

Ordenação

Prof. Marcel Hugo
Estruturas de Dados

Departamento de Sistemas e Computação
Universidade Regional de Blumenau – FURB

Slides criados a partir do Prof. Paulo Rodacki Gomes,
Disciplina de Algoritmo e Estrutura de Dados, DSC - FURB



1

1

Introdução

**Vetor com
elementos não
ordenados**



**Algoritmo de
ordenação**



**Vetor com
elementos
ordenados**

Entrada: vetor
com elementos
a serem
ordenados

Saída: vetor
com os
elementos na
ordem
especificada

3

Introdução

- ▣ Ordenação: pode ser aplicada em qualquer conjunto de dados com ordem definida
- ▣ vetores de objetos:
 - ▣ chave de ordenação escolhida entre os atributos (comparável)
 - ▣ elemento do vetor contém referência para objeto
 - ▣ troca de ordem entre dois elementos == troca de referência para objeto

4

BubbleSort

- ▣ Quando dois elementos estão fora de ordem, troque-os de posição até que o i-ésimo elemento de maior valor do vetor seja levado para as posições finais do vetor
- ▣ continue o processo até que todo o vetor esteja ordenado
- ▣ Exemplo:

	Maior elemento					2º maior elemento					3º maior elemento			
v0	4	2	5	1	v4	2	4	1	5	v6	2	1	4	5
v1	2	4	5	1	v5	2	4	1	5		1	2	4	5
v2	2	4	5	1		2	1	4	5					
v3	2	4	1	5										
	0	1	2	3		0	1	2	3		0	1	2	3

5

Exemplo

25	48	37	12	57	86	33	92	25x48
25	48	37	12	57	86	33	92	48x37 troca
25	37	48	12	57	86	33	92	48x12 troca
25	37	12	48	57	86	33	92	48x57
25	37	12	48	57	86	33	92	57x86
25	37	12	48	57	86	33	92	86x33 troca
25	37	12	48	57	33	86	92	86x92
25	37	12	48	57	33	86	<u>92</u>	final da primeira passada

o maior elemento, 92, já está na sua posição final

6

Exemplo

25	37	12	48	57	33	86	<u>92</u>	25x37
25	37	12	48	57	33	86	<u>92</u>	37x12 troca
25	12	37	48	57	33	86	<u>92</u>	37x48
25	12	37	48	57	33	86	<u>92</u>	48x57
25	12	37	48	57	33	86	<u>92</u>	57x33 troca
25	12	37	48	33	57	86	<u>92</u>	57x86
25	12	37	48	33	57	<u>86</u>	<u>92</u>	final da segunda passada

o segundo maior elemento, 86, já está na sua posição final

7

Exemplo

25 12 37 48 33 57 86 92	25x12 troca
12 25 37 48 33 57 86 92	25x37
12 25 37 48 33 57 86 92	37x48
12 25 37 48 33 57 86 92	48x33 troca
12 25 37 33 48 57 86 92	48x57
12 25 37 33 48 57 86 92	final da terceira passada

Idem para 57.

12 25 37 33 48 57 86 92	12x25
12 25 37 33 48 57 86 92	25x37
12 25 37 33 48 57 86 92	37x33 troca
12 25 33 37 48 57 86 92	37x48
12 25 33 37 48 57 86 92	final da quarta passada

Idem para 48.

8

12 25 33 37 48 57 86 92	12x25
12 25 33 37 48 57 86 92	25x33
12 25 33 37 48 57 86 92	33x37
12 25 33 37 48 57 86 92	final da quinta passada

Idem para 37.

12 25 33 37 48 57 86 92	12x25
12 25 33 37 48 57 86 92	25x33
12 25 33 37 48 57 86 92	final da sexta passada

Idem para 33.

12 25 33 37 48 57 86 92	12x25
12 25 33 37 48 57 86 92	final da sétima passada

Idem para 25 e, conseqüentemente, 12.

12 25 33 37 48 57 86 92	final da ordenação
-------------------------	--------------------

9

BubbleSort

▣ Algoritmo iterativo (versão 1)

```
Algoritmo: bubbleSort(int v[])  
  
int i, j;  
int n ← v.length;  
para i ← n - 1 até 1 faça  
    para j ← 0 até i faça  
        se v[j] > v[j + 1] então  
            int temp ← v[j];  
            v[j] ← v[j + 1];  
            v[j + 1] ← temp;
```

Algoritmo 5.1: BubbleSort (versão 1)

10

BubbleSort

▣ Algoritmo iterativo (versão 2)

```
Algoritmo: bubbleSort(int v[])  
  
int i, j;  
int n ← v.length;  
para i ← n - 1 até 1 faça  
    boolean troca ← false;  
    para j ← 0 até i faça  
        se v[j] > v[j + 1] então  
            int temp ← v[j];  
            v[j] ← v[j + 1];  
            v[j + 1] ← temp;  
            troca ← true;  
    se troca ≠ true então retorna
```

Algoritmo 5.2: BubbleSort (versão 2)

11

BubbleSort

- Implementação recursiva:

Algoritmo: bubbleSortRecursivo(int n, int v[])

```
int j;  
boolean troca ← false;  
para j ← 0 até n-1 faça  
  se v[j] > v[j+1] então  
    int temp ← v[j];  
    v[j] ← v[j+1];  
    v[j+1] ← temp;  
    troca ← true;  
se troca então  
  bubbleSortRecursivo(n-1, v);
```

maior elemento
(n=4; i=n-1=3)

4	2	5	1
2	4	5	1
2	4	5	1
2	4	1	5

2º maior elemento
(i=n-2=2)

2	4	1	5
2	4	1	5
2	1	4	5

3º maior elemento
(i=n-3=1)

2	1	4	5
1	2	4	5

0 1 2 3

Algoritmo 5.3: BubbleSort recursivo

13

BubbleSort

- Algoritmo genérico:
- independente do tipo/classe dos dados armazenados no vetor
- usa função auxiliar para comparar elementos

14