对于数组[1,11,2,9,4,8,7,5,17,5,6,3]进行排序（假定升序）

假设数组长度为M

附上：java求最大值的3种方法。<https://www.idaobin.com/archives/297.html>

相邻两两之间比较（冒泡），或都与初始位置的值比较（初始位置的值会变，位置索引不变）

希尔排序（shell sort）

“增强版”插入排序。

思想：

分组N：先将数组，分成N个子数组（长度一致，为len）,对于0 to len-1的每个元素（即索引位置）i，对N0（i）、N1（i）、N2(i)…Nn(i)进行“插入排序”，得到排序数组array1——————也即这N个子数组，取出所有位置相同的元素，作“插入排序”，进行M/N次该操作。————第一次插入排序，会导致比较“有序”，小的大致在前面，大的大致在后面———

分组N/k（此时深度为logk(N) ）然后，缩小分组，如N/2（N/3等都可以，也可以是0 to N-1之间的任意数，选N/2这种会快一点逼近1，也即进行log2(N)次上述操作后，基本上逼近到1）——继续对array1进行（分组N中的操作，即分组并对每组“插入排序”）得到arry2.

…

一直到分组为1，对整个数组，做“插入排序”————此时的整个数组“有序”多了，数组交换时间减少很多。

总结：“希尔”排序，就是

分组、组间“插入排序”（不是组内）——————先分组（组数很大），对组间“插入排序”

举例：[1,11,2,9,4,8,7,5,17,5,6,3]。

1. 先分4组，得到：子数组

[1,11,2,9]

[4,8,7,5]

[17,5,6,3]

对每列进行分组排序得到array1：

1——5——2——3

4——8——6——5

17——11——7——9

此时array1=[1,5,2,3, 4,8,6,5 ,17,11,7,9 ]

1. 分组4/2=2组，对array1排序，得到子数组:

[1,5]

[2,3]

[4,8]

[6,5]

[17,11]

[7,9]

排序得到：

1——3

2——5

4——5

6——8

7——9

17——11

得到array2: [1,3,2,5,4,5,6,8,7,9,17,11]

1. 最后分组2/2=1，直接插入排序：

得到：[1,2,3,4,5,5,6,7,8,9,11,17]

冒泡排序（bubble sort）

<https://www.cnblogs.com/RainyBear/p/5258483.html>

原理：

**算法步骤**：

1）比较相邻的元素。如果第一个比第二个大，就交换他们两个。

2）对每一对相邻元素作同样的工作，从开始第一对到结尾的最后一对。这步做完后，最后的元素会是最大的数。

3）针对所有的元素重复以上的步骤，除了最后一个。

4）持续每次对越来越少的元素重复上面的步骤，直到没有任何一对数字需要比较。

简言之：第1，2步能取出“最大值”（只不过采用的是两两比较，从某个索引处开始，数字大的往后面“冒泡”，被交换位置）

第3,4步，就是找出除最大值（排完序得到的最大值序列），然后对剩余“未排序”序列，使用“冒泡”排序手段——即相邻间比较。

“冒泡”排序和“选择”排序的区别： 选出“最小（大）值”的手段不一样。

选择排序：定义一个初始min（最小值索引位置）， 将其与数组其他每一个元素比较，若大于后者，则交换————交换到这个min位置来。

冒泡排序：第0个与第1个比较，满足交换条件，交换； 第1个与第2个，同上，…,第i与i+1个，同上，结束

快速排序：quick sort

见”数据结构与算法”的p201

quick sort步骤：

1．选取标杆值（pivot）

2.一次排序过程： 小于pivot值的放在pivot左边（子数组small），大于pivot值的放在pivot右边(子数组large)——得到两个子数组small和large

对两个子数组分别“递归”使用 quck sort

关键在于弄懂一次排序过程。

1. 移动，找出移动不了的值，直接交换（使其符合移动条件）——一直找，直到i和j游标碰到一起或者交错、再将pivot和i交换。

重点：

枢纽元的选取：-----随机选取3数中值分隔法 （左端、右端、正中间位置这3个数，取第2大值）——————最好不选最左端、最右端、或随机值。

另外，如果数组里面包含重复元素， 排序时，还要细化（和pivot相等的重复元素分到小组、大组的比例，一般各0.5最好）