Universidade Federal da Paraíba
Centro de Informática
Departamento de Computação Científica

Ordenações Lineares

Prof. Gilberto Farias de Sousa

Roteiro

- Ordenação por Contagem
- Radix Sort
- Bucket Sort

Ordenação por Contagem

A Ordenação por Contagem pressupõe que cada um dos **n** elementos de entrada é um inteiro no intervalo de 1 a K, para algum inteiro K.

Idéia Básica:

- Determinar para cada elemento de entrada **x**, o número de elementos menores que x.
- Com esta informação podemos inserir o elemento diretamente em sua posição no arranjo de saída.

  Exemplo: Se houver 17 elementos menores que x, logo x deve ser colocado na posição 18.

Ordenação por Contagem

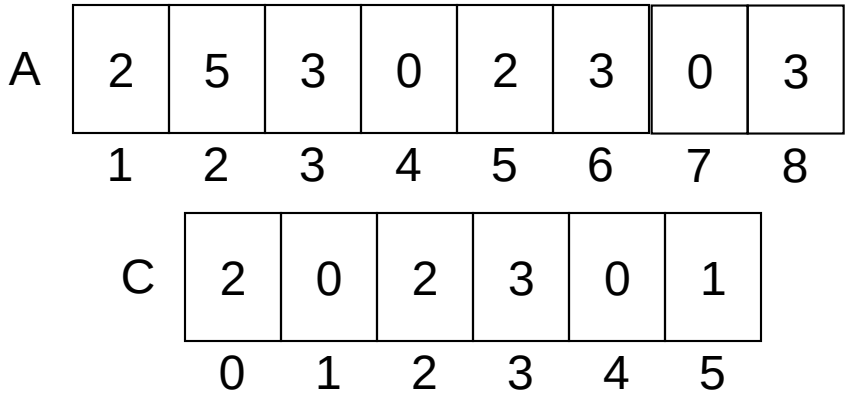
- Os dados manipulados pelo algoritmo de ordenação de Contagem são:
  - A[1..n] : arranjo desordenado de entrada
  - B[1..n] : arranjo ordenado de saída
  - C[0..k] – arranjo auxiliar
  - K – é o valor máximo contido em A.

Algoritmo – Ordenação por Contagem

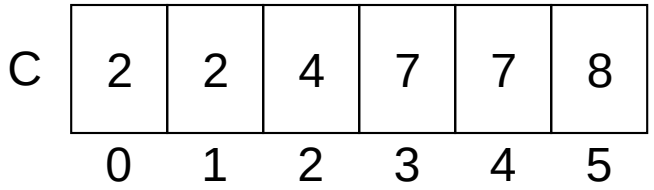
COUNTING-SORT (A, B, k)

1. *for i ← 0 to k*
2. *C[i] ← 0;*
3. *for j ← 1 to comprimento[A]*
4. *C[A[j]] ← C[A[j]] + 1;*
5. *//Agora C[i] contém o número de elementos iguais a i em A*
6. *for i ← 2 to k*
7. *C[i] ← C[i] + C[i -1];*
8. *//Agora C[i] contém o número de elementos menores ou iguais a i.*
9. *for j ← comprimento[A] downto 1*
10. *B[C[A[j]]] ← A[j];*
11. *C[A[j]] ← C[A[j]] – 1;*

