1. 下面的排序算法中哪些是稳定的：插入排序、归并排序、堆排序、快速排序和计数排序？给出一个能使任何排序算法都稳定的方法。你所给出的方法带来的额外时间和空间开销是多少？

插入排序、归并排序、计数排序。

另外开辟一个大小与待排序的元素个数相同的数组，用于存放每个元素在原始数组中的序号，当排序过程中若发现两个元素的值相同时则比较其在原始数组中的序号。

总时间开销为原来的时间的两倍，额外时间开销为原来所花的时间，额外空间开销为O(n)。

2. 假设所有元素都是互异的，说明在最坏情况下。如何使快速排序的运行时间为 O(nlogn)。

将每次序列的中位数设为快速排序的枢轴即可。其中，用最坏情况为线性时间的选择算法来找中位数。这样，在最坏情况下，

$T(n) = 2T(n/2) + \Theta(n) + cn = 2T(n/2) + \Theta(n) + \Theta(n) = 2T(n/2) + \Theta(n)$

$\Rightarrow T(n) = O(n\log n)$

3. 给定一个整数数组，其中不同的整数所包含的数字的位数可能不同。但该数组中，所有整数中包含的总数字位数为 *n*。设计算法使其可以在 *O*(*n*) 时间内对该数组进行排序。

算法思路：

1）声明一个ListNode类型vector容器来作为桶，其中ListNode为一个结构体，桶的大小根据整数中的最大位数来确定。对每个整数用while循环求出其位数，若除以10^k得到的数大于等于1，则退出循环，该整数的位数为 k+1，将该整数放入第 k+1 个桶内。

2） 分别对每个桶内的整数采用基数排序，基数排序内部对每个位数上的数字采用计数排序。计数排序的过程中，由于每个数字的取值范围为 -9 ~ 9 ，故此处建立了 -9 ~ 9 到 0 ~ 19 的映射，定义了一个大小为19的数组来存储每个数出现的次数。

3）从桶的序号为1开始，依次将桶内整数放到数组里。

时间复杂度分析：

1）对于求每个整数的位数，设某个整数的位数为d，则求其位数所需时间为 Θ(d)，设数组内的第 i 个整数的位数为 d_i，整数的个数为m，由于所有整数中包含的总数字位数为n，故求所有整数的位数所需时间为 $\sum_{i=1}^{m}\Theta(d_i) = \Theta(\sum_{i=1}^{m} d_i) = \Theta(n)$

2）对每个桶中的整数进行基数排序，设 d_i为所排序的桶 i 中的数字的位数，m_i 为桶 i 中所含数字的个数，故对桶 i 的元素进行排序的时间复杂度为 $\Theta(d_i(m_i+k)) = \Theta(d_i(m_i+19)) = \Theta(d_i m_i)$，设所有整数中的最大位数为 d ，则对所有桶中的元素排序的时间复杂度为 $\sum_{i=1}^{d}\Theta(im_i) = \Theta(\sum_{i=1}^{d} im_i) = \Theta(n)$

3）对于将桶内整数放到数组内，所需时间为 Θ(n)

故总时间复杂度为Θ(n) + Θ(n) + Θ(n) = Θ(n) ，故此算法可以在 O(n) 时间内对该数组进行排序。

4. SELECT 算法最坏情况下的比较次数 *T*(*n*) = Θ(*n*)，但是其中的常数项使非常大的。请对其进行优化，使其满足：

• 在最坏情况下的比较次数为 Θ(*n*)。

• 当 *i* 是小于 n/2 的常数时，最坏情况下只需要进行 *n*+*O*(log*n*) 次比较。

1）若 i 是大于 n/2 的常数，直接用 select 算法求解。

2）若 i 是小于 n/2 的常数，则将原始数组划分两半，分别为数组B1、B2，若数组为奇数个，则将最后一个整数单独划分出来。将 A[i] （ i = 1 , ... , n/2）和 A[i + n/2] 进行比较，若 A[i] > A[i + n/2] ，则 exchange(A[i] , A[i + n/2])。

3）对于划分的两个数组 B1、B2，将两个数组的前 i 个数合并成一个大小为 2i 的数组，对这个数组使用 select 算法找到第 i 小的数，由于在 select 算法里使用了 partition 算法，故第 i 小的数字前面的数均比它小，又有 i < n/2，故前 n/2 个数也就是数组 B1 内包含了用select 找到的第 i 小的数以及小于 第 i 小的数。

4）重复2）、3）步骤直到 i 大于当前数组大小的一半，此时对当前数组的前 i 个数与相邻数组的前 i 个数合并成一个大小为 2i 的数组，若最开始的数组为奇数个，则还需将多余的那一个数加进来，再对这个数组使用 select 算法找到第 i 小的数。

故算法时间可表达为

$$U_i(n) = \begin{cases} T(n) & i \geq \frac{n}{2} \\ \lfloor \frac{n}{2} \rfloor + U_i(\lceil \frac{n}{2} \rceil) + T(2i) & i \leq \frac{n}{2} \end{cases}$$

$U_i(n) = \lfloor \dfrac{n}{2} \rfloor + U_i(\lceil \dfrac{n}{2} \rceil) + T(2i)$
$= \lfloor \dfrac{n}{2} \rfloor + \lfloor \dfrac{n}{2^2} \rfloor + U_i(\lceil \dfrac{n}{2^2} \rceil) + T(2 \cdot 2i)$
$= \lfloor \dfrac{n}{2} \rfloor + \lfloor \dfrac{n}{2^2} \rfloor + \ldots + \lfloor \dfrac{n}{2^k} \rfloor + U_i(\lceil \dfrac{n}{2^k} \rceil) + T(k \cdot 2i)$
$= n(1 - (\lfloor \dfrac{1}{2} \rfloor)^k) + U_i(\lceil \dfrac{n}{2^k} \rceil) + T(k \cdot 2i)$

当i大于当前数组的大小的一半时，停止递归，当前数组的大小的一半为$\dfrac{n}{2^k}$，故有$\dfrac{n}{2^k} \leq i \leq 2 \cdot \dfrac{n}{2^k}$，即$n \leq 2^k \cdot i \leq 2n$

故当i是小于$n/2$的常数时，有

$U_i(n) \leq n - \dfrac{i}{2} + U_i(i) + T(\log \dfrac{2n}{i})$
$= n - \dfrac{i}{2} + U_i(i) + T(\log 2n - \log i)$
$= n + O(\log n)$