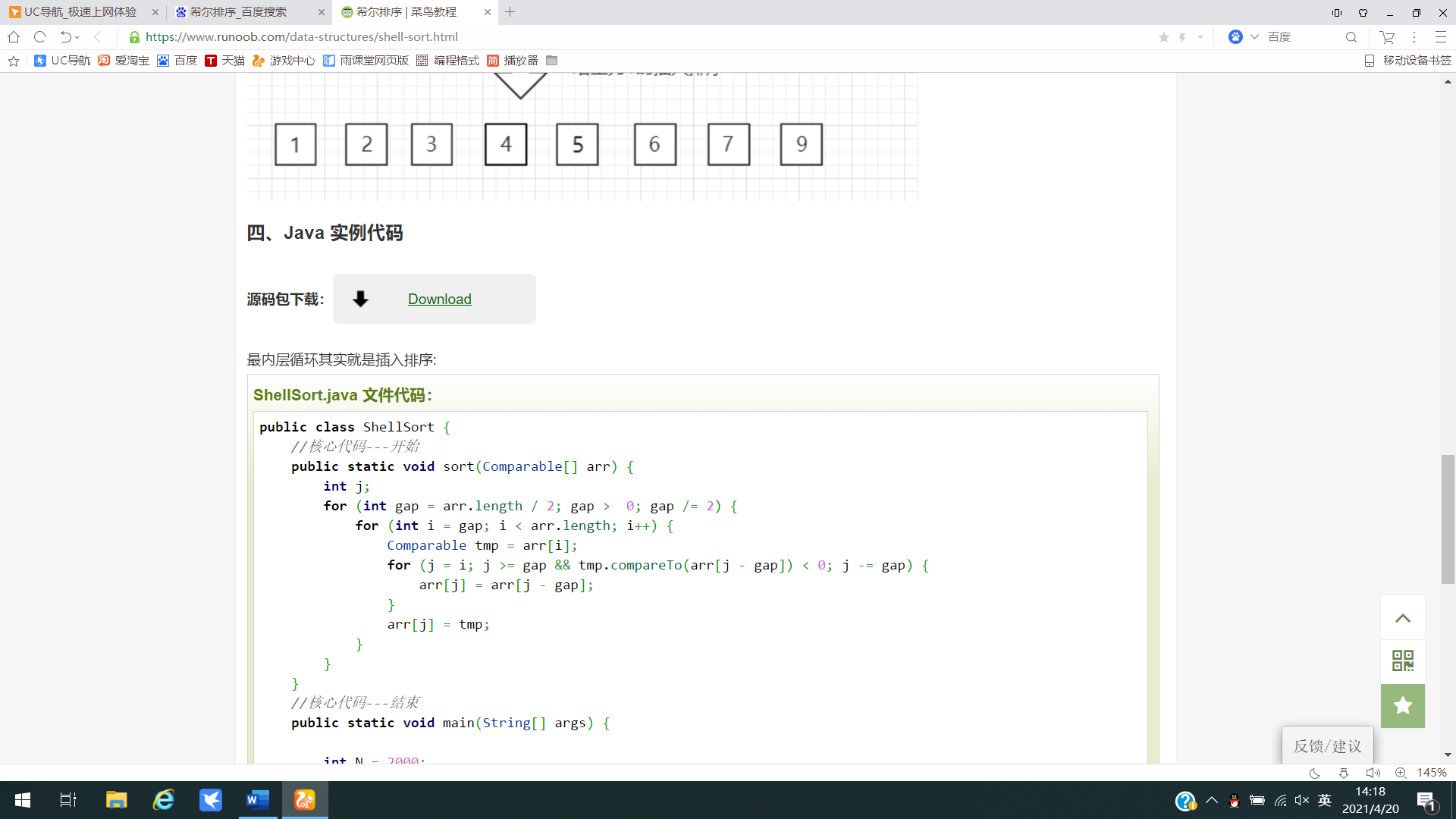
