INSTITUTO METRÓPOLE DIGITAL

UFRN UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Estrutura de Dados Básicas I

Aula 9 – Algoritmos de ordenação III

Prof. Eiji Adachi M. Barbosa

Objetivos

- Apresentar e implementar o algoritmo Bubble Sort, também chamado de ordenação por comparação sucessivas

Referências para esta aula

- Notas de aula “Análise de algoritmos”, prof. Paulo Feofiloff, IME-USP:
  - http://www.ime.usp.br/~pf/algoritmos/aulas/mrgsrt.html

ORDENAÇÃO POR COMPARAÇÕES SUCESSIVAS (*BUBBLE SORT*)

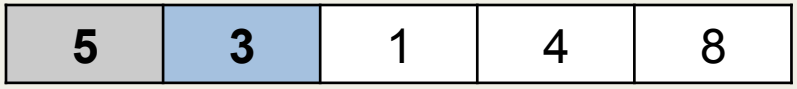
Ideia geral

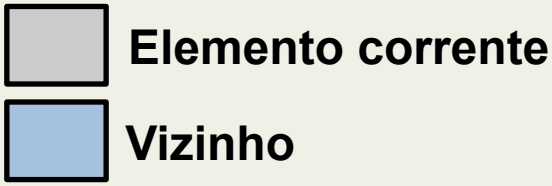
"Percorra o vetor n vezes, a cada vez 'flutuando' o i-ésimo maior elemento para a (n-i+1)-ésima posição do vetor"

Ideia geral

- Compare elemento corrente com seu vizinho:
- Se estão fora de ordem, troquem de posição

1ª Iteração

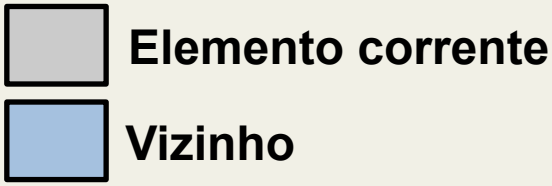




Ideia geral

- Compare elemento corrente com seu vizinho:
- Se estão fora de ordem, troquem de posição

1ª Iteração



Ideia geral

- Compare elemento corrente com seu vizinho:
- Se estão fora de ordem, troquem de posição

1ª Iteração

5	**3**	1	4	8
3	**5**	**1**	4	8

→

3	**5**	1	4	8

Elemento corrente

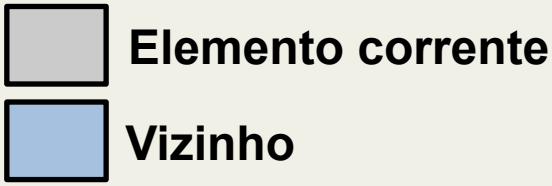
Vizinho

Ideia geral

- Compare elemento corrente com seu vizinho:
- Se estão fora de ordem, troquem de posição

1ª Iteração

5	**3**	1	4	8	→	**3**	**5**	1	4	8
3	**5**	**1**	4	8	→	3	**1**	**5**	4	8



Ideia geral

- Compare elemento corrente com seu vizinho:
- Se estão fora de ordem, troquem de posição

1ª Iteração

5	**3**	1	4	8
3	**5**	**1**	4	8
3	1	**5**	**4**	8

→

3	**5**	1	4	8
3	**1**	**5**	4	8
3	1	**4**	**5**	8

Elemento corrente

Vizinho

Ideia geral

- Compare elemento corrente com seu vizinho:
- Se estão fora de ordem, troquem de posição

1ª Iteração

5	**3**	1	4	8	→	**3**	**5**	1	4	8
3	**5**	**1**	4	8	→	3	**1**	**5**	4	8
3	1	**5**	**4**	8	→	3	1	**4**	**5**	8

Elemento corrente

Vizinho

Ideia geral

- Compare elemento corrente com seu vizinho:
- Se estão fora de ordem, troquem de posição

