Projeto e Análise de Algoritmos

Ordenação em tempo linear

Diane Castonguay
diane@inf.ufg.br

Instituto de Informática
Universidade Federal de Goiása

Melhor tempo para ordenação por comparação

Qualquer algoritmo de ordenação

por *comparação*

exige $\Omega(n \lg n)$ comparações
no pior caso.

```
/*
* Os valores de A vão de 0 a k. B é o vetor ordenado.
*/
COUNTINGSORT(A, B, k)

// inicializa o vetor C
1. para i ← 0 até k faça
2.         C[i] ← 0
3. fim-para

// calcular C[i] a quantidade de elementos iguais a i
4. para j ← 1 até comprimento[A] faça
5.         C[A[j]] ← C[A[j]] + 1
6. fim-para
```

```
// continuação de COUNTINGSORT(A, B, k)

// Contar C[i] a quantidade de elementos ≤ i
7. para i ← 2 até k faça
8.         C[i] ← C[i] + C[i-1]
9. fim-para

// Ordenar os elementos de A no vetor B
10.para j ← comprimento[A] até 1 passo -1 faça
11.        B[C[A[j]]] ← A[j]
12.        C[A[j]] ← C[A[j]] – 1
13.fim-para
```

```
// continuação de COUNTINGSORT(A, B, k)

// A ← B
14.para i ← 1 até comprimento[A] faça
15.         A[i] ← B[i]
16.fim-para
```

```
/*
* Os valores de A pertence ao intervalo [0, 1[.
*/
BUCKETSORT(A)
1. n ← comprimento[A]
2. para i ← 1 até n faça
3.      inserir A[i] na lista B[ ⌊ n A[i] ⌋ ]
4. fim-para
5. para i ← 0 até n – 1 faça
6.      ordenar o vetor B[i] com INSERTION-SORT
7. fim-para
8. concatenar as listas B[0], B[1], B[2], ... , B[n-1]
```