AULA 12"

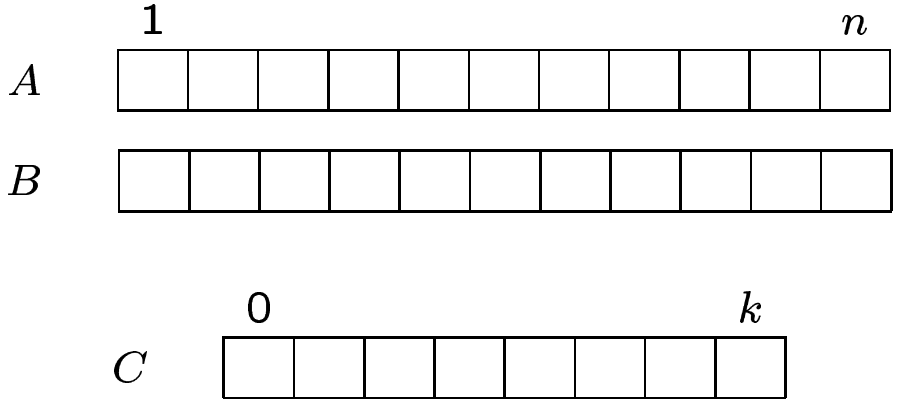
Ordenação em tempo linear

CLRS 8.2–8.3

Ordenação por contagem

Cada $A[i]$ está em $\{0, \ldots, k\}$

```
COUNTING-SORT (A, B, n, k)
1  para i ← 0 até k
2     faça C[i] ← 0
3  para j ← 1 até n
4     faça C[A[j]] ← C[A[j]] + 1
5  para i ← 1 até k
6     faça C[i] ← C[i] + C[i − 1]
7  para j ← n decrescendo até 1
8     faça B[C[A[j]]] ← A[j]
9           C[A[j]] ← C[A[j]] − 1
```



Entre linhas 4 e 5, $C[i]$ = núm de js tq $A[j] = i$
entre linhas 6 e 7, $C[i]$ = núm de js tq $A[j] \leq i$

Consumo de tempo do COUNTING-SORT:

linha	consumo na linha
1–2	$\Theta(k)$
3–4	$\Theta(n)$
5–6	$\Theta(k)$
7–9	$\Theta(n)$

Consumo total: $\Theta(n + k)$

- se $k \leq n$ então consumo é $\Theta(n)$
- se $k \leq 10n$ então consumo é $\Theta(n)$
- se $k = O(n)$ então consumo é $\Theta(n)$
- se $k \geq n^2$ então consumo é $\Theta(k)$

A propósito: COUNTING-SORT é estável

Ordenação digital (= radix sort)

Exemplo:

329	72**0**	7**2**0	**3**29
457	35**5**	3**2**9	**3**55
657	43**6**	4**3**6	**4**36
839	45**7**	8**3**9	**4**57
436	65**7**	3**5**5	**6**57
720	32**9**	4**5**7	**7**20
355	83**9**	6**5**7	**8**39

Cada $A[j]$ têm d dígitos decimais:

$$A[j] = a_d\,10^{d-1} + \cdots + a_2\,10^1 + a_1\,10^0$$

Exemplo com $d = 3$: $\quad 3 \cdot 10^2 + 2 \cdot 10 + 9$

RADIX-SORT (A, n, d)
1 para $i \leftarrow 1$ até d faça
2 $\quad \triangleright$ 1 até d e não o contrário!
3 $\quad$ ordene $A[1\,..\,n]$ pelo dígito i

Linha 3:

- faz ordenação $A[j_1 \,..\, j_n]$ de $A[1 \,..\, n]$ tal que $A[j_1]_i \leq \cdots \leq A[j_n]_i$
- ordenação deve ser estável
- use COUNTING-SORT

Consumo de tempo:

- dígitos decimais: $\Theta(dn)$
- dígitos em $0 \,..\, k$: $\Theta(d\,(n+k))$.

Exemplo com $d = 5$ e $k = 128$:
$a_5 128^4 + a_4 128^3 + a_3 128^2 + a_2 128 + a_1$
sendo $0 \leq a_i \leq 127$

TAREFA 12"

Exercício 12".A
O seguinte algoritmo promete rearranjar o vetor $A[1\,..\,n]$ em ordem crescente supondo que cada $A[i]$ está em $\{0,\ldots,k\}$. O algoritmo está correto?

```
C-SORT (A, n, k)
  para i ← 0 até k
      faça C[i] ← 0
  para j ← 1 até n
      faça C[A[j]] ← C[A[j]] + 1
  j ← 1
  para i ← 0 até k faça
      enquanto C[i] > 0
          faça A[j] ← i
               j ← j + 1
               C[i] ← C[i] − 1
```

Qual o consumo de tempo do algoritmo?

Exercício 12".B
O seguinte algoritmo promete rearranjar o vetor $A[1\,..\,n]$ em ordem crescente supondo que cada $A[j]$ está em $\{1,\ldots,k\}$. O algoritmo está correto? Estime, em notação O, o consumo de tempo do algoritmo.

```
VITO-SORT (A, n, k)
1  i ← 1
2  para a ← 1 até k − 1 faça
3      para j ← i até n faça
4          se A[j] = a
5              então A[j] ↔ A[i]      ▷ troca
6                    i ← i + 1
```

Exercício 12".C
Suponha que os components do vetor $A[1\,..\,n]$ estão todos em $\{0,1\}$. Prove que $n-1$ comparações são suficientes para rearranjar o vetor em ordem crescente.

Exercício 12".D
Qual a principal invariante do algoritmo RADIX-SORT?