1ª Iteração

5	**3**	1	4	8	→	**3**	**5**	1	4	8
3	**5**	**1**	4	8	→	3	**1**	**5**	4	8
3	1	**5**	**4**	8	→	3	1	**4**	**5**	8
3	1	4	**5**	**8**	→	3	1	4	**5**	**8**

Elemento corrente

Vizinho

Ideia geral

- Compare elemento corrente com seu vizinho:
- Se estão fora de ordem, troquem de posição

1ª Iteração

5	**3**	1	4	8	→	**3**	**5**	1	4	8
3	**5**	**1**	4	8	→	3	**1**	**5**	4	8
3	1	**5**	**4**	8	→	3	1	**4**	**5**	8
3	1	4	**5**	**8**	→	3	1	4	**5**	**8**

Elemento corrente

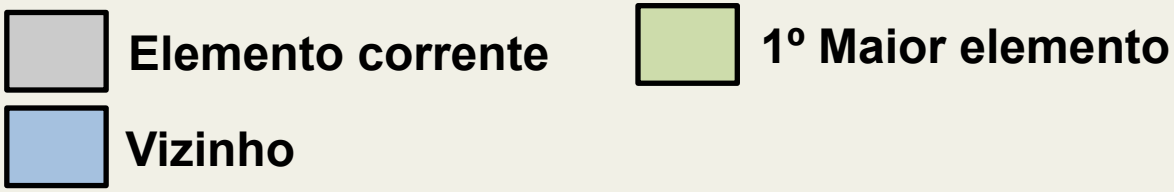
Vizinho

1º Maior elemento

Ideia geral

- Compare elemento corrente com seu vizinho:
- Se estão fora de ordem, troquem de posição

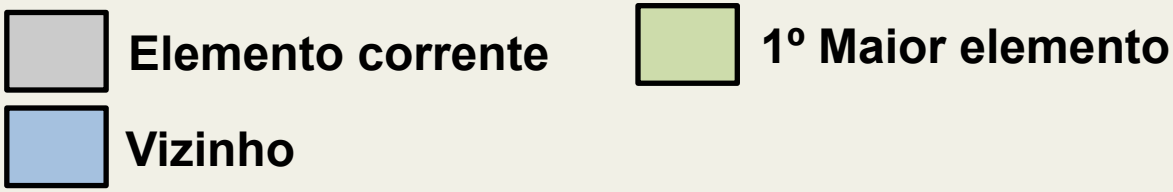
2ª Iteração



Ideia geral

- Compare elemento corrente com seu vizinho:
- Se estão fora de ordem, troquem de posição

2ª Iteração



Ideia geral

- Compare elemento corrente com seu vizinho:
- Se estão fora de ordem, troquem de posição

2ª Iteração

3	**1**	4	5	8
1	**3**	**4**	5	8

→

1	**3**	4	5	8

Elemento corrente

Vizinho

1º Maior elemento

Ideia geral

- Compare elemento corrente com seu vizinho:
- Se estão fora de ordem, troquem de posição

2ª Iteração

3	**1**	4	5	8	→	**1**	**3**	4	5	8
1	**3**	**4**	5	8	→	1	**3**	**4**	5	8

Elemento corrente

Vizinho

1º Maior elemento

Ideia geral

- Compare elemento corrente com seu vizinho:
- Se estão fora de ordem, troquem de posição

2ª Iteração

3	**1**	4	5	8
1	**3**	**4**	5	8
1	3	**4**	**5**	8

→

1	**3**	4	5	8
1	**3**	**4**	5	8

Elemento corrente

Vizinho

1º Maior elemento

Ideia geral

- Compare elemento corrente com seu vizinho:
- Se estão fora de ordem, troquem de posição

2ª Iteração

3	**1**	4	5	8	→	**1**	**3**	4	5	8
1	**3**	**4**	5	8	→	1	**3**	**4**	5	8
1	3	**4**	**5**	8	→	1	3	**4**	**5**	8

Elemento corrente

Vizinho

1º Maior elemento

Ideia geral

- Compare elemento corrente com seu vizinho:
- Se estão fora de ordem, troquem de posição

