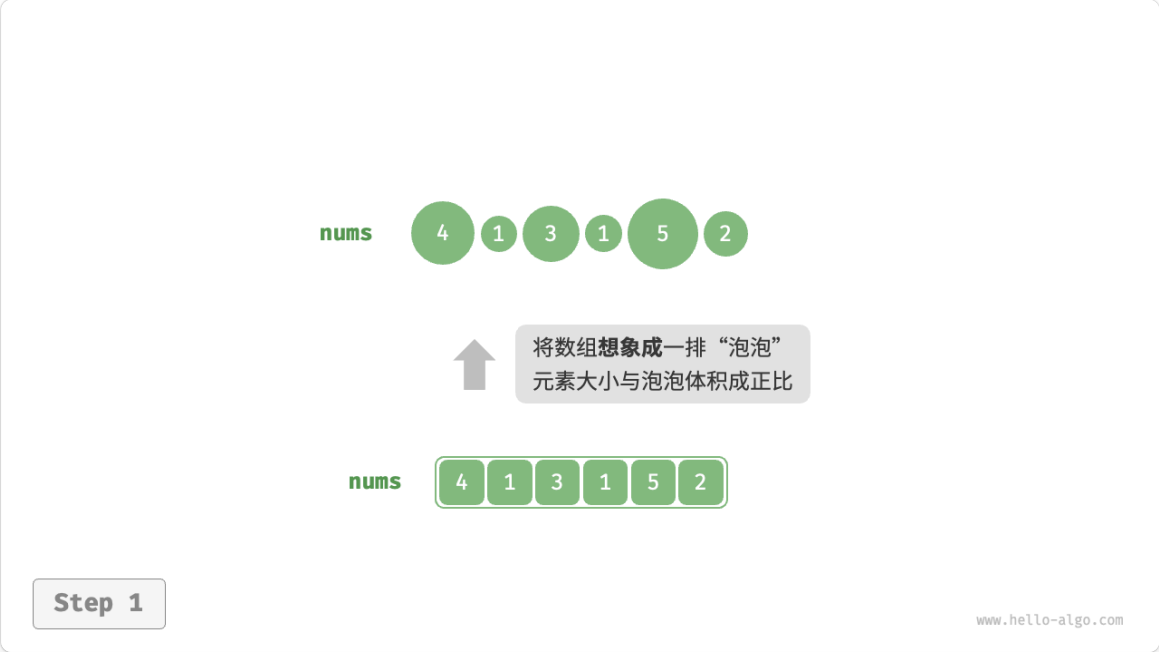
11.3 冒泡排序

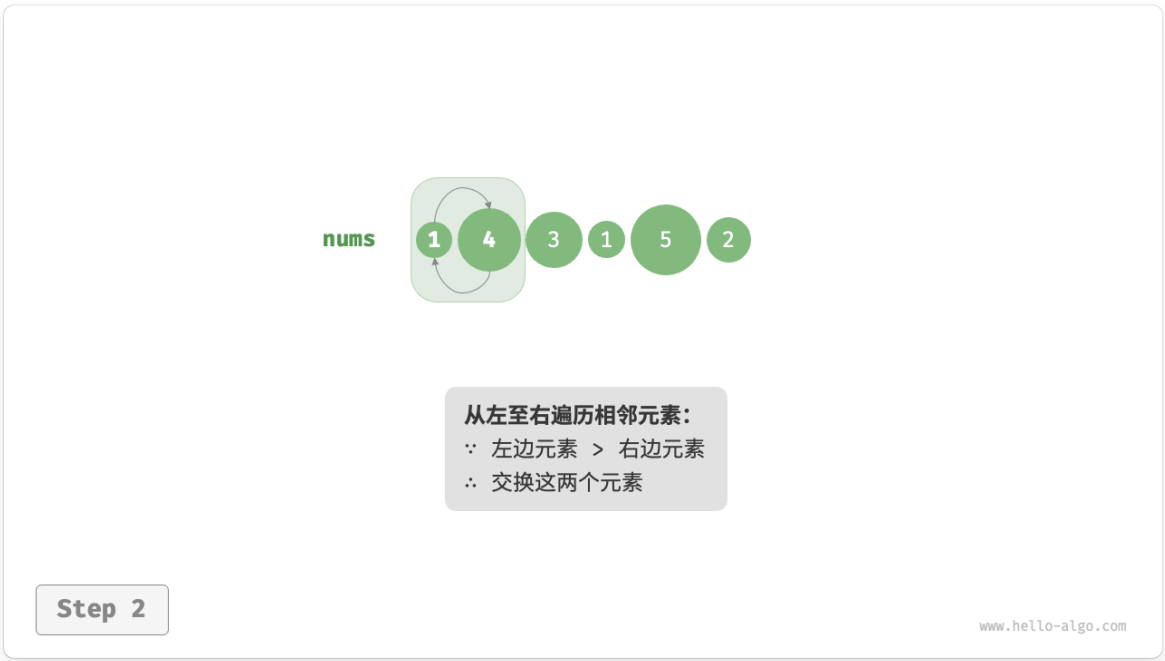
冒泡排序（bubble sort）通过连续地比较与交换相邻元素实现排序。这个过程就像气泡从底部升到顶部一样，因此得名冒泡排序。

