Estrutura de Dados Básicas I.

Aula 7 – Algoritmos de ordenação I

Prof. Eiji Adachi M. Barbosa

Objetivos

- Introduzir o problema de ordenação
- Apresentar algoritmo de ordenação por seleção
- Implementar algoritmo de ordenação por seleção

Referência extra para esta aula

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

PROBLEMA DA ORDENAÇÃO

O que é **ordenação**?

O que é necessário para ordenar?

Estabelecer relação de ordem

Relação de ordem

- Ordem não-estrita (ou ampla):
  - Reflexividade: $\forall x \in A : R(x,x)$
  - Antissimetria: $\forall x, y \in A : R(x,y) \wedge R(y,x) \rightarrow x = y$
  - Transitividade: $\forall x, y, z \in A : R(x,y) \wedge R(y,z) \rightarrow R(x,z)$

- Ordem estrita:
  - Irreflexividade: $\forall x \in A : \neg R(x,x)$
  - Assimetria: $\forall x, y \in A : R(x,y) \rightarrow \neg R(y,x)$
  - Transitividade: $\forall x, y, z \in A : R(x,y) \wedge R(y,z) \rightarrow R(x,z)$

Relação de ordem

- Ordem total
  - Todos elementos são comparáveis

- Totalidade para ordens não-estritas:

$$\forall x, y \in A : R(x,y) \vee R(y,x)$$

- Totalidade para ordens estritas:

$$\forall x, y \in A : R(x,y) \vee R(y,x) \vee x = y$$

Relação de ordem

- Exemplo de relação de ordem total não-estrita:
  – "É menor ou igual que"
  – "É maior ou igual que"

- Exemplo de relação de ordem total estrita:
  – "É Menor"
  – "É Maior"

Conjuntos ordenáveis

- Se um conjunto A admite uma ordem total, então A é totalmente ordenável

- Todo conjunto é ordenável, desde que seja estabelecida uma relação de ordem (estrita ou não-estrita)

Problema de ordenação

- Entrada:
  - A: Coleção de elementos E_1, E_2, $E_{3,}$ ... E_n
  - R: Relação de ordem sobre os elementos de A

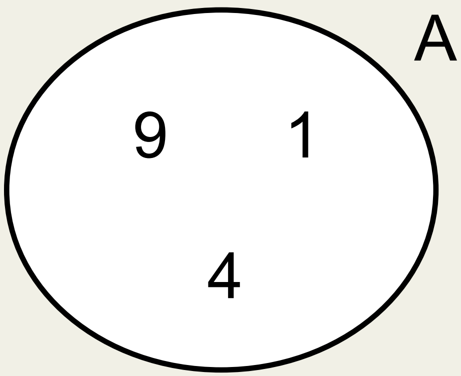
- Saída:
  - Permutação A' tal que elementos subsequentes de A' obedeçam a relação de ordem R
    - $A' = [\ E'_1, E'_2, \dots E'_n\ ]$ tal que $R(E'_1, E'_2)$ e $R(E'_2, E'_3)$ e ... $R(E'_{n-1}, E'_n)$

Permutação

- Uma permutação de um conjunto finito A é uma disposição dos elementos de A em uma sequência

Ex.: Dado o conjunto A ao lado, são permutações de A:

$P_1 = \{1, 4, 9\}$
$P_2 = \{1, 9, 4\}$
$P_3 = \{4, 1, 9\}$
$P_4 = \{4, 9, 1\}$
$P_5 = \{9, 1, 4\}$
$P_6 = \{9, 4, 1\}$



Problema de ordenação

- Ex. Ordenação de inteiros:
  - Entrada:
    - A: Coleção de inteiros [2, 5, 1, 9, 10, 7]
    - R: Relação “menor que“ (<)
  - Saída:
    - A’ = [1, 2, 5, 7, 9, 10]

Problema de ordenação

- Ex. Ordenação de inteiros:
  - Entrada:
    - A: Coleção de inteiros [2, 5, 1, 9, 10, 7]
    - R: Relação “maior que“ (>)
  - Saída:
    - A’ = [10, 9, 7, 5, 2, 1]

Problema de ordenação

- Ex. Ordenação de strings:
  - Entrada:
    - A: Coleção de inteiros [“beta”, “alfa”, “delta”, “charlie”]
    - R: Relação “ordem alfabética” (ou lexicográfica)
  - Saída:
    - A’ = [“alfa”, “beta”, “charlie”, “delta”]

Qual seria uma estratégia de ordenação muito ruim?