2ª Iteração

3	**1**	4	5	8	→	**1**	**3**	4	5	8
1	**3**	**4**	5	8	→	1	**3**	**4**	5	8
1	3	**4**	**5**	8	→	1	3	**4**	**5**	8

Elemento corrente

Vizinho

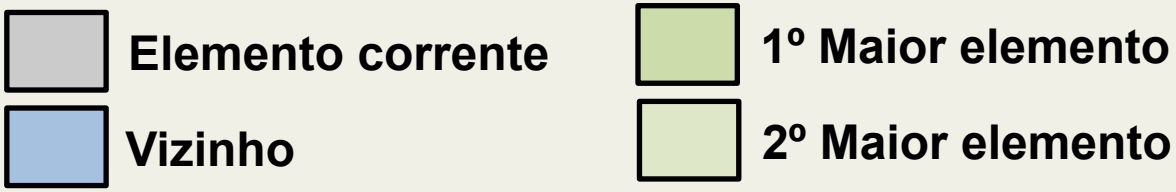
1º Maior elemento

2º Maior elemento

Ideia geral

- Compare elemento corrente com seu vizinho:
- Se estão fora de ordem, troquem de posição

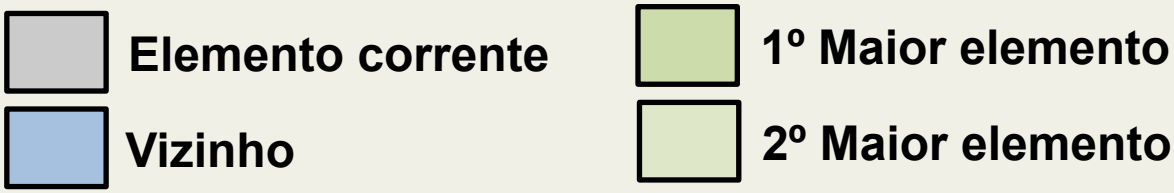
3ª Iteração



Ideia geral

- Compare elemento corrente com seu vizinho:
- Se estão fora de ordem, troquem de posição

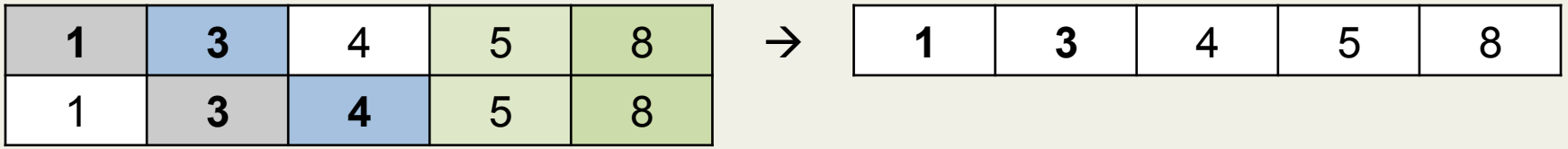
3ª Iteração



Ideia geral

- Compare elemento corrente com seu vizinho:
- Se estão fora de ordem, troquem de posição

3ª Iteração



Ideia geral

- Compare elemento corrente com seu vizinho:
- Se estão fora de ordem, troquem de posição

3ª Iteração

1	**3**	4	5	8
1	**3**	**4**	5	8

→

1	**3**	4	5	8
1	**3**	**4**	5	8

Elemento corrente

Vizinho

1º Maior elemento

2º Maior elemento

Ideia geral

- Compare elemento corrente com seu vizinho:
- Se estão fora de ordem, troquem de posição

