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

Introdução

Vetor com elementos não ordenados

Entrada: vetor

com elementos

a serem

ordenados

Algoritmo de ordenação

>

Vetor com elementos ordenados

Saída: vetor com os elementos na ordem especificada

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

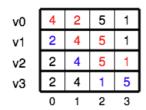
4

BubbleSort

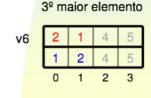
- Quando dois elementos estão fora de ordem, troque-os de posição até que o i-ésimo elemento de maior valor do vetor seja levado para as posições finais do vetor
- continue o processo até que todo o vetor esteja ordenado

2º major elemento

Exemplo: Maior elemento



v4 v5 2 4 1 5 2 1 4 5 0 1 2 3



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
                            57x86
25 37 12 48 57 86 33 92
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 92
                           final da pri<mark>meira passada</mark>
```

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

6

Exemplo

```
25 37 12 48 57 33 86 92 25x37
25 37 12 48 57 33 86 92 37x12 troca
25 12 37 48 57 33 86 92 37x48
25 12 37 48 57 33 86 92 48x57
25 12 37 48 57 33 86 92 57x33 troca
25 12 37 48 33 57 86 92 57x86
25 12 37 48 33 57 86 92 final da segunda passada
```

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

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
ldem para 57.
12 25 37 33 48 57 86 92
                           12x25
12 25 37 33 48 57 86 92
                          25x37
                          37x33 troca
12 25 37 33 48 57 86 92
12 25 33 37 48 57 86 92
                          37x48
12 25 33 37 <u>48 57 86 92</u>
                          final da quarta passada
ldem 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 sé<mark>tima passada</mark>
Idem para 25 e, consequentemente, 12.
12 25 33 37 48 57 86 92 final da ordenação
```

BubbleSort

Algoritmo iterativo (versão 1)

```
Algoritmo: bubbleSort(int v[])

int \ i, \ j;
int \ n \leftarrow v.length;
para i \leftarrow n-1 até l faça

\begin{array}{c|c} \text{para } j \leftarrow 0 \text{ até } i \text{ faça} \\ \text{se } v[j] > v[j+1] \text{ então} \\ & int \ temp \leftarrow v[j]; \\ v[j] \leftarrow v[j+1]; \\ v[j+1] \leftarrow temp; \end{array}
```

Algoritmo 5.1: BubbleSort (versão 1)

10

BubbleSort

Algoritmo iterativo (versão 2)

```
Algoritmo: bubbleSort(int v[])

int i, j;

int n \leftarrow v.length;

para i \leftarrow n-1 até 1 faça

| boolean troca \leftarrow false;

para j \leftarrow 0 até i faça

| se v[j] > v[j+1] então
| int temp \leftarrow v[j];
| v[j] \leftarrow v[j+1];
| v[j+1] \leftarrow temp;
| troca \leftarrow true;
| se troca \neq true então retorna

Algoritmo 5.2: BubbleSort (versão 2)
```

BubbleSort

Implementação recursiva:

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

int j;

boolean troca \leftarrow false;

para j \leftarrow 0 até n\text{-}1 faça

\begin{array}{c} \text{se } v[j] > v[j+1] \text{ então} \\ \text{int } temp \leftarrow v[j]; \\ v[j+1] \leftarrow temp; \\ troca \leftarrow true; \end{array}

se troca então

\begin{array}{c} \text{bubbleSortRecursivo}(n\text{-}1, v); \end{array}

Algoritmo 5.3: BubbleSort recursivo
```

13

BubbleSort

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