如图 11-4 所示，冒泡过程可以利用元素交换操作来模拟：从数组最左端开始向右遍历，依次比较相邻元素大小，如果“左元素 > 右元素”就交换二者。遍历完成后，最大的元素会被移动到数组的最右端。

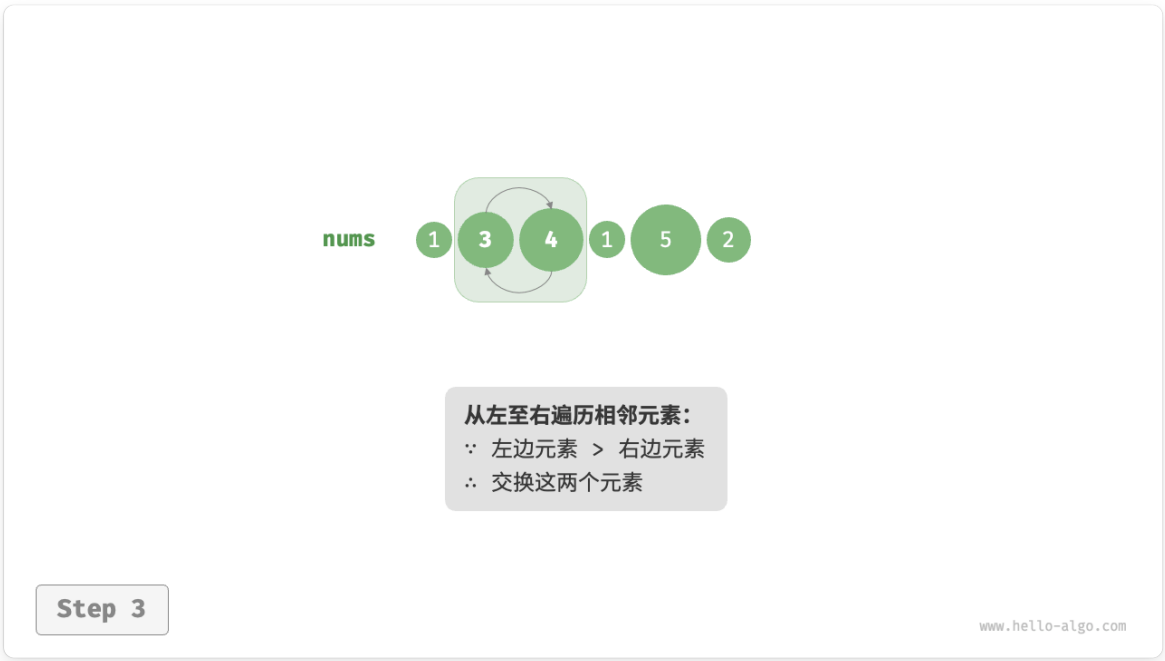
<1>



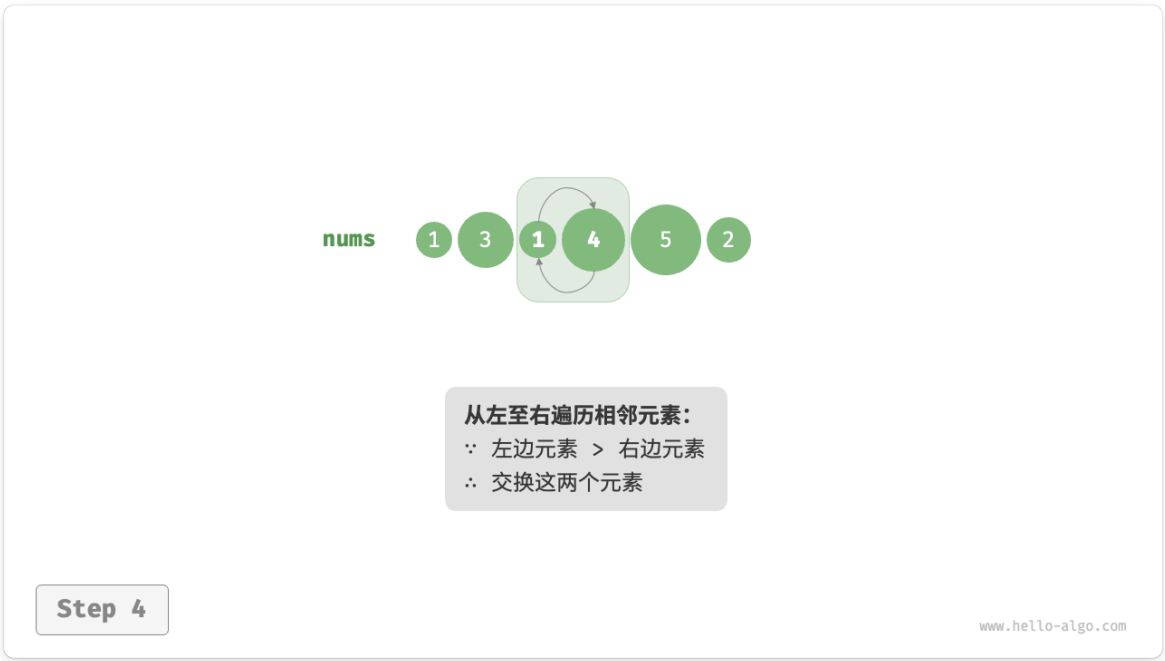
<2>


nums
1
4
3
1
5
2
从左至右遍历相邻元素：
∵ 左边元素 > 右边元素
∴ 交换这两个元素
Step 2
www.hello-algo.com

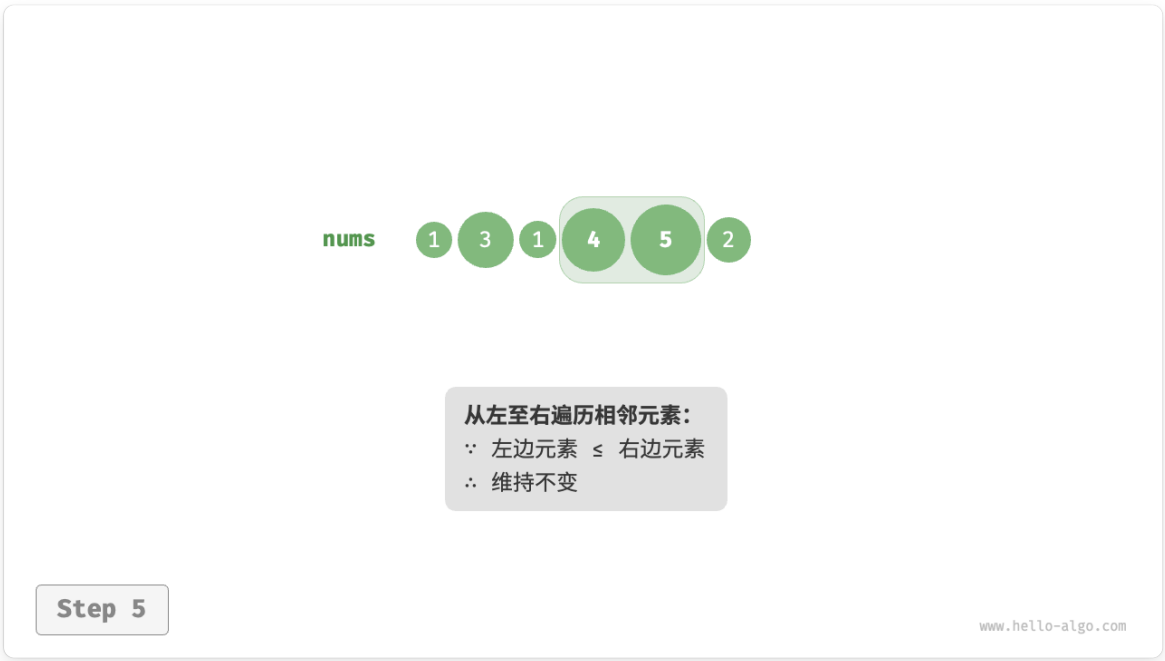
<3>


nums
1
3
4
1
5
2
从左至右遍历相邻元素：
∵ 左边元素 > 右边元素
∴ 交换这两个元素
Step 3
www.hello-algo.com

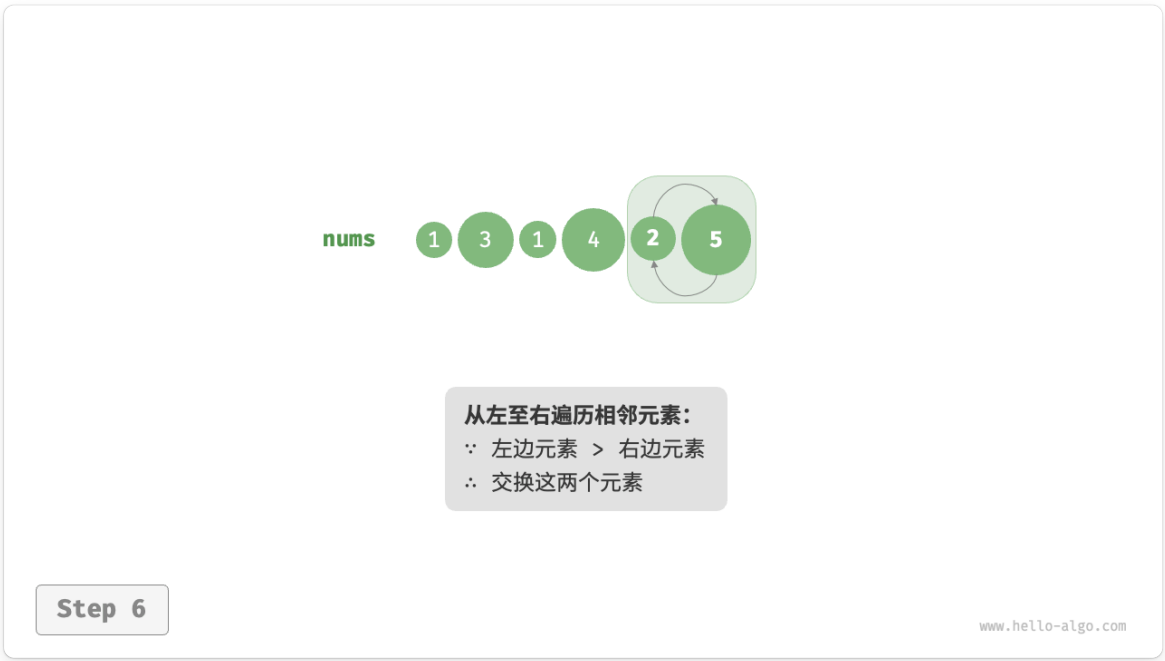
<4>


nums
1
3
1
4
5
2
