331. Verify Preorder Serialization of a Binary Tree

link

Input: "9,3,4,#,#,1,#,#,2,#,6,#,#"

给定一个前序二叉树顺序, 判断他是不是二叉树, 自己试着写了deserialize没写出来...首先把所有的元素都放进stack 然后pop如果是#就return null, 他就没用子节点, 否则left就是接下来的, right就是left处理完的. 开始是想用递归, 然后写不出来....

link

link

297. Serialize and Deserialize Binary Tree

link

递归

```
private static final String spliter = ",";
private static final String NN = "#";

// Encodes a tree to a single string.
public String serialize(TreeNode root) {
    StringBuilder sb = new StringBuilder();
    buildString(root, sb);
    return sb.toString();
}

private void buildString(TreeNode node, StringBuilder sb) {
    if (node == null) {
```

```
            sb.append(NN).append(spliter);
        } else {
            sb.append(node.val).append(spliter);
            buildString(node.left, sb);
            buildString(node.right,sb);
        }
    }
    // Decodes your encoded data to tree.
    public TreeNode deserialize(String data) {
        Deque<String> nodes = new LinkedList<>();
        nodes.addAll(Arrays.asList(data.split(spliter)));
        return buildTree(nodes);
    }

    private TreeNode buildTree(Deque<String> nodes) {
//      remove跟pop一样
        String val = nodes.remove();
        if (val.equals(NN)) return null;
        else {
            TreeNode node = new TreeNode(Integer.valueOf(val));
            node.left = buildTree(nodes);
            node.right = buildTree(nodes);
            return node;
        }
    }
```

迭代, while里面用while, 我写的时候想着一次处理一个节点, 实际上应该是不同的情况i一直加

自己写的时候想着第二个null才pop, 但是这样的话pop会少一层, 逻辑有问题, 实际上每遇到一个null就pop, 然后处理pop的元素的右节点, 只把右节点push

```
    public TreeNode deserialize(String data) {
        if (data.length()==0) return null;
        String[] node=data.split(" ");
        int n=node.length;
        Deque<TreeNode> stack=new LinkedList<>();
        TreeNode root=new TreeNode(Integer.valueOf(node[0]));
        TreeNode x=root;
        stack.push(x);

        int i=1;
        while (i<n) {
            while (i<n && !node[i].equals("null")) {
                x.left=new TreeNode(Integer.valueOf(node[i++]));
                x=x.left;
                stack.push(x);
            }
            while (i<n && node[i].equals("null")) {
                x=stack.pop();
                i++;
            }
            if (i<n) {
                x.right=new TreeNode(Integer.valueOf(node[i++]));
                x=x.right;
                stack.push(x);
```

```
            }
        }
        return root;
    }
```