# JAVA方向编程题答案

## 第一周

### day1

> [编程题]45842-统计回文
>
> 链接：https://www.nowcoder.com/questionTerminal/9d1559511b3849deaa71b576fa7009dc
>
> **【题目解析】：**
>
> 童鞋们看到这个题目应该心中一喜，没错就是咱们之前作业题中关于回文的问题，换汤不换药~我们使用String类提供的方法就可以轻松解决~
>
> **【解题思路】：**
>
> 判断回文:这里我们首先将用户输入的字符串变为一个个的char，一个指针从开头向后遍历每个char的同时另一个指针从后向前遍历，当发现字符不相等时说明不是回文串。
>
> 查找插入位置:知道如何判断回文后我们只需要将第二个字符串从第一个字符串第一个位置开始尝试插入直到判断回文的方法返回true时即找到插入位置
>
> **【示例代码】：**

```
import java.util.*;
public class Main {
    public static boolean isHuiwen(String s){
        int i = 0;
        int j = s.length()-1;
        while(i<j){
            if(s.charAt(i)!=s.charAt(j)){
                return false;
            }
            i++;
            j--;
        }
        return true;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String str1 = sc.nextLine();
        String str2 = sc.nextLine();
        int count = 0;
        for(int i = 0; i <= str1.length();i++){
            StringBuilder sb = new StringBuilder(str1);
            sb.insert(i, str2);
            if(isHuiwen(sb.toString())){
                count++;
            }
        }
        System.out.println(count);
    }
```

```
}
```

[编程题]44581-寻找第K大

链接：https://www.nowcoder.com/questionTerminal/e016ad9b7f0b45048c58a9f27ba618bf

**【题目解析】：**

老铁们，这个可是咱们上快排的时候说的原题哦~不过当时只说了思路，相信把快排分区思想理解透彻的你解决这个问题应该得心应手

**【解题思路】：**

这题应该是用快排的思想：例如找49个元素里面第24大的元素，那么按如下步骤： 1.进行一次快排（将大的元素放在前半段，小的元素放在后半段）,假设得到的中轴为p 2.判断 p - low + 1 == k ， 如果成立，直接输出a[p]， （因为前半段有k - 1个大于a[p]的元素，故a[p]为第K大的元素） 3.如果 p - low + 1 > k， 则第k大的元素在前半段，此时更新high = p - 1，继续进行步骤1 4.如果p - low + 1 < k， 则第k大的元素在后半段， 此时更新low = p + 1, 且 k = k - (p - low + 1)，继续步骤1. 由于常规快排要得到整体有序的数组，而此方法每次可以去掉“一半”的元素，故实际的复杂度不是o(nlgn), 而是o(n)。

**【示例代码】：**

```
public class Finder {
 public int findKth(int[] a, int n, int K) {
     return findKth(a, 0, n-1, K);
 }
 public int findKth(int[] a, int low, int high, int k) {
     int part = partation(a, low, high);

     if(k == part - low + 1) return a[part];
     else if(k > part - low + 1) return findKth(a, part + 1, high, k - part + low -1);
     else return findKth(a, low, part -1, k);

 }
 public int partation(int[] a, int low, int high) {
     int key = a[low];
     while(low < high) {
         while(low < high && a[high] <= key) high--;
         a[low] = a[high];
         while(low < high && a[low] >= key) low++;
         a[high] = a[low];
     }
     a[low] = key;
     return low;
 }
}
```