3ª Iteração

1	**3**	4	5	8
1	**3**	**4**	5	8

→

1	**3**	4	5	8
1	**3**	**4**	5	8

Elemento corrente

Vizinho

1º Maior elemento

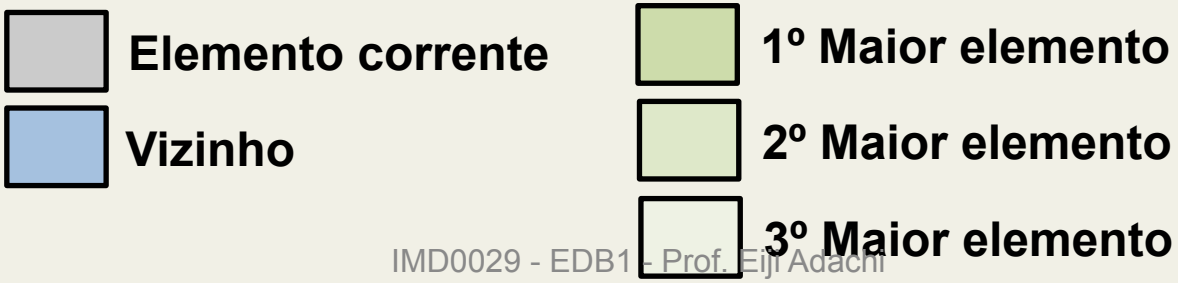
2º Maior elemento

3º Maior elemento

Ideia geral

- Compare elemento corrente com seu vizinho:
- Se estão fora de ordem, troquem de posição

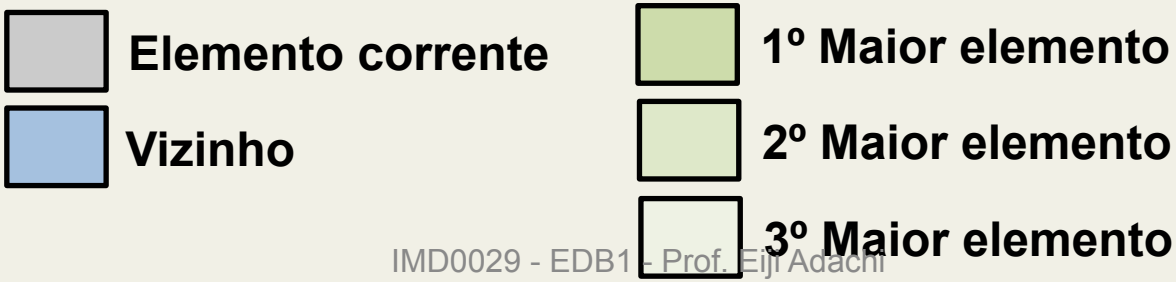
4ª Iteração



Ideia geral

- Compare elemento corrente com seu vizinho:
- Se estão fora de ordem, troquem de posição

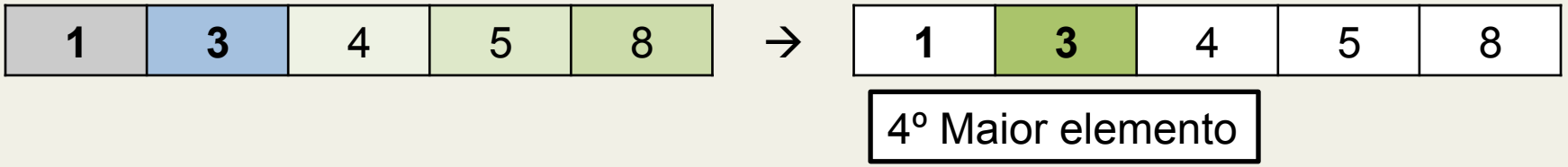
4ª Iteração

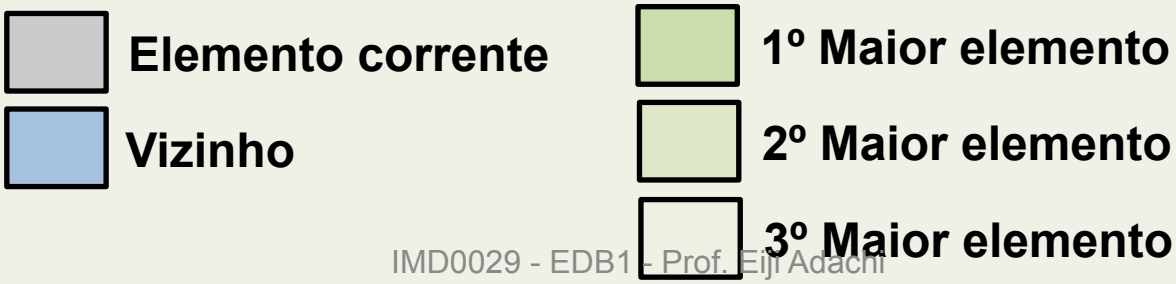


Ideia geral

- Compare elemento corrente com seu vizinho:
- Se estão fora de ordem, troquem de posição