从左至右遍历相邻元素：
∵ 左边元素 > 右边元素
∴ 交换这两个元素
Step 4
www.hello-algo.com

<5>


nums
1
3
1
4
5
2
从左至右遍历相邻元素：
∵ 左边元素 ≤ 右边元素
∴ 维持不变
Step 5
www.hello-algo.com

<6>



<7>

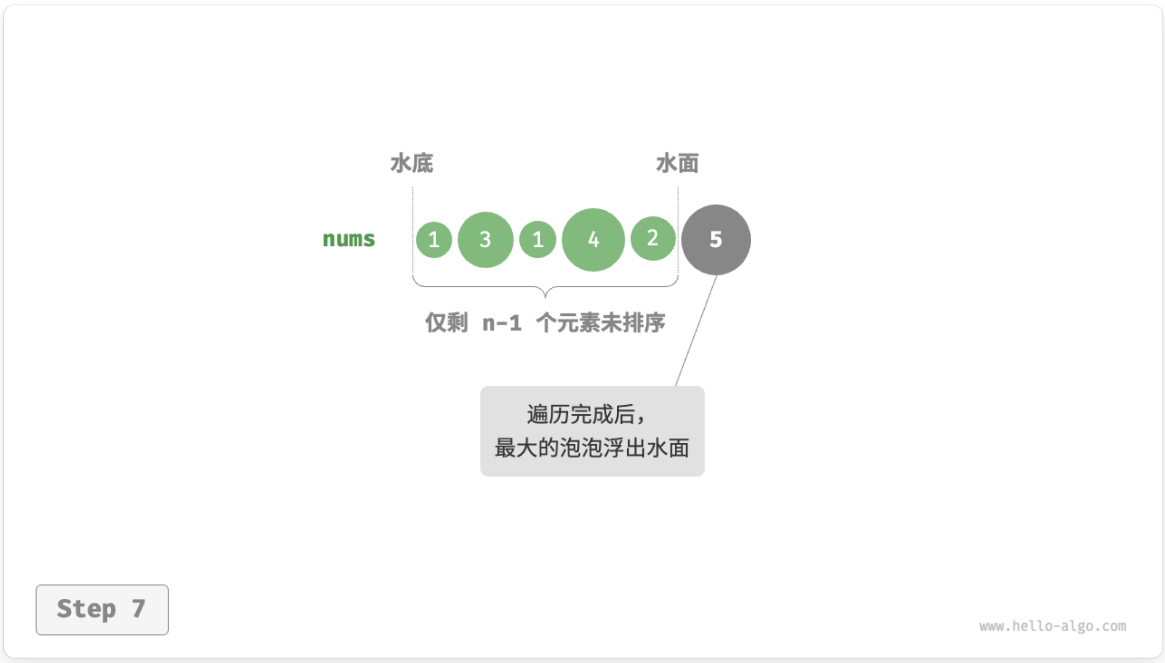


图 11-4 利用元素交换操作模拟冒泡

11.3.1 算法流程

设数组的长度为 n ，冒泡排序的步骤如图 11-5 所示。

1. 首先，对 n 个元素执行“冒泡”，**将数组的最大元素交换至正确位置**。
2. 接下来，对剩余 $n - 1$ 个元素执行“冒泡”，**将第二大元素交换至正确位置**。

3. 以此类推，经过 $n - 1$ 轮“冒泡”后，**前 $n - 1$ 大的元素都被交换至正确位置**。

4. 仅剩的一个元素必定是最小元素，无须排序，因此数组排序完成。

图 11-5 冒泡排序流程