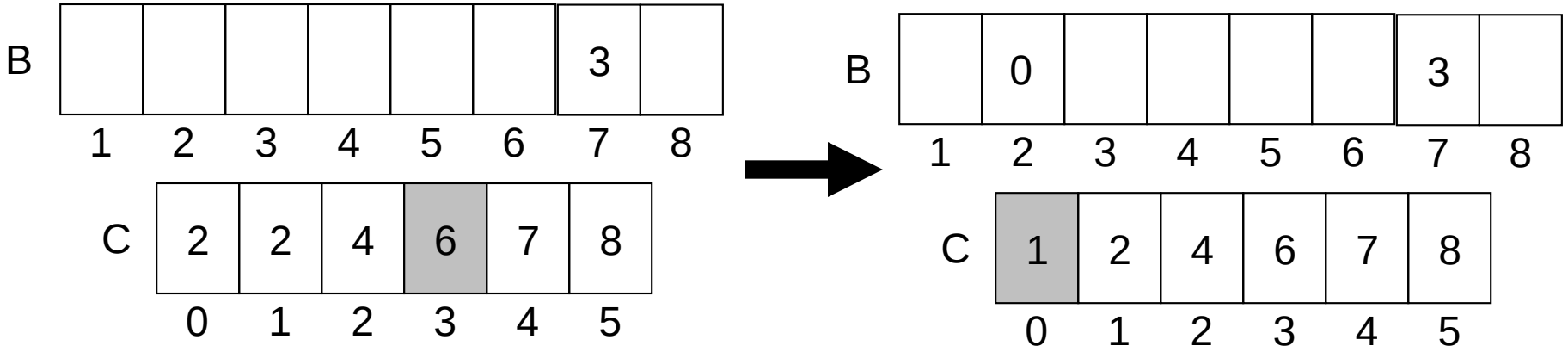
Algoritmo – Ordenação por Contagem



(a) Execução das linhas 3 e 4 do algoritmo



(b) Execução das linhas 6 e 7 do algoritmo



(c) Execução das linhas 9, 10 e 11 do algoritmo

Tempo de Execução – Ordenação por Contagem

O loop das linhas 1 e 2 demora o tempo $\Theta(k)$, o loop das linhas 3 e 4 demora o tempo $\Theta(n)$, o loop das linhas 6 e 7 de mora o tempo $\Theta(k)$ e o loop das linhas 9 a 11 demora o tempo $\Theta(n)$.

Portanto o tempo total é $O(n + k)$. Na prática usamos $k = O(n)$, passando o tempo total para $O(n)$.

Exercício

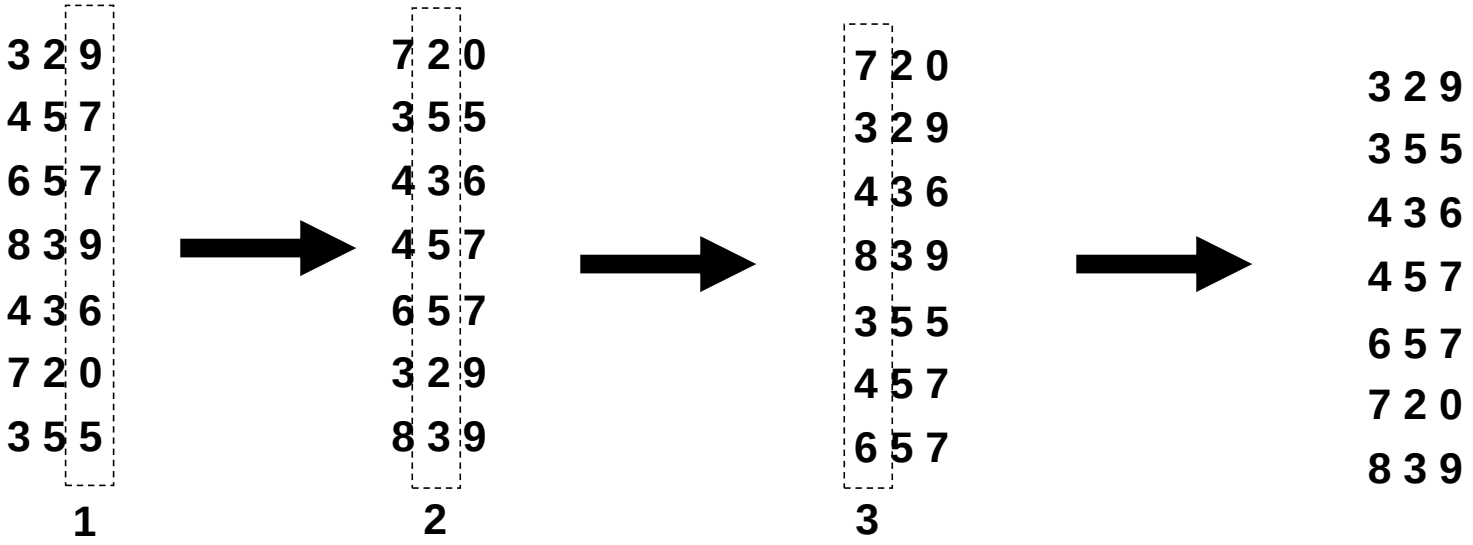
- Ilustre a operação de COUTING-SORT sobre o arranjo

  A = <6, 0, 2, 0, 1, 3, 4, 6, 1, 3, 2>

Radix Sort

- Trabalha com números de **d** dígitos.
- Ordena os elementos coluna por coluna (dígitos)
- De forma não intuitiva, a ordenação começa pelo número menos significativo
- O processo continua até os elementos terem sido ordenados sobre todos os dígitos **d**.

Algoritmo Radix Sort



Algoritmo Radix Sort

RADIX-SORT(A, d)

for i ← 1 to d

do usar uma ordenação para o arranjo A sobre o dígito i.

*Este procedimento supõe que cada elemento no arranjo de **n** elementos tem **d** dígitos, onde o dígito 1 é o dígito menos significativo e o dígito **d** é o dígito mais significativo*

Algoritmo Radix Sort

- Na prática o algoritmo radix-sort é usado para ordenar registros de informações chaveados por vários campos distintos.

Exercício: como ordenar as datas abaixo usando o algoritmo radix-sort?

