一、什么是冒泡排序

1.概念

冒泡排序（Bubble Sort）是排序算法里面比较简单的一个排序。它重复地走访要排序的数列，一次比较两个数据元素，如果顺序不对则进行交换，并一直重复这样的走访操作，直到没有要交换的数据元素为止。

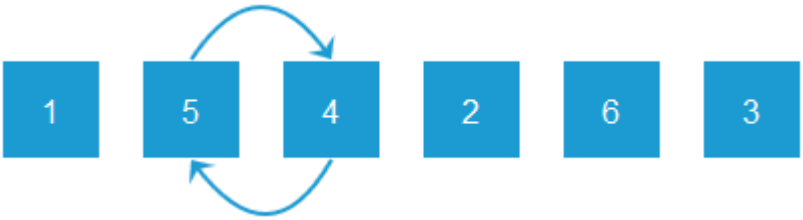
2.算法原理

这是一个无序数列：1、5、4、2、6、3，我们要将它按从小到大排序。按照冒泡排序的思想，我们要把相邻的元素两两比较，根据大小来交换元素的位置

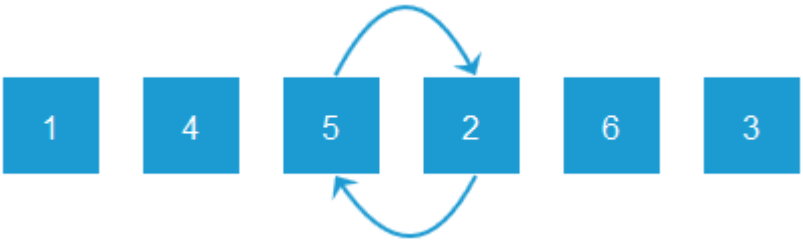
首先开始第一轮比较

第一步：比较1和5，1比5小，顺序正确，元素位置不变

第二步：比较5和4，5比4大，顺序错误，交换元素位置



第三步：比较5和2，5比2大，顺序错误，交换元素位置



经过一轮比较后，6作为最大的元素到了序列的最右侧

接下来进行第二轮比较，从1和4开始比较，到最右边的3结束，6已经是有序的，不需要再参与比较

第二轮结束后，如下所示

第三轮结束后，如下所示

第四轮结束后，如下所示

第五轮结束后，如下所示

至此所有的元素都是有序的

3.算法实现

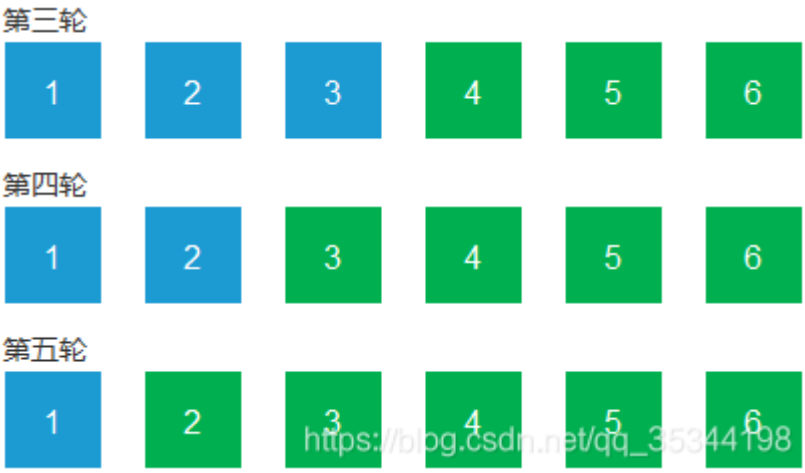
```
function sort(arr) {
    let length = arr.length;
    for (let i = 0; i < length - 1; i++) {
        for (let j = 0; j < length - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                [arr[j], arr[j + 1]] = [arr[j + 1], arr[j]];
            }
        }
    }
}

let arr = [1, 5, 4, 2, 6, 3];
```

```
sort(arr);
console.log(arr);
```

二、算法优化

优化一

从图中可以看到，在第三轮时，序列已经是有序了，但程序还是进行了第四、第五轮排序。可以在排序时做个标记，如果序列已经有序了，就不再进行后续的排序



优化后代码如下：

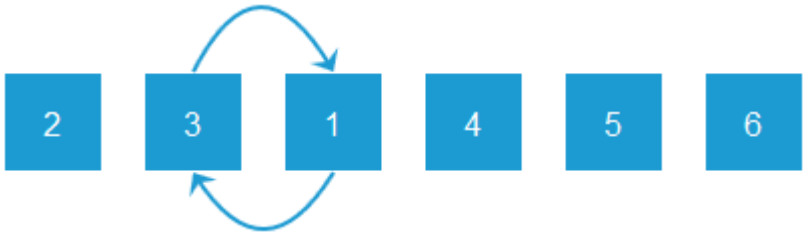
```
function sort(arr) {
    let length = arr.length;
    for (let i = 0; i < length - 1; i++) {
        // 优化，isSorted判断是否有序，已经有序，不需要再继续交换
        let isSorted = true;
        for (let j = 0; j < length - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                [arr[j], arr[j + 1]] = [arr[j + 1], arr[j]];
                isSorted = false;
            }
        }
        if (isSorted) {
            break;
        }
    }
}

let arr = [1, 5, 4, 2, 6, 3];
sort(arr);
console.log(arr);
```

优化二

看如下序列

第一步，2和3比较，2比3小，顺序正确，元素位置不变

第二步，3和1比较，3比1大，顺序错误，交换元素位置



第三步，3和4比较，3比4小，顺序正确，元素位置不变

第四步，4和5比较，4比5小，顺序正确，元素位置不变

第五步，5和6比较，5比6小，顺序正确，元素位置不变

可以看到后4位元素已经是有序了，但还是进行了比较，并且在第二轮，第三轮还会对后面有序的元素进行比较。可以通过记录每轮交换后，最后一次交换的位置，进行优化

优化后代码如下所示：

```
function sort(arr) {
    let length = arr.length;
    // 优化2，记录无序数列的边界，每次比较只需要比到这里为止
    let lastExchangeIndex = 0;
    let sortBorder = length - 1;
    for (let i = 0; i < length - 1; i++) {
        // 优化1，isSorted判断是否有序，已经有序，不需要再继续交换
        let isSorted = true;
        for (let j = 0; j < sortBorder; j++) {
```

```
            if (arr[j] > arr[j + 1]) {
                [arr[j], arr[j + 1]] = [arr[j + 1], arr[j]];
                isSorted = false;
                lastExchangeIndex = j;
            }
        }
        sortBorder = lastExchangeIndex;
        if (isSorted) {
            break;
        }
    }
}

let arr = [2, 3, 1, 4, 5, 6];
sort(arr);
console.log(arr);
```

其实这还不是最优的，有一种排序算法叫鸡尾酒排序算法，能对冒泡排序算法做进一步优化达到最优的目的，详情可参考鸡尾酒排序算法。

三、冒泡排序算法特点

1.时间复杂度

冒泡排序算法的每一轮要遍历所有元素，轮转的次数和元素数量相当，所以时间复杂度是O(N^2)

经过优化后，最优的情况，序列已经是顺序的，那么只要进行一次循环，所以最优时间复杂度是O(N)

2.空间复杂度

冒泡排序算法排序过程中需要一个临时变量进行两两交换，所需要的额外空间为1，因此空间复杂度为O(1)

3.稳定性

冒泡排序算法在排序过程中，元素两两交换时，相同元素的前后顺序并没有改变，所以冒泡排序是一种稳定排序算法