[剑指 Offer 53 - I. 在排序数组中查找数字 I](https://leetcode-cn.com/problems/zai-pai-xu-shu-zu-zhong-cha-zhao-shu-zi-lcof/)

统计一个数字在排序数组中出现的次数。

class Solution {

    public int search(int[] nums, int target) {

        int n=0;

     for(int i=0;i<nums.length;i++){

         if(nums[i]==target){

             n++;

         }

     }

     return n;

    }

}

心得：此题比较简单，只需将整个数组遍历，添加一个计数器即可，时间复杂度o（n）空间复杂度是o（1），当然，也可考虑使用哈希表作为题解。

备注：一般在生产过程中，尽量不写出n++这类代码，因为要考虑到数据的原子性问题，n++代码经历过读数据，＋1，再写数据三个过程，在多线程场景下，没有办法保证数据的一致性，加锁会增加系统开销，所以要考虑使用原子操作类。（知识拓展）

#### [剑指 Offer 53 - II. 0～n-1中缺失的数字](https://leetcode-cn.com/problems/que-shi-de-shu-zi-lcof/)

一个长度为n-1的递增排序数组中的所有数字都是唯一的，并且每个数字都在范围0～n-1之内。在范围0～n-1内的n个数字中有且只有一个数字不在该数组中，请找出这个数字。

class Solution {

    public int missingNumber(int[] nums) {

    int left=0;

    int right=nums.length-1;

    while(left<right){

        int mid =left+right>>1;

        if(nums [mid]>mid){

        right=mid;

        }

        else{

            left= mid+1;

        }

    }

    return right==nums[right] ? right+1:right;

    }

}

心得：作为一种习惯，只要是看到有序数组，必须想到二分查找，有序必然二分，因为二分查找的时间复杂度是o(logn),空间复杂度是o（1），能有效降低系统内存的开销。当然，此题使用遍历也是可以的，时间复杂度和空间复杂度都是o（n），必须将优化系统开销作为一种思维习惯，虽然代码量多了，但是系统开销少了。此题用到的是二分查找的变形，需要比较的是数组的值和数组下标的值，但是用二分查找需要注意缺失0和缺失最后一项的情况。

备注：写算法时条件不能乱放，要理解题目的意思，本人犯过的错误仅供参考：

#### [剑指 Offer 04. 二维数组中的查找](https://leetcode-cn.com/problems/er-wei-shu-zu-zhong-de-cha-zhao-lcof/)

在一个 n \* m 的二维数组中，每一行都按照从左到右递增的顺序排序，每一列都按照从上到下递增的顺序排序。请完成一个高效的函数，输入这样的一个二维数组和一个整数，判断数组中是否含有该整数。示例:

现有矩阵 matrix 如下：

[

[1, 4, 7, 11, 15],

[2, 5, 8, 12, 19],

[3, 6, 9, 16, 22],

[10, 13, 14, 17, 24],

[18, 21, 23, 26, 30]

]

给定 target = 5，返回 true。

给定 target = 20，返回 false。

限制：

0 <= n <= 1000

0 <= m <= 1000

class Solution {

    public boolean findNumberIn2DArray(int[][] matrix, int target) {

      int n;

      int m;

      for(n=0;n<matrix.length;n++){

          for(m=0;m<matrix[0].length;m++){

              if (matrix[n][m]==target)

              {

              return true;}

          }

      }

       return false;

    }

}

心得：这题我只会暴力遍历，好的解法我暂时没有想到。过段时间再更新。

#### [剑指 Offer 11. 旋转数组的最小数字](https://leetcode-cn.com/problems/xuan-zhuan-shu-zu-de-zui-xiao-shu-zi-lcof/)

把一个数组最开始的若干个元素搬到数组的末尾，我们称之为数组的旋转。输入一个递增排序的数组的一个旋转，输出旋转数组的最小元素。例如，数组 [3,4,5,1,2] 为 [1,2,3,4,5] 的一个旋转，该数组的最小值为1。

class Solution {

    public int minArray(int[] numbers) {

    int left=0,right=numbers.length-1;

    while(left<right){

        int mid = left+right>>1;

        if(numbers[mid]>numbers[right]){

        left= mid+1;

        }

        else if(numbers[mid]<numbers[right]){

            right=mid;