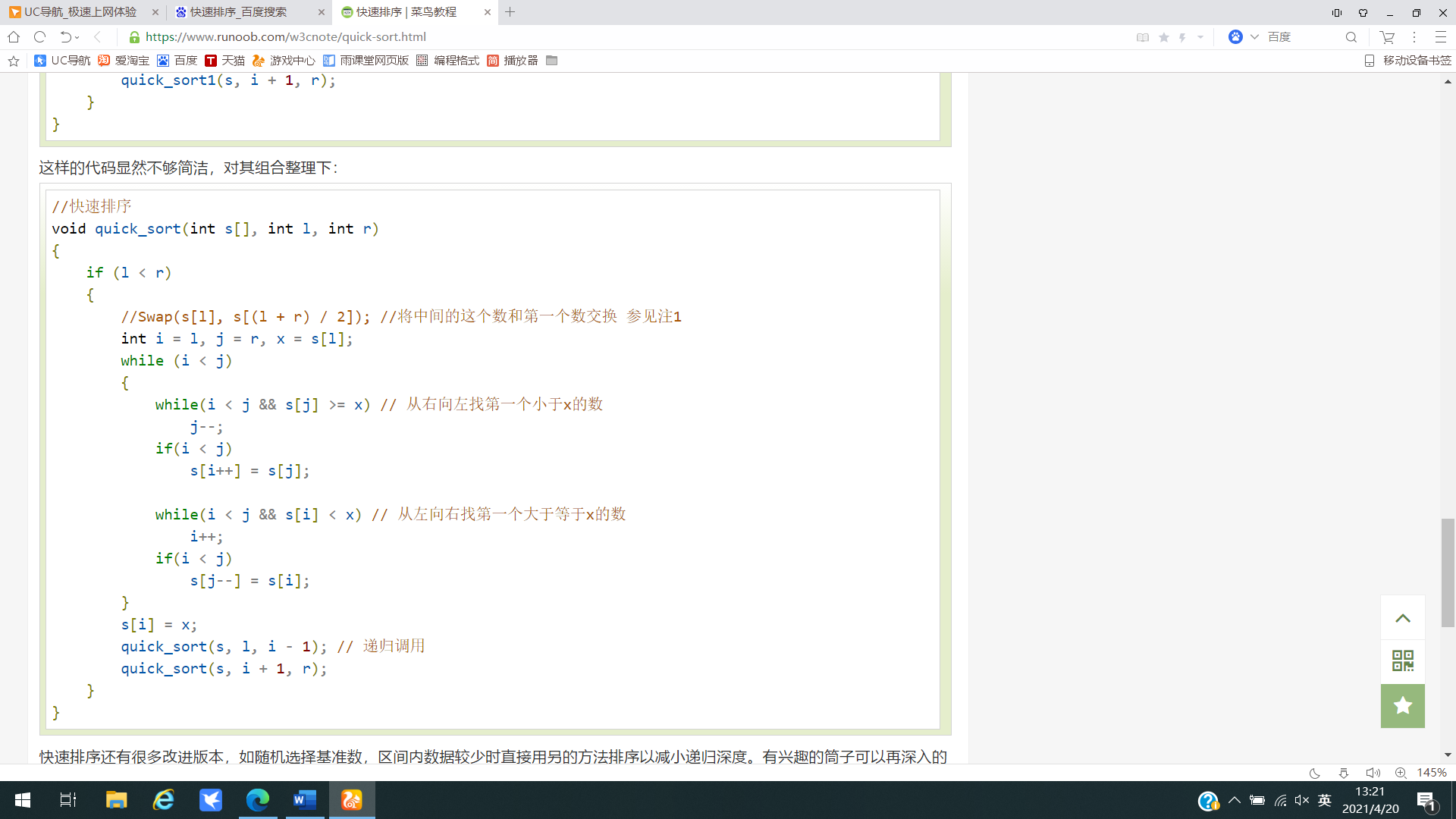
* **排序算法：**

