算法思路

冒泡排序的思想是相邻两个比较，如果前面比后面的大，那么进行交换，不断比较，那么第一轮完毕，最大的就到了最右边

然后开启下一轮，继续从头开始，相邻两个比较，根据结果决定是否交换，由于第一轮已将最大的放到了最右边，所有第二轮比较范围比第

依次类推，直到N - 1 轮比较完毕，就可以拿到结果

文字说明

开始第一轮

第一个和第二个比较，将大的那一个元素存放在第二个位置，小的那个放在第一个位置

第二个和第三个比较，将大的那一个元素存放在第三个位置，小的那一个放在第二个位置

依次类推，当第一轮比较完毕，那么最大的放在了最后一个位置。

开始第二轮

第二轮中排除第一轮最大的那一个

第一个和第二个比较，大的放在第二个位置，小的放在第一个位置

第二个和第三个比较，将大的那一个元素存放在第三个位置，小的那一个放在第二个位置

依次类推，第二轮结束，第二轮中最大的放在倒数第二个位置。

最终

经过N-1轮比较，即可得到最终结果。 N 表示数组长度

图解说明

代码实现

```java
package day11.排序;

/**
 * 冒泡排序
 */
public class BubbleSort {

    public static void main(String[] args) {
        int[] array = {4, 2, 8, 9, 5, 7, 6, 1, 3};
        //未排序数组顺序为
        System.out.println("未排序数组顺序为: ");
        display(array);
        System.out.println("-----------------------");
        bubble2(array);
        System.out.println("经过冒泡排序后的数组顺序为: ");
        display(array);
    }

    public static void display(int[] array) {
        for (int i = 0; i < array.length; i++) {
            System.out.print(array[i] + " ");
        }
        System.out.println();
    }
```

```java

    public static void bubble1(int[] array) {
        for (int i = 0; i < array.length; i++) {
            for (int j = 0; j < array.length - i - 1; j++) {
                if (array[j] > array[j + 1]) {
                    int tmp = array[j + 1];
                    array[j + 1] = array[j];
                    array[j] = tmp;
                }
            }
        }
    }

    public static void bubble2(int[] array) {
        // 冒泡排序会在某次比较后达到有序，此时我们就没有必要再去进行for循环比较了。可以提交结束比较
        // 我们可以设置一个标志位，若是在某次循环中没有发生序列变化，就代表了已经排序完成，我们不需要后面的排序了
        for (int i = 0; i < array.length; i++) {
            boolean flag = true;
            for (int j = 0; j < array.length - i - 1; j++) {
                if (array[j] > array[j + 1]) {
                    int tmp = array[j + 1];
                    array[j + 1] = array[j];
                    array[j] = tmp;
                    flag = false;
                }
            }
            if (flag) {
                break;
            }
            //第 i轮排序的结果为
            System.out.print("第" + (i + 1) + "轮排序后的结果为:");
            display(array);
        }
    }
}

/* 输出

未排序数组顺序为:
4 2 8 9 5 7 6 1 3
------------------------
第1轮排序后的结果为:2 4 8 5 7 6 1 3 9
第2轮排序后的结果为:2 4 5 7 6 1 3 8 9
第3轮排序后的结果为:2 4 5 6 1 3 7 8 9
第4轮排序后的结果为:2 4 5 1 3 6 7 8 9
第5轮排序后的结果为:2 4 1 3 5 6 7 8 9
第6轮排序后的结果为:2 1 3 4 5 6 7 8 9
第7轮排序后的结果为:1 2 3 4 5 6 7 8 9
经过冒泡排序后的数组顺序为:
1 2 3 4 5 6 7 8 9

*/
```

总结

假设参与比较的数组元素个数为 N，则第一轮排序有 N-1 次比较，第二轮有 N-2 次，如此类推，这种序列的公式为

当 N 的值很大时，算法比较次数约为 N2/2次比较，忽略减1。

假设数据是随机的，那么每次比较可能要交换位置，可能不会交换，假设概率为50%，那么交换次数为 N2/4。不过如果是最坏的情况，要交换位置。

交换和比较次数都和N2 成正比。由于常数不算大 O 表示法中，忽略 2 和 4，那么冒泡排序运行都需要 O(N2) 时间级别。

其实无论何时，只要看见一个循环嵌套在另一个循环中，我们都可以怀疑这个算法的运行时间为 O(N2)级，外层循环执行 N 次，内层循几分之N次）。这就意味着大约需要执行N2次某个基本操作