4ª Iteração

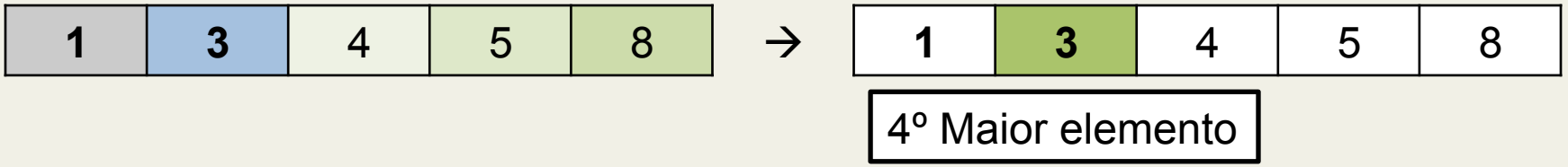


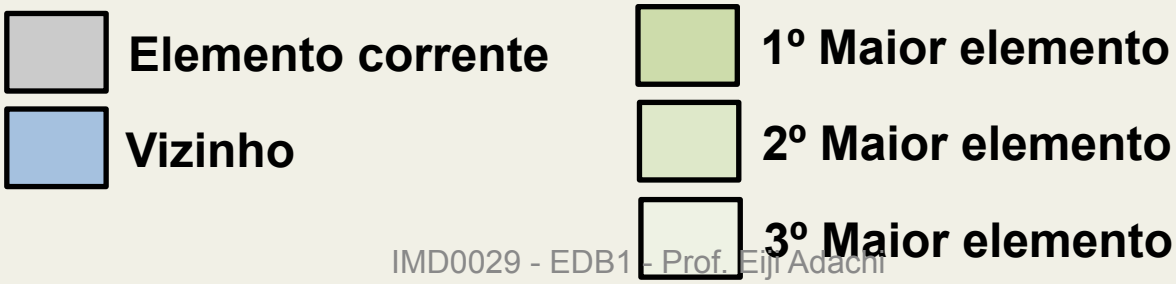


Ideia geral

- Na i-ésima iteração, i-ésimo maior elemento “flutua” para o “alto” do vetor (ou para a n-i+1 ésima posição)

4ª Iteração





Bubble sort

```
Sort(v[n]):
  FOR i = 0; i < n i++:
     FOR j = 0; j <n-i-1; j++:
          IF v[j] > v[j+1] :
               Swap v[j], v[j+1]
          END_IF
     END_FOR
  END_FOR
END
```

Qual a complexidade (pior e melhor caso)?

Bubble sort

```
Sort(v[n]):
   mudou = TRUE
   fim = n
   WHILE mudou :
        mudou = FALSE
        FOR i = 0; i < fim; i++:
               IF v[i] > v[i+1] ENTÃO:
                      Swap v[i], v[i+1]
                      mudou = TRUE
               END_IF
        END_FOR
        fim = fim-1
   END_WHILE
END
```

Qual a complexidade (pior e melhor caso)?

Ordenação Bubble sort

- Algoritmo simples
  - Pior e melhor caso: $\Theta(n^2)$
    - Qualquer vetor

- Algoritmo melhorado
  - Pior caso: $\Theta(n^2)$
    - Vetor ordenado em ordem inversa
  - Melhor caso: $\Theta(n)$
    - Vetor já ordenado

INSTITUTO METRÓPOLE DIGITAL

UFRN UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Estrutura de Dados Básicas I

Aula 9 – Algoritmos de ordenação III

Prof. Eiji Adachi M. Barbosa