        }

        else{

            right--;

        }

    }

     return numbers[right];

    }

}

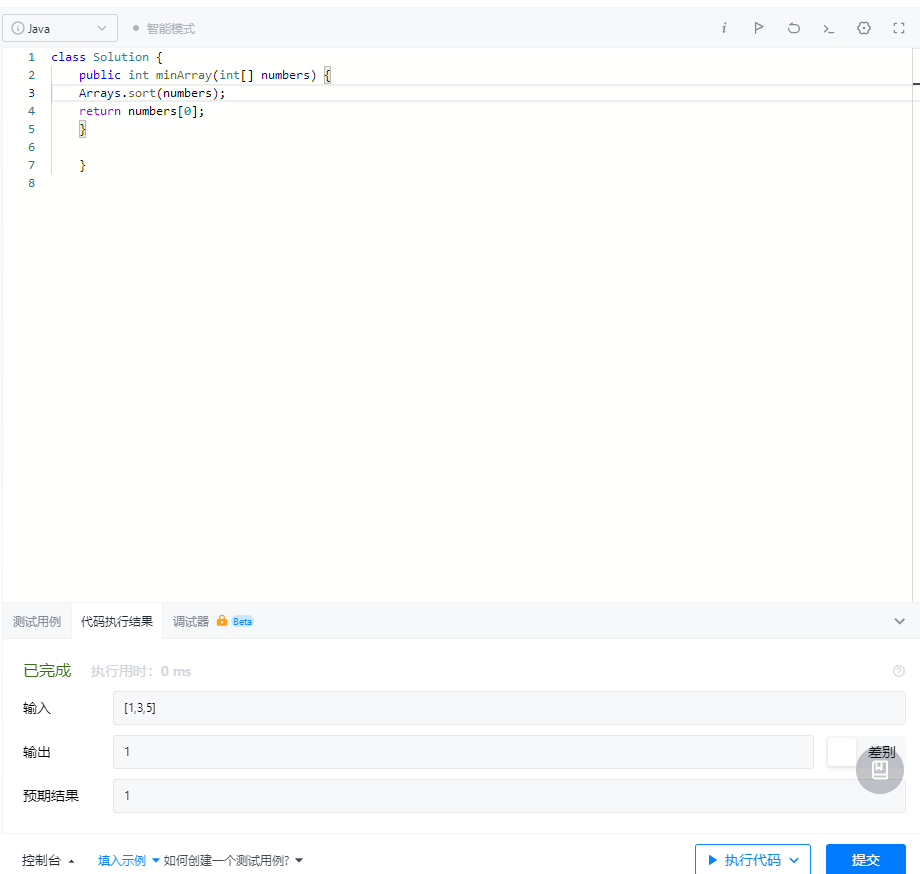
心得：首先要理解题目的意思，通俗说就是把数组的前n项插到数组的末尾，在示例中n=2，

主要思路是先考虑旋转点在哪里，把数组分开两半，旋转点可能在前半部分，也可能在后半部分，用二分法缩小查询范围，用right—缩小查询访问。

备注：left+right>>1与left+(right-left)>>1的区别，在数值计算上，两者没有区别，区别在于前者在right特别大时，会出现内存溢出的情况（整型越界），后者就不会出现这种情况。

另外，要注意if和else if的区别，if条件语句是无论如何都会执行的，else if则是上一句的条件判断失败，才会执行，如果判断成功，则不会执行，if和elseif本质上是互相排斥的。

当然，这种奇技淫巧也是可以的，思路是直接排序，找最小的值，但是要深入学习sort排序的本质。



#### [剑指 Offer 32 - III. 从上到下打印二叉树 III](https://leetcode-cn.com/problems/cong-shang-dao-xia-da-yin-er-cha-shu-iii-lcof/)

难度中等133

请实现一个函数按照之字形顺序打印二叉树，即第一行按照从左到右的顺序打印，第二层按照从右到左的顺序打印，第三行再按照从左到右的顺序打印，其他行以此类推。

例如:  
给定二叉树: [3,9,20,null,null,15,7],

3

/ \

9 20

/ \

15 7

返回其层次遍历结果：

[

[3],

[20,9],

[15,7]

]

