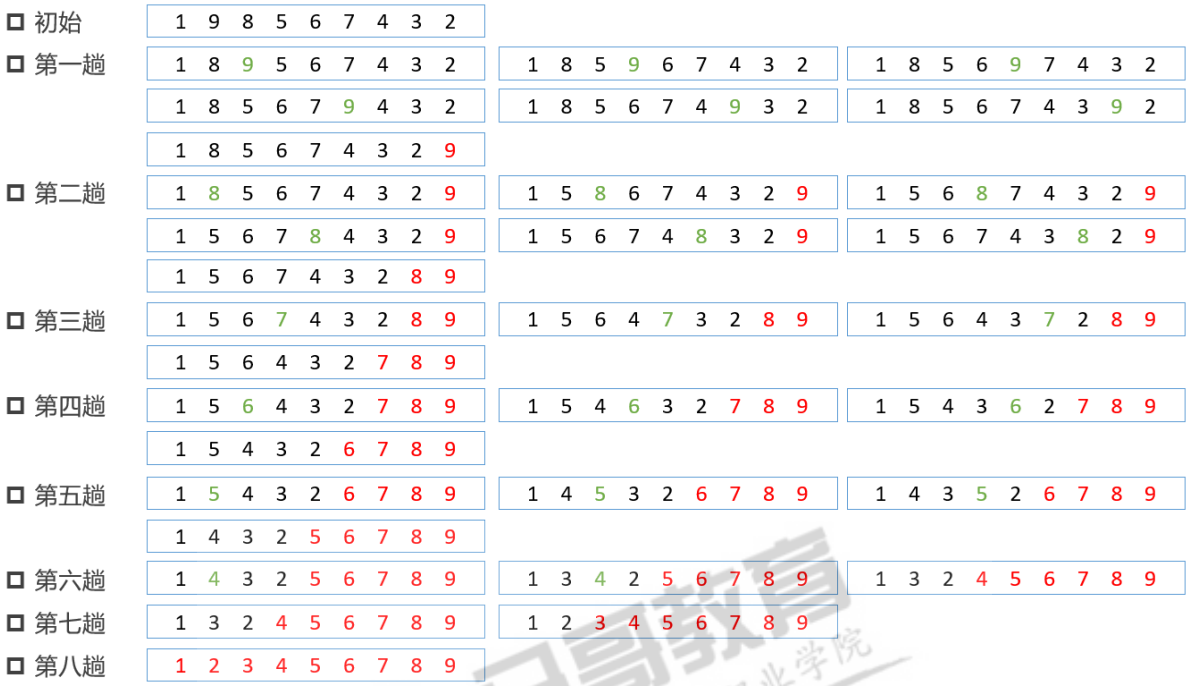
冒泡排序Bubble Sort

- **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



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。

```
nums = [1, 9, 8, 5] # 数字少好思考
print(nums)
length = len(nums) # 4
# 第一趟
i = 0

# 本趟内两两比较，大数换到右边
# 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
for j in range(length-1): # j 为0, 1, 2
    print(j, nums[j], nums[j+1])
    if nums[j] > nums[j+1]: # 只有大于才交换，小于等于就不用了
        temp = nums[j]
        nums[j] = nums[j+1]
        nums[j+1] = temp

print(nums)
```

```
# i取0、1、2试一试
```

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。

```
nums = [1, 9, 8, 5] # 数字少好思考
#nums = [1, 9, 8, 5, 6, 7, 4, 3, 2]
print(nums)
length = len(nums) # 4

# i 控制趟数
for i in range(length-1):
    # 本趟内两两比较，大数换到右边
    # 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
    for j in range(length-1-i): # j 为0, 1, 2
        print(j, nums[j], nums[j+1])
        if nums[j] > nums[j+1]: # 只有大于才交换，小于等于就不用了
            temp = nums[j]
            nums[j] = nums[j+1]
            nums[j+1] = temp
    print(nums)

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

```
nums = [1, 9, 8, 5] # 数字少好思考
#nums = [1, 9, 8, 5, 6, 7, 4, 3, 2]
print(nums)
length = len(nums) # 4
count = 0
count_swap = 0
# i 控制趟数
for i in range(length-1):
    # 本趟内两两比较，大数换到右边
    # 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
    for j in range(length-1-i): # j 为0, 1, 2
        count += 1
        if nums[j] > nums[j+1]: # 只有大于才交换，小于等于就不用了
            temp = nums[j]
            nums[j] = nums[j+1]
            nums[j+1] = temp
            count_swap += 1
    print(nums)

print('result = ', nums)
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？

```
#nums = [1, 9, 8, 5] # 数字少好思考
#nums = [1, 9, 8, 5, 6, 7, 4, 3, 2]
nums = [9, 8, 1, 2, 3, 4, 5, 6, 7]
print(nums)
length = len(nums) # 4
count = 0
count_swap = 0
# 假设这一趟不需要交换了
finished = True # 定义标记

# i 控制趟数
for i in range(length-1):
    # 本趟内两两比较，大数换到右边
    # 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
    for j in range(length-1-i): # j 为0, 1, 2
        count += 1
        if nums[j] > nums[j+1]: # 只有大于才交换，小于等于就不用了
            temp = nums[j]
            nums[j] = nums[j+1]
            nums[j+1] = temp
            count_swap += 1
            finished = False # 有一次交换就要标记为False
    if finished:
        break
    print(nums)

print('result = ', nums)
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。

```
#nums = [1, 9, 8, 5] # 数字少好思考
#nums = [1, 9, 8, 5, 6, 7, 4, 3, 2]
nums = [9, 8, 1, 2, 3, 4, 5, 6, 7]
print(nums)
length = len(nums) # 4
count = 0
count_swap = 0

# i 控制趟数
for i in range(length-1):
    # 假设这一趟不需要交换了
    finished = True  # 定义标记
    # 本趟内两两比较，大数换到右边
    # 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
    for j in range(length-1-i): # j 为0, 1, 2
        count += 1
        if nums[j] > nums[j+1]: # 只有大于才交换，小于等于就不用了
            temp = nums[j]
```

```
            nums[j] = nums[j+1]
            nums[j+1] = temp
            count_swap += 1
            finished = False # 有一次交换就要标记为False
    if finished:
        break
    print(nums)

print('result = ', nums)
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,...,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)