Java方向编程题答案

day2

[编程题]36846-汽水瓶

https://www.nowcoder.com/practice/fe298c55694f4ed39e256170ff2c205f?tpId=37&&tqId=21245&rp=1&ru=/activity/oj&qru=/ta/huawei/question-ranking

【题目解析】：童鞋们在遇到这种类似数学的问题时，切记不要慌，我们来看一下这个题目。题目表达的意思很明确，3个空瓶子换一瓶饮料。如果给你10个空瓶，问你可以换多少饮料喝？我们可以在纸上简单画一下，这个题目的思路就出来了？

【解题思路】：当有n个空瓶时候，我们设总共可以喝total瓶饮料，那么当n>2时，说明我们可以去换饮料喝。3个换一瓶，那么可以换n/3瓶饮料。接下来，这些饮料喝完后，势必还有空瓶。那么喝完n/3瓶饮料之后的空瓶数是多少？答案是：n/3+n%3.这里需要注意：n/3代表换来的饮料，但是这些喝完之后，不就是空瓶吗？相同的为什么需要n%3?因为比如n==4,最后的空瓶数应该是：2.不就是n/3+n%3吗？到这里，我们在看一下。n>2时，可以去换，那么n==2时，我们可以去商铺借一瓶饮料。那么又多喝了一瓶。至此这个题目就已经可以下手去做了。

【示例代码】：

```
import java.util.*;
public class Main {
    public static int drink(int n) {
        int total = 0;//饮料总数
        //当大于两个空瓶的时候才可以换饮料
        while(n > 2) {
            //n/3-->能够换来的饮料
            total = total+n/3;
            //新的空瓶子个数--》n%3可能会不能整除
            n = n/3 + n%3;
        }
        //如果n==2，说明有两个空瓶可以喝商家借一瓶饮料
        if(n == 2) {
            total = total + 1;
        }
        return total;
    }
    public static void main(String[] args){
        //创建键盘录入对象
        Scanner sc=new Scanner(System.in);
        int n;
        while(sc.hasNext())
        {
            n=sc.nextInt();
            System.out.println(drink(n));
        }

    }
}
```

[编程题]24990-数组中的逆序对

https://www.nowcoder.com/questionTerminal/bb06495cc0154e90bbb18911fd581df6

【题目解析】： 数组的逆序对考题较多，比如示例的数组：[1,2,3,4,5,6,7,0] 那么，它包含逆序对有：{1,0},{2,0},{3,0},{4,0},{5,0},{6,0},{7,0}总共7个逆序对。再比如数组{7,5,6,4}，逆序对总共有5对，{7,5},{7,6},{7,4},{5,4,{6,4}；

【解题思路】： 思路1：暴力解法，顺序扫描整个数组，每扫描到一个数字的时候，逐个比较该数字和它后面的数字的大小。如果后面的数字比它小，则这两个数字就组成一个逆序对。假设数组中含有n个数字，由于每个数字都要和O(n)个数字作比较，因此这个算法的时间复杂度是O(n2)。

思路2：分治思想，采用归并排序的思路来处理，如下图，先分后治：先把数组分隔成子数组，先统计出子数组内部的逆序对的数目，然后再统计出两个相邻子数组之间的逆序对的数目。在统计逆序对的过程中，还需要对数组进行排序，其实这个排序过程就是归并排序的思路 .

逆序对的总数=左边数组中的逆序对的数量+右边数组中逆序对的数量+左右结合成新的顺序数组时中出现的逆序对的数量；

优秀博客链接：http://www.cnblogs.com/tongkey/p/7815179.html

【示例代码】：

```
import java.util.*;
public class AntiOrder {
    public int count(int[] A, int n) {
        if (A == null || n == 0) {
            return 0;
        }
        return mergeSortRecursion(A, 0, n - 1);
    }

    public static int mergeSortRecursion(int[] arr, int l, int r) {
        if (l == r) {
            return 0;
        }
        int mid = (l + r) / 2;
        //逆序对的总数=左边数组中的逆序对的数量+右边数组中逆序对的数量+左右结合成新的顺序数组时中出现的
逆序对的数量；
        return mergeSortRecursion(arr, l, mid) + mergeSortRecursion(arr, mid + 1, r) +
merge(arr, l, mid, r);
    }

    public static int merge(int[] arr, int left, int mid, int right) {
        int[] temp = new int[right - left + 1];
        int index = 0;
        int i = left;
        int j = mid + 1;
        int inverseNum = 0;// 新增，用来累加数组逆序对
        while (i <= mid && j <= right) {
            if (arr[i] <= arr[j]) {
                temp[index++] = arr[i++];
            } else {
                // 当前一个数组元素大于后一个数组元素时，累加逆序对

                // s[i] > s[j] 推导出 s[i]...s[mid] > s[j]
```

```
                inverseNum += (mid - i + 1);
                temp[index++] = arr[j++];
            }
        }
        while (i <= mid) {
            temp[index++] = arr[i++];
        }
        while (j <= right) {
            temp[index++] = arr[j++];
        }
        for (int k = 0; k < temp.length; k++) {
            arr[left++] = temp[k];
        }
        return inverseNum;
    }
}
```