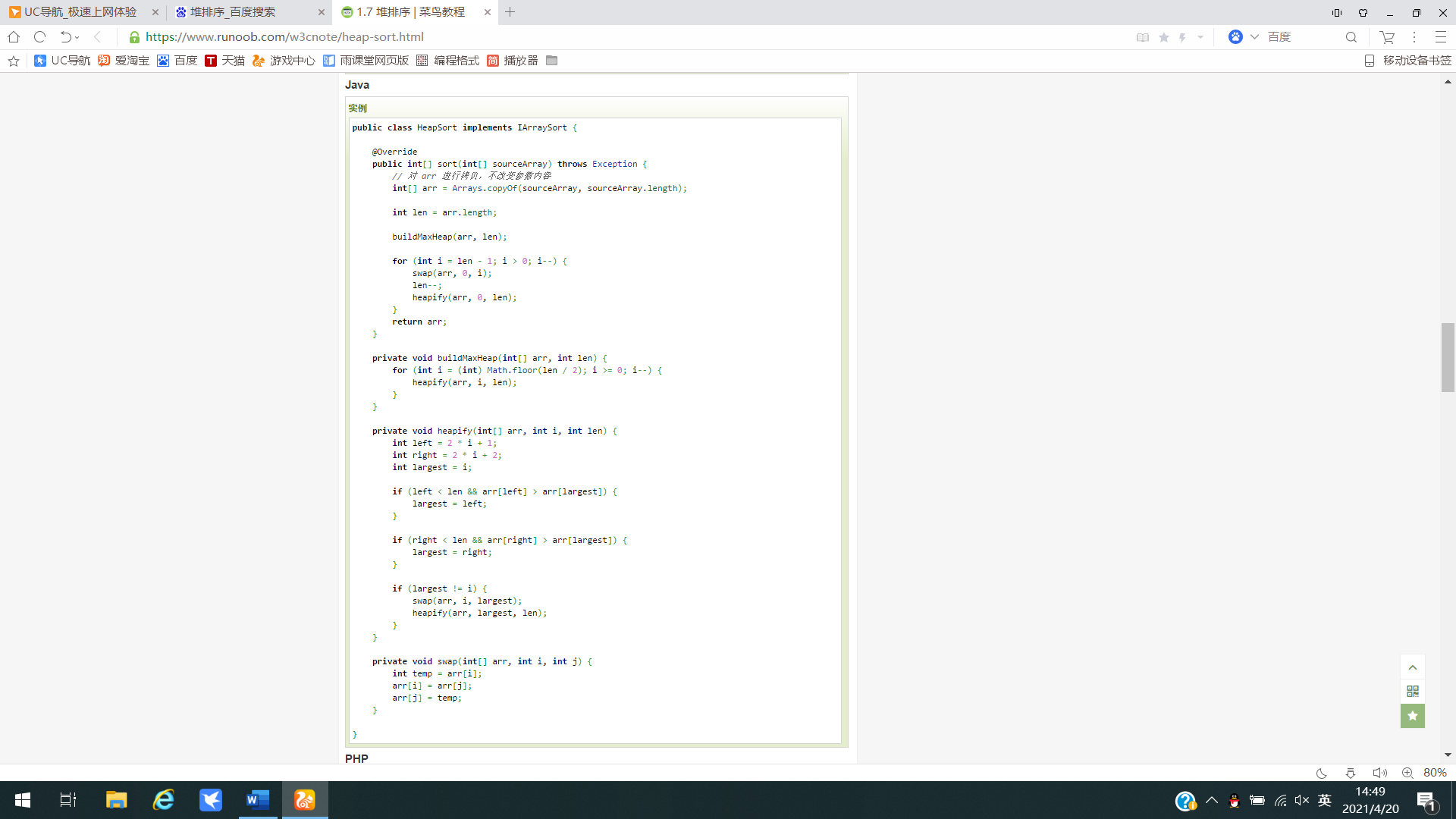
1. 直接插入排序：“哨兵”减少一层if语句
2. 希尔排序：分组的插入排序 希尔排序无法使用 “哨兵”。



