二分查找、冒泡排序、选择排序、插入排序、快速排序

笔记本:	微信		
创建时间:	2022/10/8 15:55	**更新时间:**	2022/10/8 18:04
作者:	729034664@qq.com		
URL:	about:blank		

二分查找

```java
public class FuTest {
    public static void main(String[] args) {
        int [] array = {1,5,8,11,15,33,35,45,56,77,89};
        int target = 45;

        int idx = binarySearch(array, target);
        System.out.println(idx);
    }

    public static int binarySearch(int[] a, int t){
        int l=0, r=a.length-1, m;

        while(l <= r){
            m = (l + r) >>> 1;
            if(a[m] == t){
                return m;
            }else if (a[m] < t){
                l = m + 1;
            }else{
                r = m -1;
            }
        }
```

位运算解决整数溢出问题

如果是用 m = (l + r) / 2;的话，如果数值超过整数的最大范围（21亿多），然后目标又比左边大的话，（l+r）就会超出整数的最大范围，除以2得到的是负数。
这里需要用位运算来解决，而且运行效率更高。

能力要求：
能够手写冒泡、快排的代码（熟练）

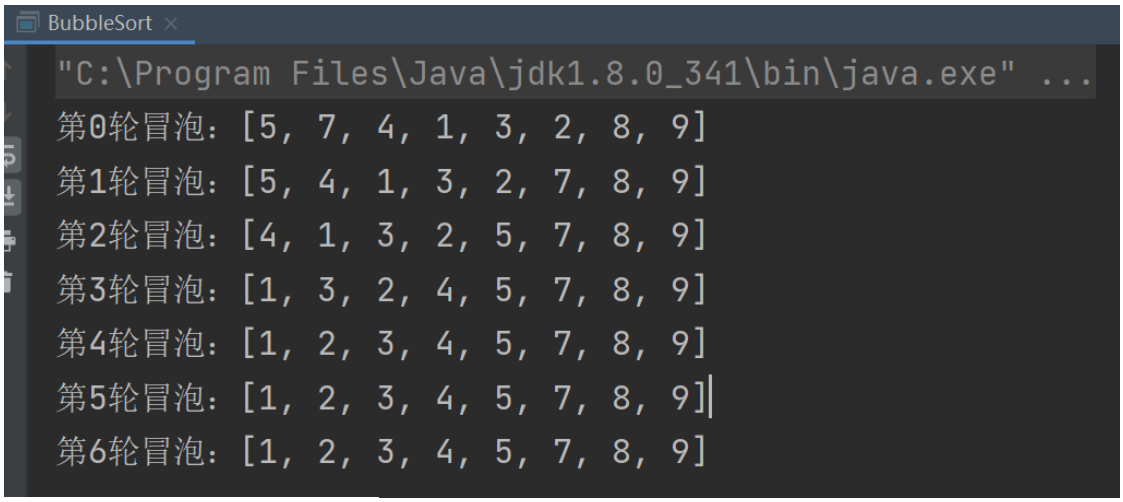
冒泡排序

初步实现：

```java
public class BubbleSort {
    public static void main(String[] args) {
```

```
        int[] a = {5, 9, 7, 4, 1, 3, 2, 8};


        bubble(a);
    }


    public static void bubble(int[] a){
        for(int j = 0; j < a.length -1; j++){
            // 一轮冒泡
            for (int i = 0; i < a.length - 1; i++){
                if (a[i] > a[i + 1]){
                    swap(a, i, i + 1);
                }
            }
            System.out.println("第"+j+"轮冒泡：" + Arrays.toString(a));
        }
    }


    public static void swap(int[] a, int i, int j){
        int t = a[i];
        a[i] = a[j];
        a[j] = t;
    }
}
```

```
public class BubbleSort {
    public static void main(String[] args) {
        int[] a = {5, 9, 7, 4, 1, 3, 2, 8};

        bubble(a);
    }

    public static void bubble(int[] a){
        for(int j = 0; j < a.length -1; j++){
            // 一轮冒泡
            for (int i = 0; i < a.length - 1; i++){
                if (a[i] > a[i + 1]){
                    swap(a, i, j: i + 1);
                }
            }
            System.out.println("第"+j+"轮冒泡：" + Arrays.toString(a));
        }
    }

    public static void swap(int[] a, int i, int j){
        int t = a[i];
        a[i] = a[j];
        a[j] = t;
    }
}
```



这个代码的效率问题：
第一次循环9到了最后，第2次循环的时候，8还要再次和9去做比较（无效比较）。
优化方法：内存循环的时候，比较次数应该为：a.length - 1 - j。

从第4轮开始，已经有序了。
如果某次比较，没有发生一次交换，那么整个数组都已经有序了，就不需要再继续进行比较了。