**提示：**

1. 节点总数 <= 1000

#### [剑指 Offer 32 - II. 从上到下打印二叉树 II](https://leetcode-cn.com/problems/cong-shang-dao-xia-da-yin-er-cha-shu-ii-lcof/)

从上到下按层打印二叉树，同一层的节点按从左到右的顺序打印，每一层打印到一行。

例如:

给定二叉树: [3,9,20,null,null,15,7],

3

/ \

9 20

/ \

15 7

返回其层次遍历结果：

[

[3],

[9,20],

[15,7]

]

提示：

节点总数 <= 1000

#### [剑指 Offer 32 - I. 从上到下打印二叉树](https://leetcode-cn.com/problems/cong-shang-dao-xia-da-yin-er-cha-shu-lcof/)

难度中等126收藏分享切换为英文接收动态反馈

从上到下打印出二叉树的每个节点，同一层的节点按照从左到右的顺序打印。

例如:  
给定二叉树: [3,9,20,null,null,15,7],

3

/ \

9 20

/ \

15 7

返回：

[3,9,20,15,7]

**提示：**

1. 节点总数 <= 1000

/\*\*

\* Definition for a binary tree node.

\* public class TreeNode {

\* int val;

\* TreeNode left;

\* TreeNode right;

\* TreeNode(int x) { val = x; }

\* }

\*/

class Solution {

public int[] levelOrder(TreeNode root) {

if (root == null) {

return new int[0];

}

Queue<TreeNode> nodeQueue = new LinkedList();

nodeQueue.add(root);

List<Integer> list = new ArrayList();

while (!nodeQueue.isEmpty()) {

TreeNode node = nodeQueue.poll();

list.add(node.val);

if (node.left != null) {

nodeQueue.add(node.left);

}

if (node.right != null) {

nodeQueue.add(node.right);

}

}

int[] res = new int[list.size()];

for(int i = 0; i < list.size(); i++) {

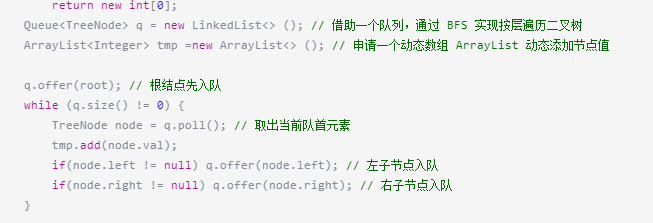
res[i] = list.get(i);

}

return res;

}

}

心得：首先要理解题目的意思，是需要将遍历的节点输出到数组中，解题思路是，先将二叉树的元素放入队列当中，再将队列中的元素，按照先进先出的原则，塞进动态数组当中，再通过动态数组的值和元素，打印出 需要的数组。思路可参考截图如下：

#### [剑指 Offer 63. 股票的最大利润](https://leetcode-cn.com/problems/gu-piao-de-zui-da-li-run-lcof/)

假设把某股票的价格按照时间先后顺序存储在数组中，请问买卖该股票一次可能获得的最大利润是多少？

示例 1:

输入: [7,1,5,3,6,4]

输出: 5

解释: 在第 2 天（股票价格 = 1）的时候买入，在第 5 天（股票价格 = 6）的时候卖出，最大利润 = 6-1 = 5 。

注意利润不能是 7-1 = 6, 因为卖出价格需要大于买入价格。

示例 2:

输入: [7,6,4,3,1]

输出: 0

解释: 在这种情况下, 没有交易完成, 所以最大利润为 0。

限制：

0 <= 数组长度 <= 10^5

class Solution {

    public int maxProfit(int[] prices) {

        int res=0;

        if (prices.length<2){

            return 0;

        }

        int low = prices[0];

        for(int i=0;i<prices.length;i++){

            if(prices[i]>low){

                res =Math.max(res,prices[i]-low);

            }

            else if (prices[i]<=low){

                low=prices[i];

            }

        }

        return res;

    }

}

心得：

要先考虑，当数组长度小于2时，股票是不能卖出的。对数组进行循环遍历，目的是找出两两之间最大的差值，当后面的值比前面的大时，就是没有卖出机会的意思，直接返回0即可。

可能会使用到max方法，同时引申一下max\_value。

（可以使用动态规划）。

