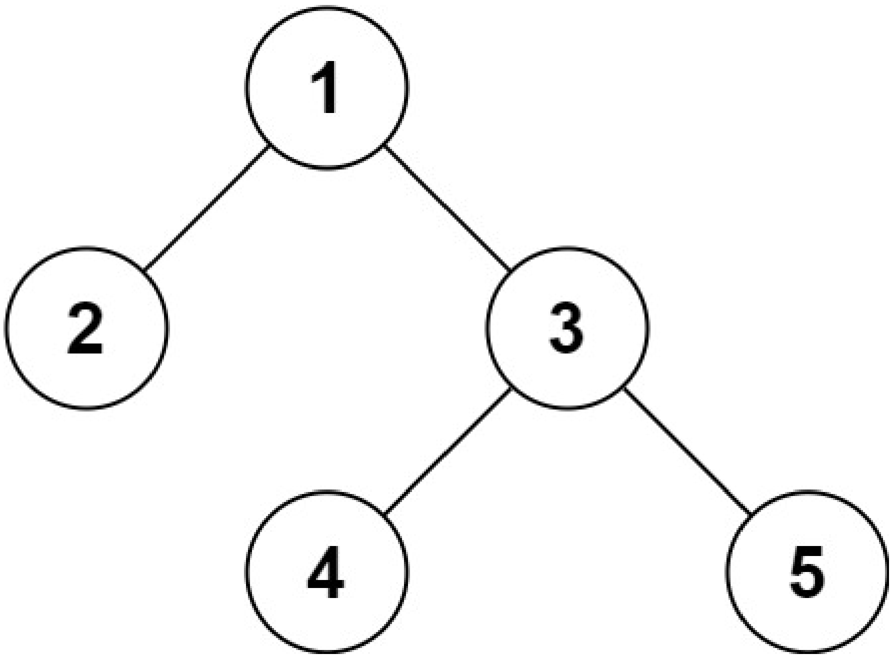
297. Serialize and Deserialize Binary Tree

Serialization is the process of 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 tree. There is no restriction on how your serialization/deserialization algorithm should work. You just need to ensure that a binary tree can be serialized to a string and this string can be deserialized to the original tree structure.

Clarification: The input/output format is the same as how LeetCode serializes a binary tree. You do not necessarily need to follow this format, so please be creative and come up with different approaches yourself.

Example 1:



```
Input: root = [1,2,3,null,null,4,5]
Output: [1,2,3,null,null,4,5]
```

Example 2:

```
Input: root = []
Output: []
```

Example 3:

```
Input: root = [1]
Output: [1]
```

Example 4:

```
Input: root = [1,2]
Output: [1,2]
```

Constraints:

- The number of nodes in the tree is in the range `[0, 10<sup>4</sup>]`.
- `-1000 <= Node.val <= 1000`

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

class Codec:
    def __init__(self):
        self.idx = 0
    def serialize(self, root):
        ans = []
        self.serializeHelper(root,ans)
        return ''.join(ans)

    def deserialize(self, data):
        ans = data.split(',')
        ans.pop()
        root = self.deserializeHeler(ans)
        return root
        # print(ans)
    def deserializeHeler(self,ans):
        if self.idx>=len(ans) or ans[self.idx]=='None':
            self.idx+=1
            return None

        node = TreeNode(int(ans[self.idx]))
        self.idx+=1
        node.left = self.deserializeHeler(ans)
        node.right = self.deserializeHeler(ans)
        return node
    def serializeHelper(self,root,ans):
        if root is None:
            ans.append('None'+',')
```

```python
            return
        temp = str(root.val)+','
        ans.append(temp)
        self.serializeHelper(root.left,ans)
        self.serializeHelper(root.right,ans)



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