冒泡排序（Bubble Sort）

■ 冒泡排序也叫做起泡排序

■ 执行流程（本课程统一以升序为例子）

① 从头开始比较每一对相邻元素，如果第1个比第2个大，就交换它们的位置

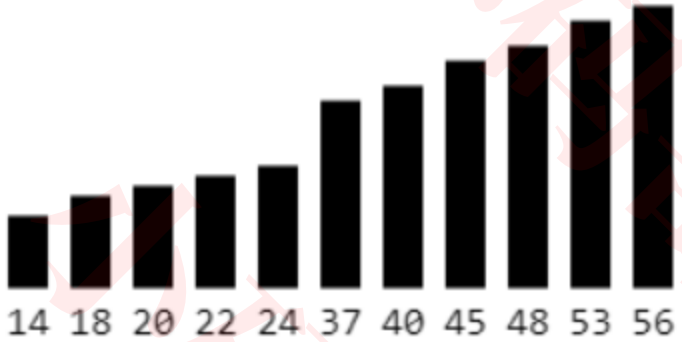
✓ 执行完一轮后，最末尾那个元素就是最大的元素

② 忽略 ① 中曾经找到的最大元素，重复执行步骤 ①，直到全部元素有序

```
for (int end = array.length - 1; end > 0; end--) {
    for (int begin = 1; begin <= end; begin++) {
        if (cmp(begin, begin - 1) < 0) {
            swap(begin, begin - 1);
        }
    }
}
```

冒泡排序 – 优化①

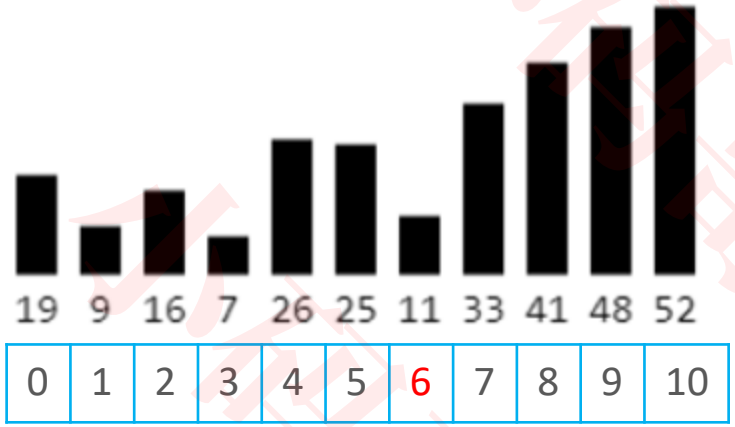
- 如果序列已经完全有序，可以提前终止冒泡排序




```
for (int end = array.length - 1; end > 0; end--) {
    boolean sorted = true;
    for (int begin = 1; begin <= end; begin++) {
        if (cmp(begin, begin - 1) < 0) {
            swap(begin, begin - 1);
            sorted = false;
        }
    }
    if (sorted) break;
}
```

冒泡排序 – 优化②

- 如果序列尾部已经局部有序，可以记录最后1次交换的位置，减少比较次数



- 最后1次交换的位置是 6

```
for (int end = array.length - 1; end > 0; end--) {
    int sortedIndex = 1;
    for (int begin = 1; begin <= end; begin++) {
        if (cmp(begin, begin - 1) < 0) {
            swap(begin, begin - 1);
            sortedIndex = begin;
        }
    }
    end = sortedIndex;
}
```

- 最坏、平均时间复杂度：$O(n^2)$
- 最好时间复杂度：$O(n)$
- 空间复杂度：$O(1)$

排序算法的稳定性（Stability）

- 如果相等的2个元素，在排序前后的相对位置保持不变，那么这是稳定的排序算法
  - 排序前：$5, 1, 3_a, 4, 7, 3_b$
  - 稳定的排序：$1, 3_a, 3_b, 4, 5, 7$
  - 不稳定的排序：$1, 3_b, 3_a, 4, 5, 7$

- 对自定义对象进行排序时，稳定性会影响最终的排序效果

- 冒泡排序属于稳定的排序算法
  - 稍有不慎，稳定的排序算法也能被写成不稳定的排序算法，比如下面的冒泡排序代码是不稳定的

```
for (int end = array.length - 1; end > 0; end--) {
    for (int begin = 1; begin <= end; begin++) {
        if (cmp(begin, begin - 1) <= 0) {
            swap(begin, begin - 1);
        }
    }
}
```

原地算法（In-place Algorithm）

- 何为原地算法?
  - 不依赖额外的资源或者依赖少数的额外资源，仅依靠输出来覆盖输入
  - 空间复杂度为 $O(1)$ 的都可以认为是原地算法

- 非原地算法，称为 Not-in-place 或者 Out-of-place

- 冒泡排序属于 In-place