


## Lista 4 - AED 2 -

1- Ordenando por contagem, teremos:

$A = [6, 0, 2, 0, 1, 3, 4, 6, 1, 3, 2]$



2	2	2	2	1	0	2	0	0	0	0
---	---	---	---	---	---	---	---	---	---	---

0 1 2 3 4 5 6 7 8 9 10

0, 0, 1, 1, 2, 2, 3, 3, 4, 6, 6, ou seja,  $A = [0, 0, 1, 1, 2, 2, 3, 3, 4, 6, 6]$

2- Lista = { cou, dog, ra, rug, rou, mob, box, tab, bar, ear, tar, dig, big, tea, nou, fox }, ordenando por radix, temos:

a b ... g ... r ... u ... x, teremos a lista:  
2 2 4 3 3 2

Lista = { ra, tea, mob, tab, dog, rug, dig, big, bar, ear, tar, cou, rou, nou, box, fox }

a ... e ... i ... o ... u, teremos a lista:  
4 2 2 7 1

Lista = { tab, bar, ear, tar, ra, tea, dig, big, mob, dog, cou, rou, nou, box, fox, rug }

... b c d e f ... m n ... r s t, a lista ficará:

3 1 2 1 1 1 1 2 1 3

Lista = { bar, big, box, cow, dig, dog, ear, fox, mob, now, row, rug, sa, tab, tar, tea }.

3-a) Teoricamente, o counting Sort é o algoritmo mais otimizado com relação ao tempo de execução ( $O(n)$ ), entretanto, não é nem de perto o mais otimizado na área do uso de memória. O mais otimizado em memória e um bom algoritmo em tempo de execução é o Insertion Sort (Tempo:  $O(n^2)$  para o pior caso e  $O(n)$  para o melhor).

b) Teoricamente, o counting Sort também é o mais otimizado para o tempo de execução (Para ser decrescente, o vetor deve ser invertido), entretanto, na área de uso de memória, temos que o Selection Sort é o otimizado na relação de memória e tem tempo de execução  $O(n^2)$ .

c) O counting Sort é o mais otimizado para o tempo de execução ao custo de memória, e como o Insertion Sort é tido como o melhor para ordenar crescentemente, memória e bom para o tempo de execução.

d) O melhor algoritmo estável é o counting Sort para o tempo de execução, entretanto para a memória, é novamente o Insertion (Insertion é estável).