**1. 如何在时间内对0到-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ba-1，因此可以将0到-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，所以该算法的算法复杂度也是。

|  |
| --- |
| 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，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，所以第二个区间半径的范围就是-，以此类推，第i个区间的半径范围就是-

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 rj  for I = 0 to n-1  do sort list B[i] with insert sort  concatenate lists B[0]……B[n-1] together |