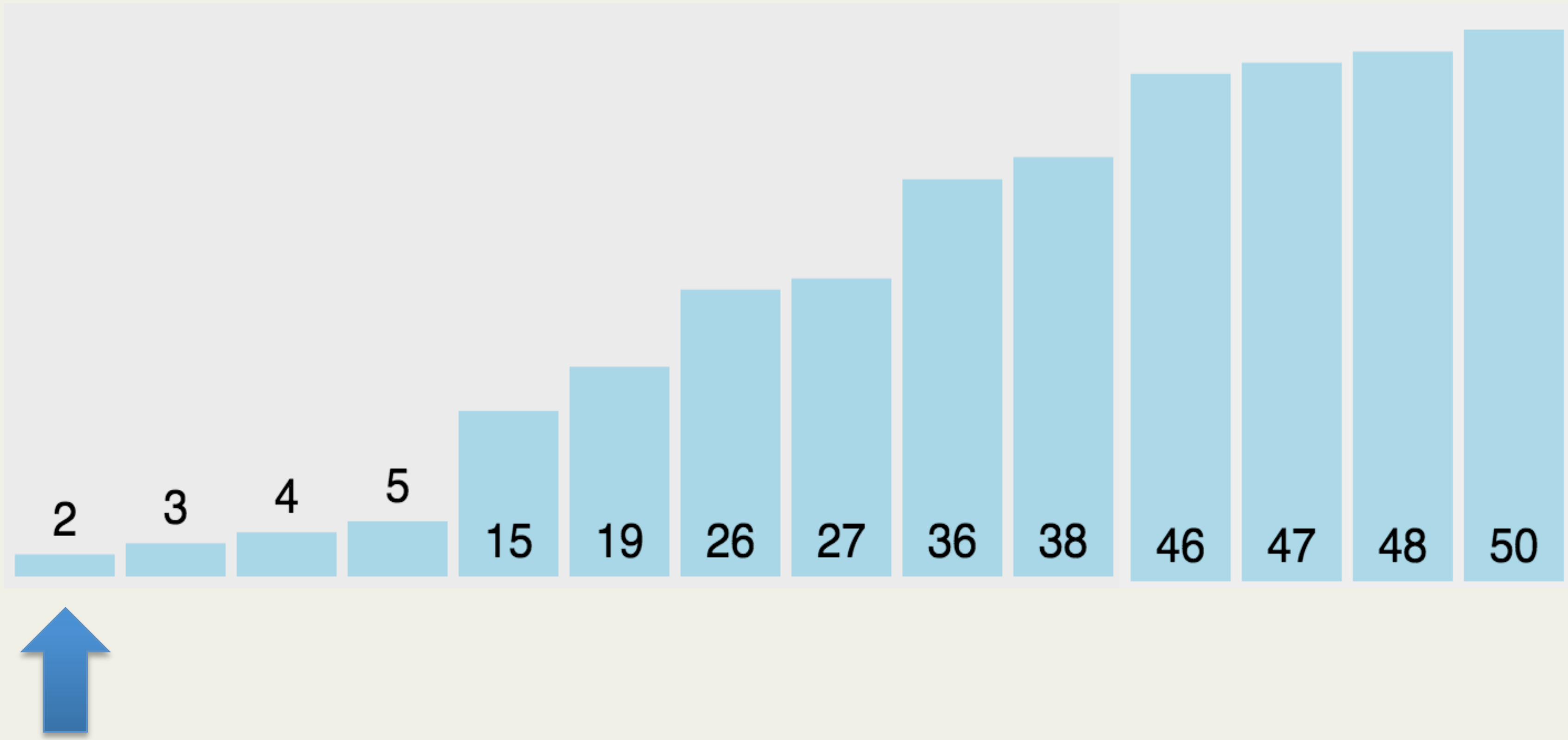
Solução "ruim"

```
Ordene( V[], N ) :
   Gere todas as possíveis permutações de V
   PARA CADA permutação P, FAÇA:
      Verifique se P está ordenada
      SE P está ordenada, ENTÃO:
         V = P
         RETORNE
      FIM_SE
   FIM_PARA
FIM
```

Qual a complexidade desta solução?
R: N!