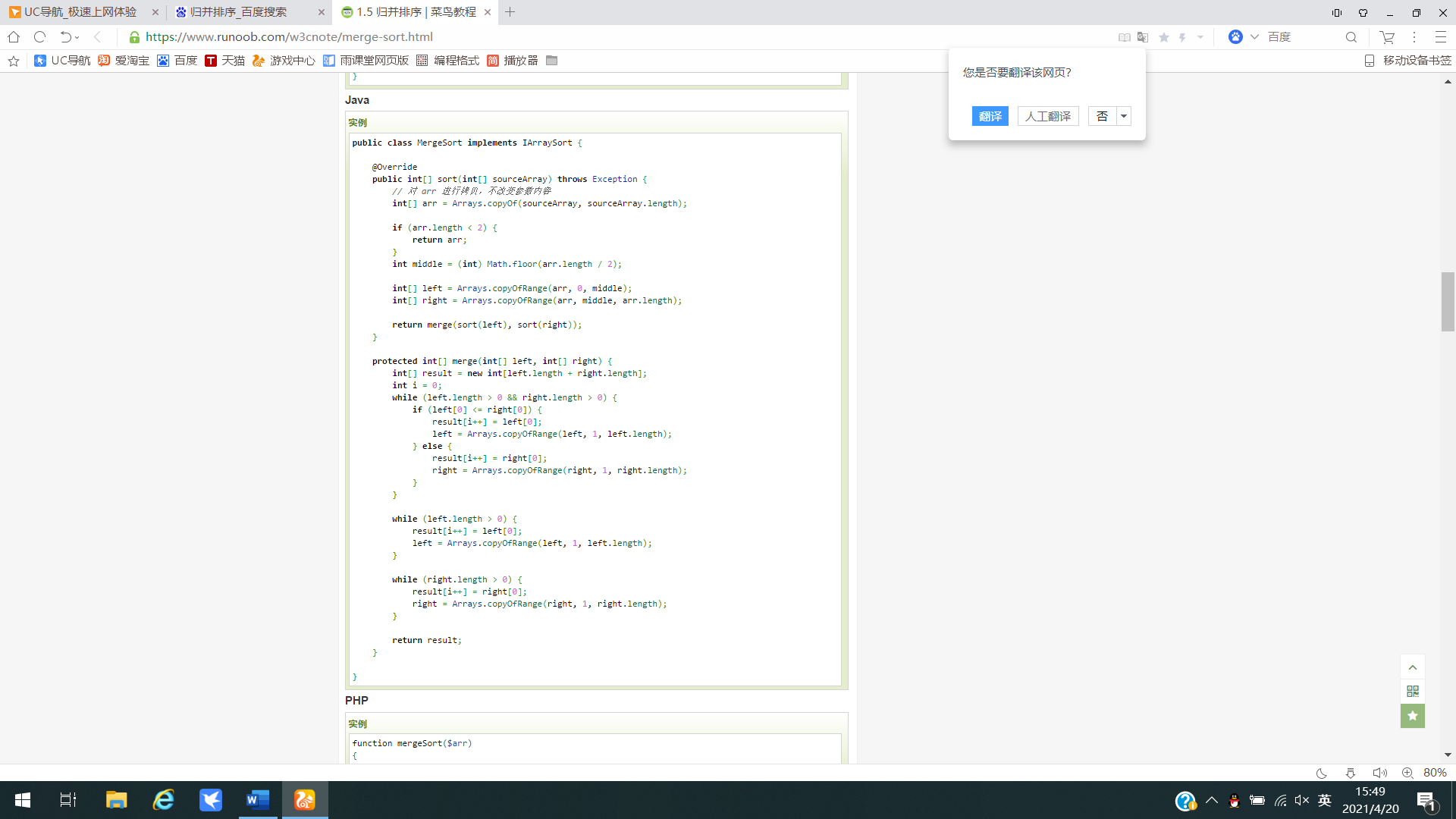
1. 冒泡排序：优化：boolean flag = true;
2. 快速排序：给“pivot”确定合适的位置，再把待排序列分成两部分，重复此操作。



