1. 如何在$O(n)$时间内对 0 到n^3-1 区间内的 n 个整数进行排序?

已知 a 位的 b 进制无符号整数的范围是 0 到 b^a-1，因此可以将 0 到n^3-1 区间内的整数理解为 3 位的 n 进制数，使用 COUNT_SORT 对每一位进行排序，一共需要调用 3 次 COUNT_SORT，因为计数排序的算法复杂度是$O(n)$，所以该算法的算法复杂度也是$O(n)$。

```
for i=1 to 3
    do COUNT_SORT(A , i)
```

对每一位进行计数排序的 COUNT_SORT 伪代码则是在计数排序的基础上加上位数的判断。

```
mark = 1 <<(i-1)
for j = 0 to K
    do C[j] = 0
for j = 1 to A.length
    do C[(A[j]&mark)>>(j-1)] ++
for j = 1 to K
    do C[i] = C[i] + C[i-1]
for j = A.length downto 1
    do B[C[(A[j]&mark)>>(j-1)]] = A[j]
       C[(A[j]&mark)>>(j-1)] --
```

2. 在单位圆内给定 n 个点，设计一个平均情况下$\Theta(n)$时间代价的算法，能按照点到原点之间的距离对这 n 个点进行排序。

按半径 r 来进行划分。假设将所有的点划分为 N 个桶，也就是需要将半径分为 N 个区间，每个区间点的分布是均匀的，因为点在面积上的分布是均匀的，所以第一个区间的点占据了圆的面积的 1/N，所以第一个点的半径满足$r_1 = \sqrt{\frac{1}{N}}$，由于第一个区间和第二个区间的总面积需要为圆的 2/N，对应的半径$r_2 = \sqrt{\frac{2}{N}}$，所以第二个区间半径的范围就是$\sqrt{\frac{2}{N}}$-$\sqrt{\frac{1}{N}}$，以此类推，第 i 个区间的半径范围就是$\sqrt{\frac{i}{N}}$-$\sqrt{\frac{i-1}{N}}$

在划分好区间之后，使用桶排序来对这些点进行排序，伪代码表示为:

```
R = calculateRadius(A)
n = R.length
for i = 1 to n
    do insert A[i] into list B by compare R[i] with r_j
for I = 0 to n-1
    do sort list B[i] with insert sort
concatenate lists B[0]……B[n-1] together
```