

冒泡排序Bubble Sort

• 交换排序

- 相邻元素两两比较大小，有必要则交换
- 元素越小或越大，就会在数列中慢慢的交换并“浮”向顶端，如同水泡咕嘟咕嘟往上冒

□ 初始	1 9 8 5 6 7 4 3 2		
□ 第一趟	1 8 9 5 6 7 4 3 2	1 8 5 9 6 7 4 3 2	1 8 5 6 9 7 4 3 2
	1 8 5 6 7 9 4 3 2	1 8 5 6 7 4 9 3 2	1 8 5 6 7 4 3 9 2
	1 8 5 6 7 4 3 2 9		
□ 第二趟	1 8 5 6 7 4 3 2 9	1 5 8 6 7 4 3 2 9	1 5 6 8 7 4 3 2 9
	1 5 6 7 8 4 3 2 9	1 5 6 7 4 8 3 2 9	1 5 6 7 4 3 8 2 9
	1 5 6 7 4 3 2 8 9		
□ 第三趟	1 5 6 7 4 3 2 8 9	1 5 6 4 7 3 2 8 9	1 5 6 4 3 7 2 8 9
	1 5 6 4 3 2 7 8 9		
□ 第四趟	1 5 6 4 3 2 7 8 9	1 5 4 6 3 2 7 8 9	1 5 4 3 6 2 7 8 9
	1 5 4 3 2 6 7 8 9		
□ 第五趟	1 5 4 3 2 6 7 8 9	1 4 5 3 2 6 7 8 9	1 4 3 5 2 6 7 8 9
	1 4 3 2 5 6 7 8 9		
□ 第六趟	1 4 3 2 5 6 7 8 9	1 3 4 2 5 6 7 8 9	1 3 2 4 5 6 7 8 9
□ 第七趟	1 3 2 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	
□ 第八趟	1 2 3 4 5 6 7 8 9		

核心算法

- 排序算法，一般都实现为就地排序，输出为升序
- 扩大有序区，减小无序区。图中红色部分就是增大的有序区，反之就是减小的无序区
- 每一趟比较中，将无序区中所有元素依次两两比较，升序排序将大数调整到两数中的右侧
- 每一趟比较完成，都会把这一趟的最大数推倒当前无序区的最右侧

基本实现

思考时，将问题规模减小，一般2次不一定能找到规律，3次基本上可以看出规律。所以，我们思考时认为列表里面就只有4个元素。

```
1  nums = [1, 9, 8, 5] # 数字少好思考
2  print(nums)
3  length = len(nums) # 4
4  # 第一趟
5  i = 0
6
7  # 本趟内两两比较，大数换到右边
8  # 2个数比1下，3个数比2下，那么比较次数就是当前比较数个数-1
9  for j in range(length-1): # j 为0, 1, 2
10     print(j, nums[j], nums[j+1])
11     if nums[j] > nums[j+1]: # 只有大于才交换，小于等于就不用了
12         temp = nums[j]
13         nums[j] = nums[j+1]
14         nums[j+1] = temp
15
16 print(nums)
```

上面代码已经基本上完成了核心比较交换算法。下面解决每一趟的问题。

每一趟无序区是减小1个的，所以考虑使用两个循环，外面 i 循环表示趟，里面 j 循环表示每一趟的两两比较。内层 j 循环正好可以使用外层的 i，用在 range(length-1-i)，因为 i 第一次为 0 相当于没有减，第二次就是 1 了，这里 range 当计数器用表示走几趟。

i 循环控制趟数，4 个数比较 3 趟就可以了。

```

1  nums = [1, 9, 8, 5] # 数字少好思考
2  #nums = [1, 9, 8, 5, 6, 7, 4, 3, 2]
3  print(nums)
4  length = len(nums) # 4
5
6  # i 控制趟数
7  for i in range(length-1):
8      # 本趟内两两比较，大数换到右边
9      # 2个数比1下，3个数比2下，那么比较次数就是当前比较数个数-1
10     for j in range(length-1-i): # j 为0, 1, 2
11         print(j, nums[j], nums[j+1])
12         if nums[j] > nums[j+1]: # 只有大于才交换，小于等于就不用了
13             temp = nums[j]
14             nums[j] = nums[j+1]
15             nums[j+1] = temp
16     print(nums)
17
18 print('result = ', nums)

