examples
Example1:
Input:
str = (h[e{I<I>o}!]~)()()()(

Output:

0

Explanation:
All the brackets in the string *str* are not balanced.

Example2:
Input:
str = []{}<>

Output:
1

Explanation:
All the brackets in the string *str* are balanced.