Algoritmos de Ordenação e Busca Não-recursivos
MÓDULO 6 (adendo)

Priscila Machado Vieira Lima

Abril-2018

Universidade Federal do Rio de Janeiro

Roteiro

- **Ordenação**
- Algoritmos de Seleção e Inserção
- Busca
- Algoritmo de Busca Binária
- Considerações finais e resumo

Ordenação

Code	First name	Last name
9090	James	Smith
3131	James	Smith
1313	Emmy	Robertson
7007	James	Bond

Ordenação de Listas Lineares

- "Dada uma lista linear **L** com **n** elementos, deseja-se ordená-los em ordem crescente através da troca de suas posições originais".

- Exemplo
  - Lista L, n = 8

10	5	9	-2	13	-8	4	-5

  - Lista L ordenada, crescentemente

-8	-5	-2	4	5	9	10	13

Ordenação

- Idéia???

Roteiro

- Ordenação
- Algoritmos de **Seleção** e Inserção
- Busca
- Algoritmo de Busca Binária
- Considerações finais e resumo

Ordenação por Seleção

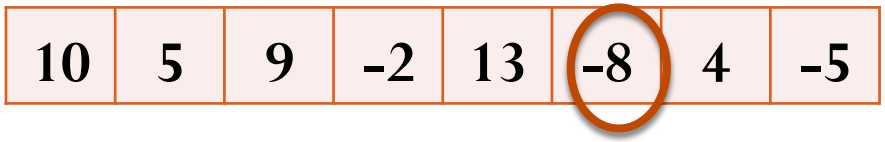
- Idéia
  - SELECIONAR o menor elemento da lista
  - Trocá-lo com o primeiro
  - Repetir o procedimento para o segundo elemento
    - E assim sucessivamente….

- Exemplo: V =

10	5	9	-2	13	-8	4	-5

Ordenação por Seleção

- Elemento assinalado é o menor da parte não-ordenada



Ordenação por Seleção

- Elemento assinalado é o menor da parte não-ordenada

10	5	9	-2	13	-8	4	-5
-8	5	9	-2	13	10	4	-5

- Parte de **V** acinzentada é a já ordenada

Ordenação por Seleção

- Elemento assinalado é o menor da parte não-ordenada

- Parte de **V** acinzentada é a já ordenada

10	5	9	-2	13	-8	4	-5
-8	5	9	-2	13	10	4	-5
-8	-5	9	-2	13	10	4	5

Ordenação por Seleção

- Elemento assinalado é o menor da parte não-ordenada

- Parte de **V** acinzentada é a já ordenada

10	5	9	-2	13	-8	4	-5
-8	5	9	-2	13	10	4	-5
-8	-5	9	-2	13	10	4	5
-8	-5	-2	9	13	10	4	5

Ordenação por Seleção

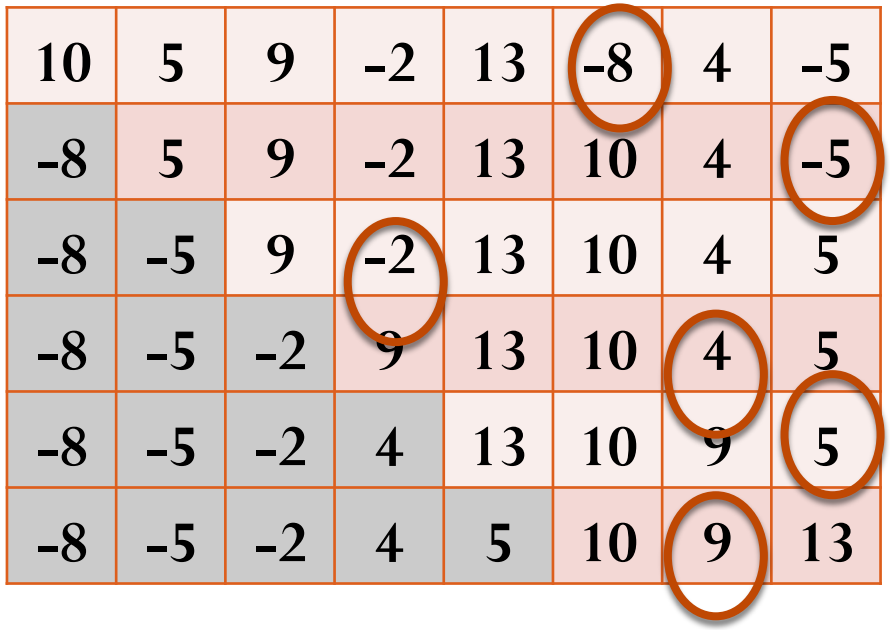
- Elemento assinalado é o menor da parte não-ordenada