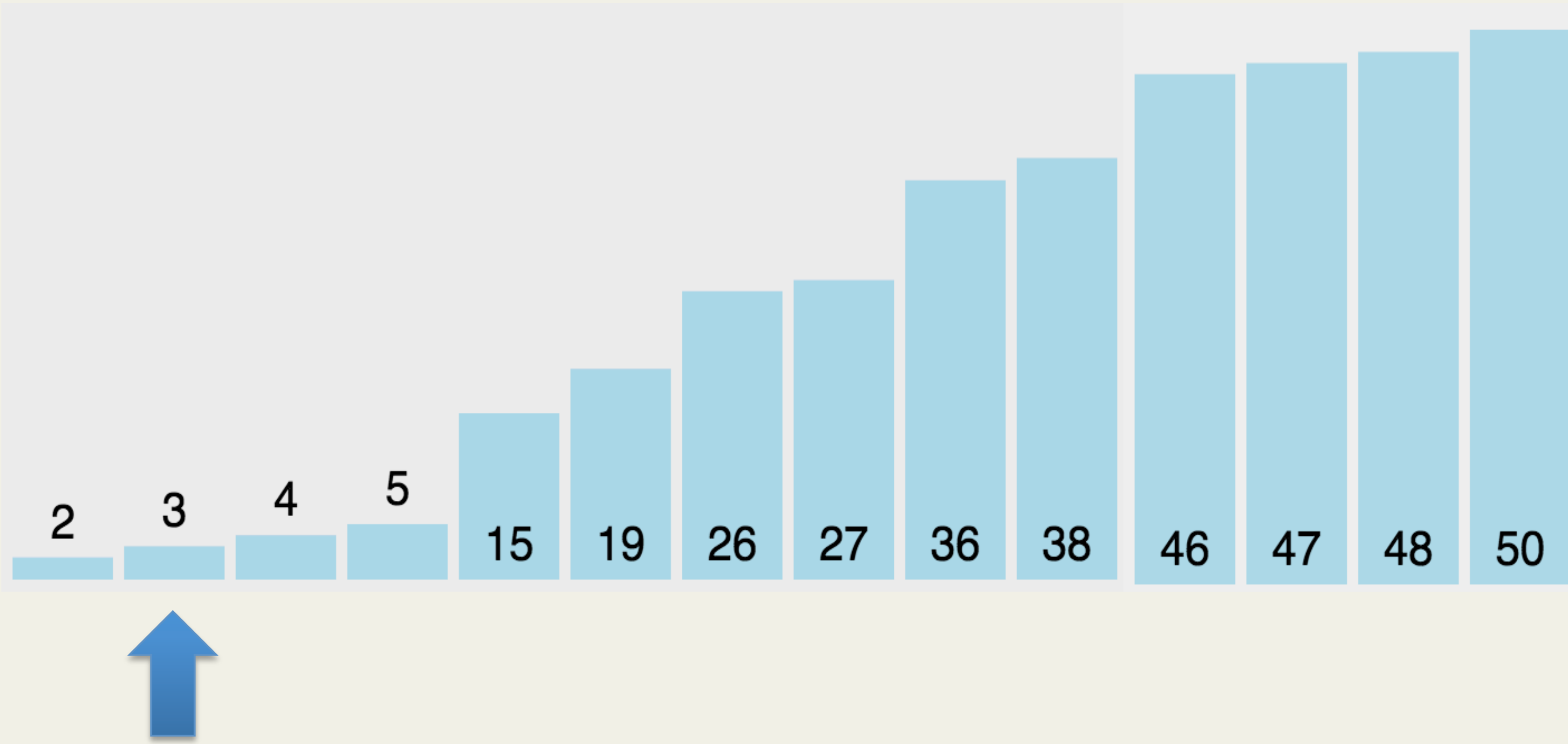
ORDENAÇÃO POR **SELEÇÃO**

Ideia geral



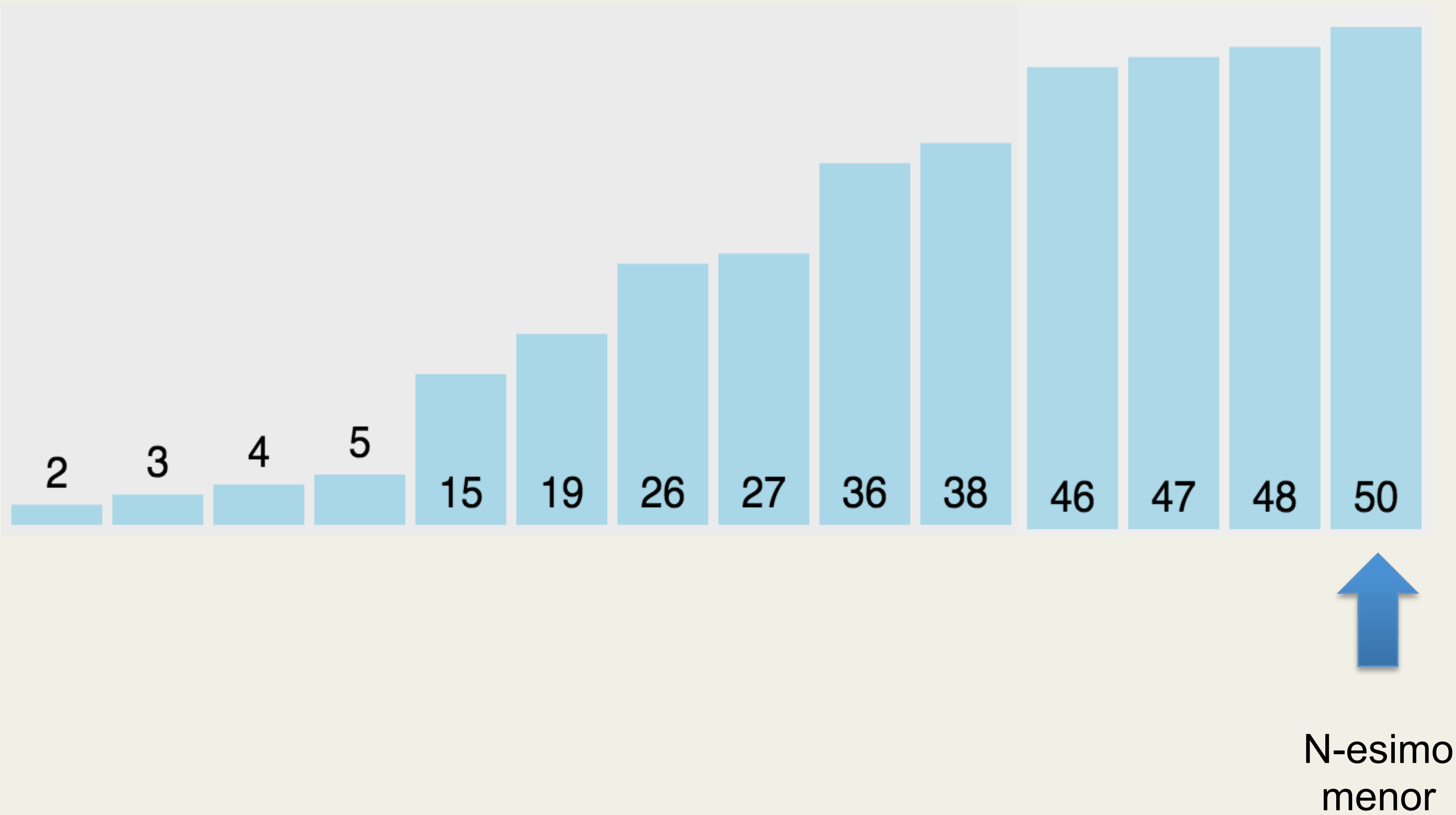
1º menor

Ideia geral



2º menor

Ideia geral



Ordenação por seleção

- Ideia geral:
  - Dado um vetor de entrada v[N]
    - Selecione o 1º menor elemento e coloque-o na 1ª posição
    - Selecione o 2º menor elemento e coloque-o na 2ª posição
    - ...
    - Selecione o N-ésimo menor elemento e coloque-o na N-ésima posição