示例代码如下：

Python

bubble_sort.py

```python
def bubble_sort(nums: list[int]):
    """冒泡排序"""
    n = len(nums)
    # 外循环：未排序区间为 [0, i]
    for i in range(n - 1, 0, -1):
        # 内循环：将未排序区间 [0, i] 中的最大元素交换至该区间的最右端
        for j in range(i):
            if nums[j] > nums[j + 1]:
                # 交换 nums[j] 与 nums[j + 1]
                nums[j], nums[j + 1] = nums[j + 1], nums[j]
```

C++

bubble_sort.cpp

```cpp
/* 冒泡排序 */
void bubbleSort(vector<int> &nums) {
    // 外循环：未排序区间为 [0, i]
    for (int i = nums.size() - 1; i > 0; i--) {
        // 内循环：将未排序区间 [0, i] 中的最大元素交换至该区间的最右端
        for (int j = 0; j < i; j++) {
            if (nums[j] > nums[j + 1]) {
                // 交换 nums[j] 与 nums[j + 1]
                // 这里使用了 std::swap() 函数
                swap(nums[j], nums[j + 1]);
            }
        }
```

可视化运行

全屏观看 >

11.3.2 效率优化

我们发现，如果某轮“冒泡”中没有执行任何交换操作，说明数组已经完成排序，可直接返回结果。因此，可以增加一个标志位 `flag` 来监测这种情况，一旦出现就立即返回。

