**前言**：今天重温了树的遍历（前序遍历，中序遍历和后序遍历），可以用来判断两个树是否结构相同。又接触到了对称二叉树，可以用递归和迭代两种方法解决，后文会着重讲述迭代的方法，因为顺便学习队列的知识了。

* StringBuffe,StringBuilderr中没有对equals函数重写，里面相当于是判断变量==变量，即判断是否指向同一个地址，这个时候要比较值是否相等可以转换为String类型。
* java中Queue的使用方法

1. 添加元素 add(),offer() 在未超出容量时都从队尾压入元素，并返回这个元素。但当超出容量时，add()方法会抛出异常（程序终止），offer()则返回false
2. 删除元素 remove(),poll() 在队列大于0的时候，都是从队头删除并返回该元素。但在队列为空时，remove()会抛出异常，poll()返回false
3. 获取队头元素 element(),peek() 当队列为空时element会抛出异常，peek()则返回null



* ArrayList和LinkedList两个的实现方式不一样，前者是数组，后者是双向链表，前者访问快，后者插入和删除快。占内存则是后者多，因为还需要多存储两个指针用于前后的指向。

List和ArrayList的区别，前者是接口，后者则是继承了前者，并且实现了很多方法。List不能直接创建对象，但是可以通过ArrayList创建一个对象然后指向List即List list = new ArrayList()，这是list只能访问List类里的方法，不能访问子类的。

* 对于List的反转。Collections.reverse()可以实现，但是它没有返回值，而是直接修改列表内部的元素。

**Leetcode101 Symmetric Tree**

    //对称二叉树，用迭代的形式

    public boolean isSymmetric(TreeNode root) {

    Queue<TreeNode>q=new LinkedList<>();

    if(root==null)return true;

    q.offer(root.left);

    q.offer(root.right);

    while(!q.isEmpty()){

        TreeNode l=q.poll();

        TreeNode r=q.poll();

        if(l==null&&r==null)continue;

        if(l==null||r==null)return false;

        if(l.val!=r.val)return false;

        q.offer(l.left);

        q.offer(r.right);

        q.offer(l.right);

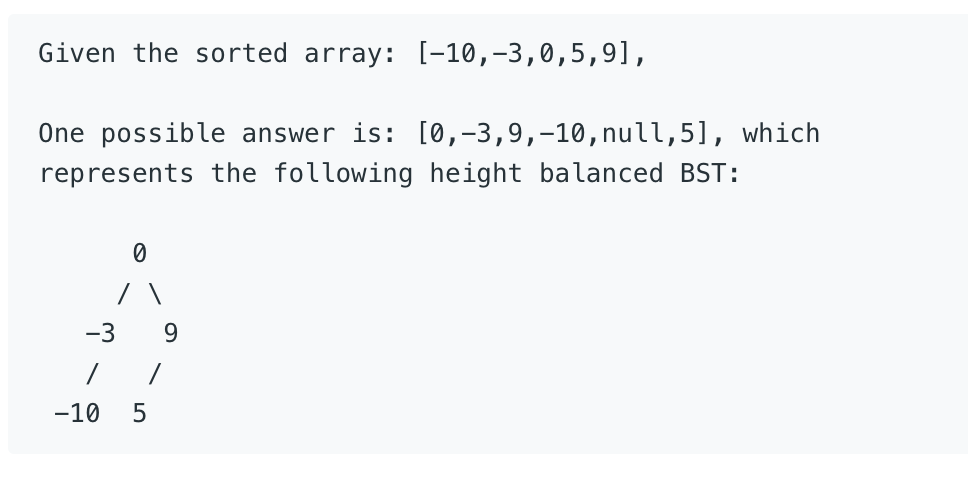
        q.offer(r.left);

    }

    return true;

    }

#### LeetCode [108. Convert Sorted Array to Binary Search Tree](https://leetcode-cn.com/problems/convert-sorted-array-to-binary-search-tree/)



给定一组已经排序的数组，构建一个高度平衡的二叉搜索树。高度平衡要求两个子树的节点深度不能超过1.首先回顾一下**二叉搜索树**的概念：要求大小比较时，左子树<根结点<右子树，是二叉树里最基本的（可惜我刚做的时候还是忘了，尴尬。。。。）为了达到高度平衡，可以每次都让数组的中间节点作为根结点，将左右子树的个数一分为二。

源代码如下：

 public TreeNode sortedArrayToBST(int[] nums) {

        //BST的中序遍历是按照从小到大的顺序排列的，所以左子树<根结点<右子树。所以一直选择中间节点作为根结点，创建BST

            return Created(nums,0,nums.length-1);

    }

    private TreeNode Created(int[] nums,int left,int right)

    {

        if(left>right)return null;

        int mid=(left+right)/2;

        TreeNode root=new TreeNode(nums[mid]);

        root.left=Created(nums,left,mid-1);

        root.right=Created(nums,mid+1,right);

        return root;

    }

**总结：**

**今天做的六都题都和树有关，重新熟悉了树的遍历（迭代和递归两种方式）还有构建二叉搜索树。**