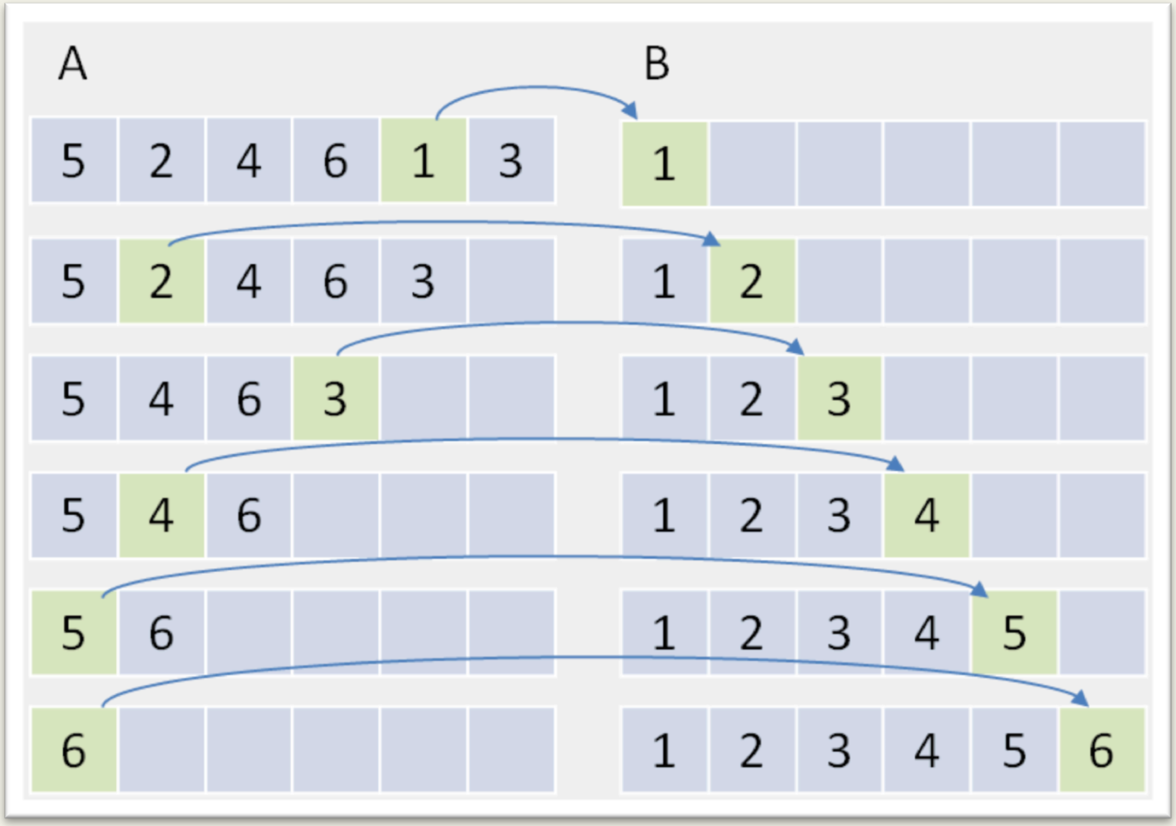
Ordenação por seleção

- Algoritmo: 1ª Versão

```
Sort( v[], N )
  DECLARE w[N]
  FOR i = 0; i < N; i++:
      min_i = select_i_min(v, i+1)
      w[i] = v[min_i]
  END_FOR
  v = w
END
```

Ordenação por seleção

- Algoritmo: 1ª Versão



Ordenação por seleção

- Algoritmo: 1ª Versão

```
Sort( v[], N )
   DECLARE w[N]
   FOR i = 0; i < N; i++:
       min_i = select_i_min(v, i+1)
       w[i] = v[min_i]
   END_FOR
   v = w
END
```

Uso desnecessário de espaço de memória extra

Ordenação por seleção

- Algoritmo: 2ª Versão (Sem memória extra)

```
SORT( v[], N ){
  FOR i = 0; i < N; i++ :
     min_i = MIN(v, i, N-1)
     Swap v[i], v[min_i]
  END_FOR
END
```

Basta selecionar o mínimo entre [i, N-1]

Ordenação por seleção

- Visualizar funcionamento em:
  - http://visualgo.net/sorting.html#

Ordenação por seleção

- Atividades:
  - Qual a complexidade do algoritmo de ordenação por seleção (Pior e melhor caso)?
  - Modifique o algoritmo abaixo para que não seja necessário fazer a chamada a função “min”

```
SORT( v[], N ){
  FOR i = 0; i < N; i++:
      min_i = MIN(v, i,N-1)
      Swap v[i], v[min_i]
  END_FOR
END
```

Ordenação por seleção

```
Sort( v[], N ):
  FOR i = 0; i < N; i++:
      min = i
      FOR j = i+1; j < N; j++:
            IF v[j] < v[min] :
                  min = j
            END_IF
      IF i ≠ min:
            Swap v[i], v[min]
      END_IF
  END_FOR
END
```

Prática

- Baixe os arquivos disponíveis no SIGAA
- Implemente o algoritmo de ordenação por seleção na função sort do arquivo SelectionSort.cpp
  - Programa espera dois parâmetros de entrada:
    - Arquivo de dados contendo números aleatórios
    - Quantidade de dados de entrada

Estrutura de Dados Básicas I.

Aula 7 – Algoritmos de ordenação I

Prof. Eiji Adachi M. Barbosa