```

可以增加两个变量：count 表示比较的次数，count_swap 表示交换的次数

```

1  nums = [1, 9, 8, 5] # 数字少好思考
2  #nums = [1, 9, 8, 5, 6, 7, 4, 3, 2]
3  print(nums)
4  length = len(nums) # 4
5  count = 0
6  count_swap = 0
7  # i 控制趟数
8  for i in range(length-1):
9      # 本趟内两两比较，大数换到右边
10     # 2个数比1下，3个数比2下，那么比较次数就是当前比较数个数-1
11     for j in range(length-1-i): # j 为0, 1, 2
12         count += 1
13         if nums[j] > nums[j+1]: # 只有大于才交换，小于等于就不用了
14             temp = nums[j]
15             nums[j] = nums[j+1]
16             nums[j+1] = temp
17             count_swap += 1
18     print(nums)
19
20 print('result = ', nums)
21 print(count, count_swap)

```

优化

思考：如果某一趟两两比较后没有发生任何交换，说明什么？

```
1 #nums = [1, 9, 8, 5] # 数字少好思考
2 #nums = [1, 9, 8, 5, 6, 7, 4, 3, 2]
3 nums = [9, 8, 1, 2, 3, 4, 5, 6, 7]
4 print(nums)
5 length = len(nums) # 4
6 count = 0
7 count_swap = 0
8 # 假设这一趟不需要交换了
9 finished = True # 定义标记
10
11 # i 控制趟数
12 for i in range(length-1):
13     # 本趟内两两比较，大数换到右边
14     # 2个数比1下，3个数比2下，那么比较次数就是当前比较数个数-1
15     for j in range(length-1-i): # j 为0, 1, 2
16         count += 1
17         if nums[j] > nums[j+1]: # 只有大于才交换，小于等于就不用了
18             temp = nums[j]
19             nums[j] = nums[j+1]
20             nums[j+1] = temp
21             count_swap += 1
22             finished = False # 有一次交换就要标记为False
23     if finished:
24         break
25     print(nums)
26
27 print('result = ', nums)
28 print(count, count_swap)
```

上面代码合适吗？

不合适。因为假设的是每一趟，只要有一趟没有发生交换，就可以认为已经是目标顺序了。要把标记放在 i 循环里。

```
1 #nums = [1, 9, 8, 5] # 数字少好思考
2 #nums = [1, 9, 8, 5, 6, 7, 4, 3, 2]
3 nums = [9, 8, 1, 2, 3, 4, 5, 6, 7]
4 print(nums)
5 length = len(nums) # 4
6 count = 0
7 count_swap = 0
8
9 # i 控制趟数
10 for i in range(length-1):
11     # 假设这一趟不需要交换了
12     finished = True # 定义标记
13     # 本趟内两两比较，大数换到右边
14     # 2个数比1下，3个数比2下，那么比较次数就是当前比较数个数-1
15     for j in range(length-1-i): # j 为0, 1, 2
16         count += 1
17         if nums[j] > nums[j+1]: # 只有大于才交换，小于等于就不用了
18             temp = nums[j]
```

```
19         nums[j] = nums[j+1]
20         nums[j+1] = temp
21         count_swap += 1
22         finished = False # 有一次交换就要标记为False
23     if finished:
24         break
25     print(nums)
26
27 print('result = ', nums)
28 print(count, count_swap)
```

总结

- 冒泡法需要数据一趟趟比较
- 可以设定一个标记判断此轮是否有数据交换发生，如果没有发生交换，可以结束排序，如果发生交换，继续下一轮排序
- 最差的排序情况是，初始顺序与目标顺序完全相反，遍历次数 $1, \dots, n-1$ 之和 $n(n-1)/2$
- 最好的排序情况是，初始顺序与目标顺序完全相同，遍历次数 $n-1$
- 时间复杂度 $O(n^2)$

