Problem 449: Serialize and Deserialize BST

Problem Information

Difficulty: Medium
Acceptance Rate: 59.08%
Paid Only: No
Tags: String, Tree, Depth-First Search, Breadth-First Search, Design, Binary Search Tree, Binary Tree

Problem Description

Serialization is converting a data structure or object into a sequence of bits so that it can be stored in a file or memory buffer, or transmitted across a network connection link to be reconstructed later in the same or another computer environment.

Design an algorithm to serialize and deserialize a **binary search tree**. There is no restriction on how your serialization/deserialization algorithm should work. You need to ensure that a binary search tree can be serialized to a string, and this string can be deserialized to the original tree structure.

The encoded string should be as compact as possible.

Example 1:

Input: root = [2,1,3] **Output:** [2,1,3]

Example 2:

Input: root = [] **Output:** []

Constraints:

* The number of nodes in the tree is in the range `[0, 104]`. * `0 <= Node.val <= 104` * The input tree is **guaranteed** to be a binary search tree.

Code Snippets

C++:

```
/**
* Definition for a binary tree node.
* struct TreeNode {
* int val;
* TreeNode *left;
* TreeNode *right;
* TreeNode(int x) : val(x), left(NULL), right(NULL) {}
* };
*/
class Codec {
public:

// Encodes a tree to a single string.
string serialize(TreeNode* root) {

}

// Decodes your encoded data to tree.
TreeNode* deserialize(string data) {

}
};

// Your Codec object will be instantiated and called as such:
// Codec* ser = new Codec();
// Codec* deser = new Codec();
// string tree = ser->serialize(root);
// TreeNode* ans = deser->deserialize(tree);
// return ans;
```

Java:

```
/**
* Definition for a binary tree node.
* public class TreeNode {
* int val;
* TreeNode left;
* TreeNode right;
* TreeNode(int x) { val = x; }
```

```
* }
*/
public class Codec {

// Encodes a tree to a single string.
public String serialize(TreeNode root) {

}

// Decodes your encoded data to tree.
public TreeNode deserialize(String data) {

}
}

// Your Codec object will be instantiated and called as such:
// Codec ser = new Codec();
// Codec deser = new Codec();
// String tree = ser.serialize(root);
// TreeNode ans = deser.deserialize(tree);
// return ans;
```

Python3:

```
# Definition for a binary tree node.
# class TreeNode:
# def __init__(self, x):
# self.val = x
# self.left = None
# self.right = None

class Codec:

def serialize(self, root: Optional[TreeNode]) -> str:
"""Encodes a tree to a single string.
"""


def deserialize(self, data: str) -> Optional[TreeNode]:
"""Decodes your encoded data to tree.
"""
```

```
# Your Codec object will be instantiated and called as such:
# Your Codec object will be instantiated and called as such:
# ser = Codec()
# deser = Codec()
# tree = ser.serialize(root)
# ans = deser.deserialize(tree)
# return ans
```