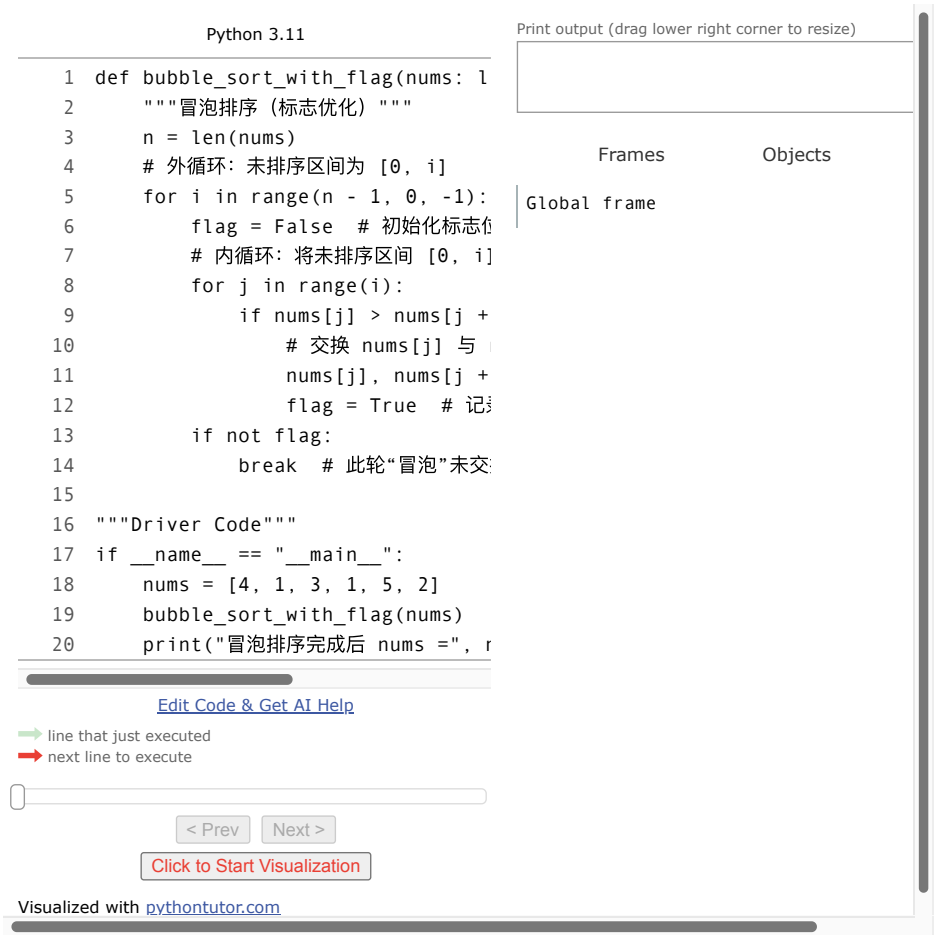
经过优化，冒泡排序的最差时间复杂度和平均时间复杂度仍为 $O(n^2)$；但当输入数组完全有序时，可达到最佳时间复杂度 $O(n)$。

Python

bubble_sort.py

```python
def bubble_sort_with_flag(nums: list[int]):
    """冒泡排序（标志优化）"""
    n = len(nums)
    # 外循环：未排序区间为 [0, i]
    for i in range(n - 1, 0, -1):
        flag = False  # 初始化标志位
        # 内循环：将未排序区间 [0, i] 中的最大元素交换至该区间的最右端
        for j in range(i):
            if nums[j] > nums[j + 1]:
                # 交换 nums[j] 与 nums[j + 1]
                nums[j], nums[j + 1] = nums[j + 1], nums[j]
                flag = True  # 记录交换元素
        if not flag:
            break  # 此轮“冒泡”未交换任何元素，直接跳出
```

可视化运行



全屏观看 >

11.3.3 算法特性

- **时间复杂度为 $O(n^2)$、自适应排序**：各轮"冒泡"遍历的数组长度依次为 $n-1$、$n-2$、…、2、1，总和为 $(n-1)n/2$。在引入 `flag` 优化后，最佳时间复杂度可达到 $O(n)$。
- **空间复杂度为 $O(1)$、原地排序**：指针 i 和 j 使用常数大小的额外空间。
- **稳定排序**：由于在"冒泡"中遇到相等元素不交换。

上一页 11.2 选择排序
下一页 11.4 插入排序

欢迎在评论区留下你的见解、问题或建议