- Parte de **V** acinzentada é a já ordenada

10	5	9	-2	13	-8	4	-5
-8	5	9	-2	13	10	4	-5
-8	-5	9	-2	13	10	4	5
-8	-5	-2	9	13	10	4	5
-8	-5	-2	4	13	10	9	5

Ordenação por Seleção

- Elemento assinalado é o menor da parte não-ordenada
- Parte de **V** acinzentada é a já ordenada



Ordenação por Seleção

- Elemento assinalado é o menor da parte não-ordenada
- Parte de **V** acinzentada é a já ordenada

10	5	9	-2	13	-8	4	-5
-8	5	9	-2	13	10	4	-5
-8	-5	9	-2	13	10	4	5
-8	-5	-2	9	13	10	4	5
-8	-5	-2	4	13	10	9	5
-8	-5	-2	4	5	10	9	13
-8	-5	-2	4	5	9	10	13

Ordenação por Seleção

- Elemento assinalado é o menor da parte não-ordenada
- Parte de **V** acinzentada é a já ordenada

10	5	9	-2	13	-8	4	-5
-8	5	9	-2	13	10	4	-5
-8	-5	9	-2	13	10	4	5
-8	-5	-2	9	13	10	4	5
-8	-5	-2	4	13	10	9	5
-8	-5	-2	4	5	10	9	13
-8	-5	-2	4	5	9	10	13
-8	-5	-2	4	5	9	10	13

Ordenação por Seleção

- Elemento assinalado é o menor da parte não-ordenada
- Parte de **V** acinzentada é a já ordenada
- **V** ordenado! ➔

10	5	9	-2	13	-8	4	-5
-8	5	9	-2	13	10	4	-5
-8	-5	9	-2	13	10	4	5
-8	-5	-2	9	13	10	4	5
-8	-5	-2	4	13	10	9	5
-8	-5	-2	4	5	10	9	13
-8	-5	-2	4	5	9	10	13
-8	-5	-2	4	5	9	10	13
-8	-5	-2	4	5	9	10	13

Ordenação por Seleção

- Entrada: Vetor **V** com **n** elementos desordenados

```
para i←0; i < n-1; i++ % considerar a lista da posição i em
 inicio_para           % diante
   menor ← i           % para cada valor de i
   para j←i+1; j < n; j++ % procurar o menor
     inicio_para
    (*) …se V[j] < V[menor]
            então menor ← j
     fim_para
   aux ← V[i]      % fazer a troca (pode testar se i ≠ menor)
   V[i] ← V[menor]
   V[menor] ← aux
 fim_para
```

- Complexidade: Nro de comparações da linha (*) = $\theta(n^2)$

Ordenação

- Outra idéia???

Ordenação

Roteiro

- Ordenação
- Algoritmos de Seleção e **Inserção**
- Considerações finais e resumo

Ordenação por Inserção

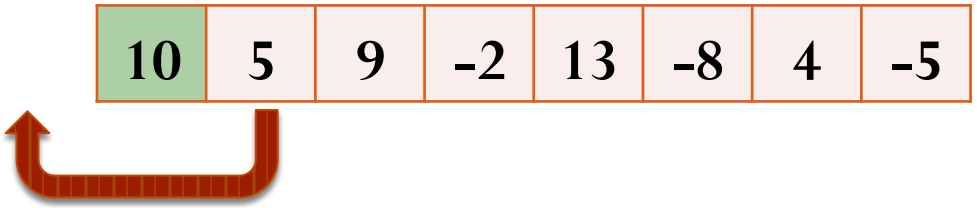
- Idéia: Inicializar a lista ordenada contendo só o primeiro elemento:
  - REMOVER primeiro elemento da lista ainda desordenada
  - INSERIR o elemento no local correto da lista já ordenada
  - Mover os elementos restantes da lista ordenada para a direita
  - Repetir o procedimento para o segundo elemento
    - E assim sucessivamente….

