

C 1. 已知数据序列 (8, 9, 10, 4, 5, 6, 20, 1, 2) 是某种排序算法第二趟排序后得到的结果，则该算法可能是

一定不是因为冒泡一趟后 max/min 已到达应在位置

若为选排，则每一趟都能找到剩下元素中 max/min，应与冒泡类似。前一个已经是整个数组中的 min 值，之后不会再变的那个中

☒ A. 选择排序

☒ B. 起泡排序

☒ C. 插入排序

☐ D. 堆排序

同冒泡，选择类似，第一趟之后

2. 设使用某种排序方法对数据序列进行排序，两趟排序后得到结果是 (8, 9, 10, 4, 5, 6, 20, 1, 2)，则该排序方法只能是

A. 选择排序

B. 起泡排序

C. 插入排序

D. 堆排序

第一趟时，前几个已排好

最后一趟已是最终 max(min) 结果

3. 下列选项中，每一趟都能选出一个元素放在其最终位置上，并且是稳定的排序算法是

☒ A. 起泡排序

☒ B. 希尔排序

☒ C. 直接选择排序

☒ D. 快速排序

稳定

不在最终位置

能到最终位置

4. 采用递归方式对顺序表进行快速排序，下列关于递归次数的叙述中，正确的是

A. 递归次数与初始数据的排列次序无关

B. 每次划分后，先处理较长的分区可以减少递归次数

C. 每次划分后，先处理较短的分区可以减少递归次数

☒ D. 递归次数与每次划分后得到的分区处理顺序无关

⑨ 8 7 6 5 4

4 5 6 7 8 9

5. 已知两个长度分别为 m 和 n 的升序链表，若将他们合并成一个长度为 m+n 的降序链表，则最坏情况下的时间复杂度是

A.  $O(n)$

B.  $O(m \cdot n)$

C.  $O(\min(m, n))$

D.  $O(\max(m, n))$

1 2 3 ; 4 5 6 7 8 min(m, n) + 逆序 O(m+n)

1 2 3 4 5 ; 6 7 8 max(m, n)

6. 一个长度为 L ( $L \geq 1$ ) 的升序序列 S，处在第  $\lceil L/2 \rceil$  个位置的数称为 S 的中位数。例如，若序列  $S_1 = (11, 13, 15, 17, 19)$ ，则  $S_1$  的中位数是 15。两个序列的中位数是含它们所有元素的升序序列的中位数。例如，若  $S_2 = (2, 4, 6, 8, 20)$ ，则  $S_1$  和  $S_2$  的中位数是 11。现有两个等长升序序列 A 和 B，试设计一个在时间和空间两方面都尽可能高效的算法，找出两个序列 A 和 B 的中位数。

1) 若  $a=b$ ，则 a 或 b 即为所求中位数，算法结束。

2) 若  $a < b$ ，则舍弃序列 A 中较小的一半，同时舍弃序列 B 中较大的一半，要求两次舍弃的长度相等。

3) 若  $a > b$ ，则舍弃序列 A 中较大的一半，同时舍弃序列 B 中较小的一半，要求两次舍弃的长度相等。

在保留的两个升序序列中，重复过程 1)、2)、

3)，直到两个序列中均只含一个元素时为止，上面所求的中位数较小者即为所求的中位数。