25/07/2010
29/02/2001
07/01/2011
30/12/1988

Bucket Sort

- A entrada é uniformemente distribuída sobre o intervalo [0,1).
- A idéia é dividir o intervalo [0,1) em subintervalos de mesmo tamanho, chamados **baldes.**
- Considerando que há uma uniformidade na distribuição, os baldes terão uma quantidade de elementos próximos.
- Ordenando cada balde e depois percorrendo os baldes em ordem, listando os elementos contidos em cada um.

Algoritmo Bucket Sort

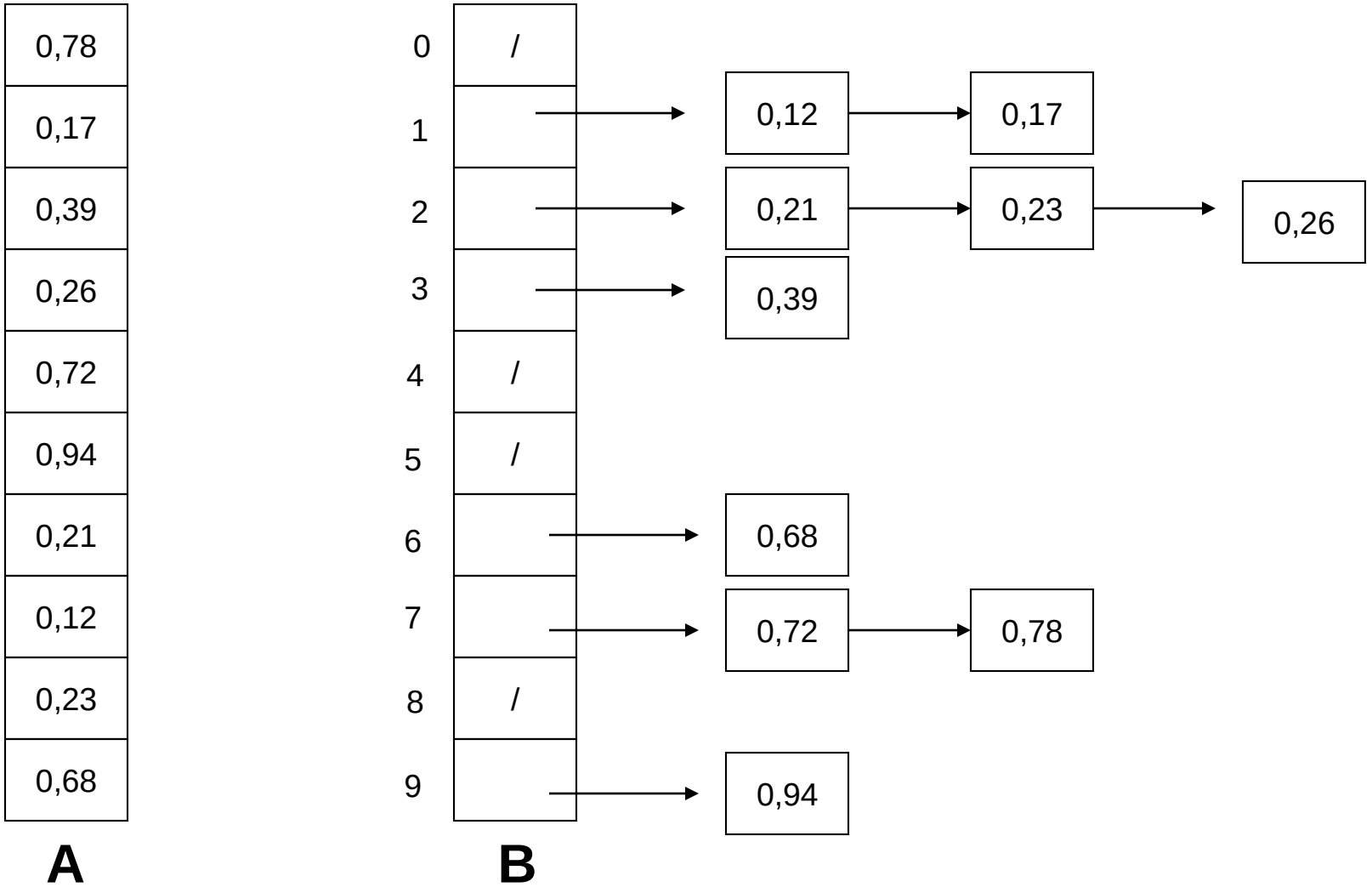
- O algoritmo tem como entrada o arranjo A[1..n], onde $0 \leq A[i] < 1$
- E trabalha com um arranjo auxiliar B[0 .. n-1] de listas ligadas (baldes)

Bucket – Sort (A)

1. n ← comprimento[A]
2. for i ← 1 to n
3. inserir A[i] na lista B[A[i] *10]
4. for i ← 0 to n-1
5. Ordenar lista B[i] com ordenação por inserção

6.Concatenar as listas pela ordem encontrada

Algoritmo Bucket Sort



Algoritmo Bucket Sort

- Havendo uma distribuição uniforme nos valores dos dados de entrada, o algoritmo executa em um tempo linear O(n).