- Exemplo: V =

10	5	9	-2	13	-8	4	-5

Ordenação por Inserção

- Elemento de onde parte a seta é o que irá ser inserido

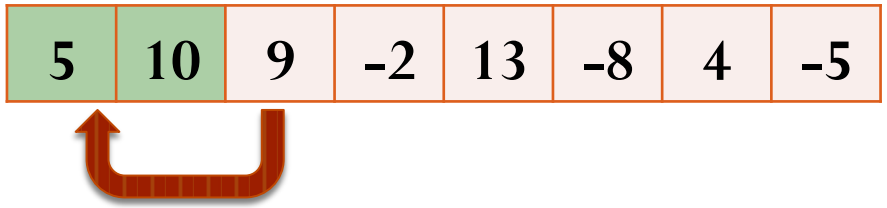


- A seta aponta para a posição de inserção

- Parte de **V** esverdeada é a já ordenada

Ordenação por Inserção

- Elemento de onde parte a seta é o que irá ser inserido

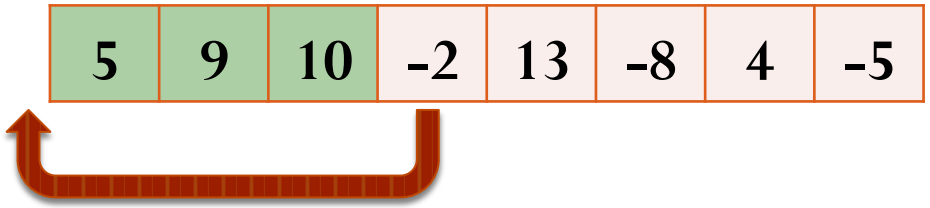


- A seta aponta para a posição de inserção

- Parte de **V** esverdeada é a já ordenada

Ordenação por Inserção

- Elemento de onde parte a seta é o que irá ser inserido

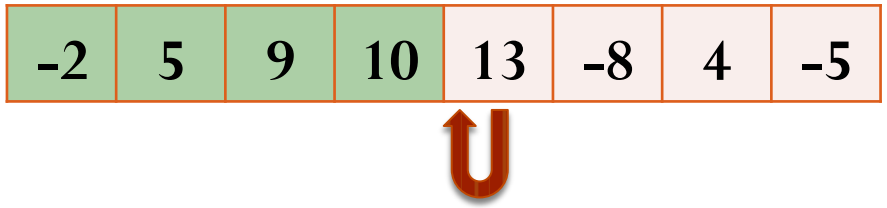


- A seta aponta para a posição de inserção

- Parte de **V** esverdeada é a já ordenada

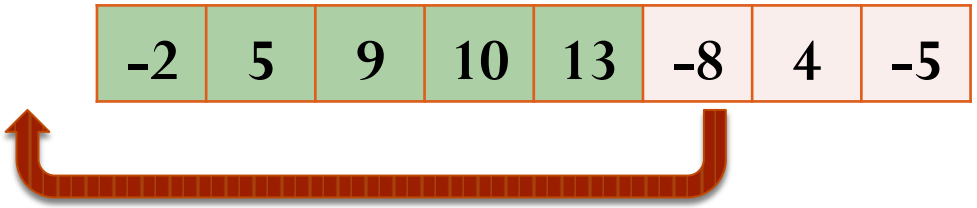
Ordenação por Inserção

- Elemento de onde parte a seta é o que irá ser inserido
- A seta aponta para a posição de inserção
- Parte de **V** esverdeada é a já ordenada



Ordenação por Inserção

- Elemento de onde parte a seta é o que irá ser inserido

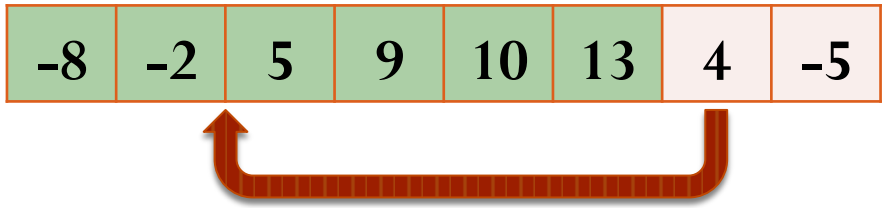


- A seta aponta para a posição de inserção

- Parte de **V** esverdeada é a já ordenada

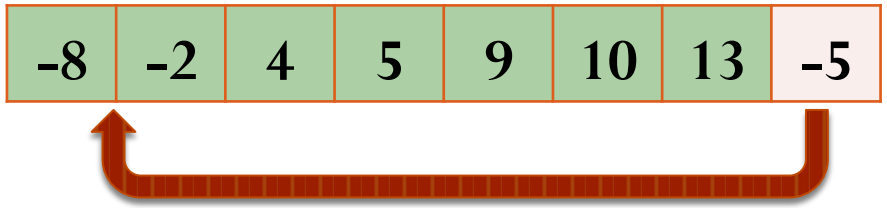
Ordenação por Inserção

- Elemento de onde parte a seta é o que irá ser inserido
- A seta aponta para a posição de inserção
- Parte de **V** esverdeada é a já ordenada



Ordenação por Inserção

- Elemento de onde parte a seta é o que irá ser inserido
- A seta aponta para a posição de inserção
- Parte de **V** esverdeada é a já ordenada