1. 简单选择排序：选最小元素交换
2. 堆排序：【注：代码中的copyof没有意义】

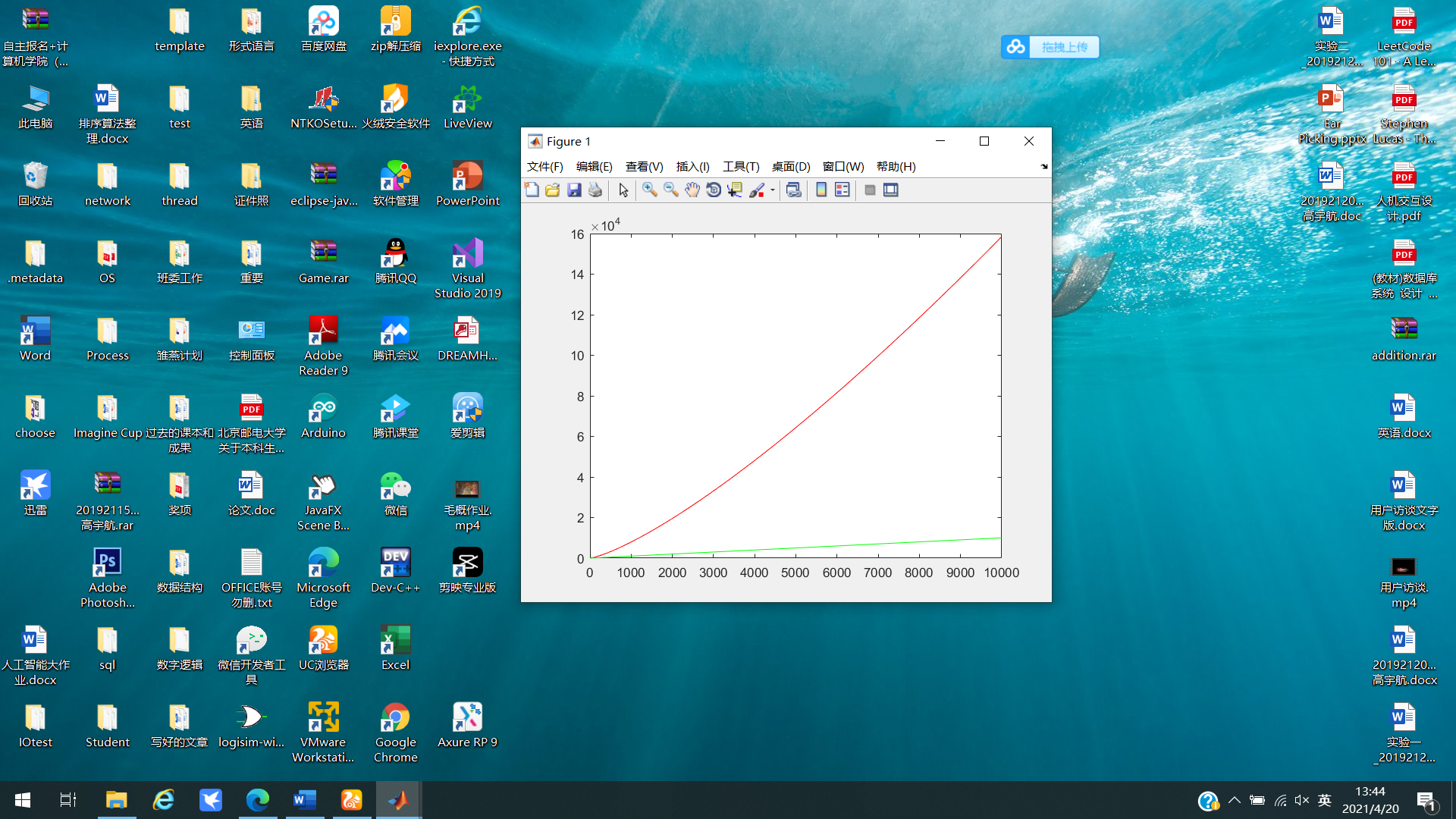


1. 归并排序：【注：代码中的copyof没有意义】



* **常见算法平均时间复杂度：**

1. 直接插入排序：O(n^2) 最好情况：顺序 O(n-1)
2. 希尔排序：O(n^1.3) 希尔排序好于n^2的排序法，但是比n\*log2(n) 的排序法表现差。



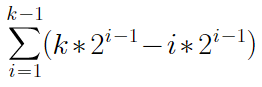
1. 冒泡排序：O(n^2)
2. 快速排序：O(log2n\*n) 最坏情况是逆序情况，相当于冒泡排序，时间复杂度为O(n^2)
3. 简单选择排序：O(n^2)
4. 堆排序：O(log2n\*n)
5. 初始化堆的时间复杂度分析

设元素个数为n，则堆的高度k=log（n+1）≈log n，非叶子结点的个数为2^（k-1）-1

假设每个非叶子结点都需要进行调整，则第i层的非叶子结点需要的操作次数为k-i，

第i层共有2^（i-1）个结点，则第i层的所有结点所做的操作为k\*2^（i-1）- i\*2^（i-1），

共k-1层非叶子结点，总的操作次数为



化简可得，上式=2^k-k+1，将k=log（n+1）≈log n代入，得n - log n +1，

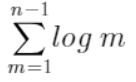
所以，初始化堆的复杂度为O(n)

1. 调整堆的时间复杂度分析

假设根节点和排在最后的序号为m的叶子结点交换，并进行调整，那么调整的操作次数 =

原来m结点所在的层数 = 堆的高度（因为m结点在堆的最后）= log m

共n个结点，调整的总操作次数为



化简可得，上式=log (n-1)! ≈ n\*log n 所以，调整堆的复杂度为O(n\*log n)

因此，总体复杂度为O(n\*log n)

1. 归并排序：O(log2n\*n)

* **常见算法空间复杂度：**

1. 直接插入排序：O(1)
2. 希尔排序：O(1)
3. 冒泡排序：O(1)
4. 快速排序：O(log2n)：要栈辅助实现递归 所以最坏情况是O(n)
5. 简单选择排序：O(1)
6. 堆排序：O(1)
7. 归并排序：O(n) 实际是O(log2n)（栈）+ O(n)（merge的空白数组）

* **常见排序算法稳定性：**

只有直接插入算法、冒泡算法和归并算法是稳定的。

* **堆排序、快速排序、归并排序比较：**

**1.如果想选择一种稳定的算法，只能选择归并算法**

**2.如果选择一种节省空间的算法，那么选择堆排序**

**3.快速排序在最坏情况下时间复杂度会退化**