Ordenação por Inserção

- Elemento de onde parte a seta é o que irá ser inserido
- A seta aponta para a posição de inserção
- Parte de **V** esverdeada é a já ordenada

Ordenação por Inserção

- Entrada: Vetor **V** com **n** elementos desordenados

```
para j ← 1; j < n; j++ // considerar a lista desordenada
 início_para            // da posição j em diante
   chave ← V[j]
  // inserir V[j] na parte já ordenada da lista
   i ← j - 1
   enquanto (i > -1) AND (V[i] > chave)
     inicio_enquanto
       V[i+1] ← V[i]
       i← i - 1
       V[i+1] ← chave
     fim_enquanto

 fim_para
```

- Complexidade: $\theta(n^2)$

Ordenação por Inserção

- Incremental

Haveria outra estratégia para Projeto de Algoritmos?

Reflexões

- Algumas questões a considerar
  - Simplicidade
  - Custo da(s) instrução(ões) que se repete(m)
  - Ordenação "no-lugar" (*in-place*)
  - Algoritmo interno ou externo
  - Duplicatas mantidas em ordem ou não (estabilidade)

Exercícios

1. Ordenar as seguintes sequências pelos três algoritmos vistos:
   a) 7, 8, 1, 1, 1, 0, 9, 3
   b) 1, 2, 3, 4, 5, 6, 7
   c) 16, 15, 13, 12, 10, 9, 7
2. Pesquise versão(ões) não-recursiva(s) do Algoritmo Mergesort.

Roteiro

- Ordenação
- Algoritmos de Seleção e Inserção
- **Busca**
- Algoritmo de Busca Binária
- Considerações finais e resumo

Roteiro

- Ordenação
- Algoritmos de Seleção e Inserção
- Busca
- **Algoritmo de Busca Binária**
- Considerações finais e resumo

Busca Binária

- Entrada: Vetor **V** com **n** elementos ordenados elemento elem que se busca
- Complexidade: θ **(logn)**

Busca Binária

```
achei ← 0
prim ← 0
ult ← n - 1
meio ← (prim+ult)/2
 enquanto ((prim <= ult) AND (NOT achei)
   inicio_enqto
      se (V[meio] < elem) entao
         prim ← meio+ 1
      senão
         se (V[meio] == elem) entao
           inicio_entao
             escrever("%d encontrado em %d.\n", elem, meio+1)
             achei ← 1
           fim_entao
        senao
         ult ← meio - 1
     meio ← (prim + ult)/2
   fim_enqto
   se (prim > ult) entao
      escrever(”Nao encontrado! %d ausente da lista.\n", elem)
```

Roteiro

- Ordenação
- Algoritmos de Seleção e Inserção
- Busca
- Algoritmo de Busca Binária
- **Considerações finais e resumo**

Resumindo

- **Métodos de Ordenação incrementais**
  - Alternativa: Divisão e Conquista (usa recursividade)
  - Mergesort recursivo
  - (Recursão será vista junto com Funções e Procedimentos
- **É preciso considerar diversas características na escolha/projeto de algoritmo de ordenação**.

Bibliografia (1)

Em Português

- SCHMITZ, E. A. e A. A. S. TELES. Pascal e Técnicas de Programação. Terceira edição, Rio de Janeiro: LTC – Livros Técnicos e Científicos, 1985.
- SZWARCFITER, J. e L. MARQUEZON. Estruturas de Dados e Seus Algoritmos. Rio de Janeiro: LTC – Livros Técnicos e Científicos, 1994.
- VELOSO, P. A., C. S. Santos, P. Azevedo e A. L. Furtado. Estruturas de Dados. Rio de Janeiro: Editora Campus, 1993

Bibliografia (2)

- CORMEN, T. H., C. E. LEISERSON, R. L. RIVEST and C. STEIN. **Introduction o Algorithms**. Third edition, MIT Press, 2009.

  Obs: Livro mais avançado, para ser usado como consulta auxiliar.

- Knuth, D. E. **The Art of Computer Programming 1: Fundamental Algorithms**, Addison Wesley, Reading, Ma, 1968.

  Obs: Clássico.

- Knuth, D. E. **The Art of Computer Programming 3: Sorting and Searching**, Addison Wesley, Reading, Ma, 1973.

  Obs: Clássico.

- SEDGEWICK, R. **Algorithms**. First edition, Addison-Wesley, 1983.

  Obs: Ótimo capítulo sobre ordenação (Sorting). Edição com algoritmos em Pascal; existem edições outras mais recentes com códigos e C, C++ e MODULA, alem de uma com Java (em preparação).