Ordenação de dados

Profª. Rose Yuri Shimizu

Roteiro

Ordenação de dados - importância

- Ordenação é organização
- Organização otimiza as buscas
  - Lógica de sequencialidade: previsibilidade
- Ordenação de itens (arquivos, estruturas)
  - A chave é a parte do item utilizada como parâmetro/controle de ordenação

Recomendações

- RobertSedgewickAlgorithmsinC,AddisonWesley,3nded.
- Algorithms,4thEdition-RobertSedgewickeKevinWayne
- https://brunoribas.com.br/apostila-eda/ordenacao-elementar.html
- https://www.youtube.com/@ProfBrunoRibas
- https://www.ime.usp.br/~pf/algoritmos/aulas/ordena.html
- https://github.com/bcribas/benchmark-ordenacao

Roteiro

Algoritmos de Ordenação - Características

1. Complexidade (espacial, temporal)
   - Quadráticos: simples e suficiente para arquivos pequenos
   - Linearítmicos: mais complexos (*overhead*) e eficientes para arquivos grandes

Algoritmos de Ordenação - Características

1. Complexidade (espacial, temporal)
2. Estabilidade
   - Mantém a posição relativa dos elementos
   - Não há saltos
   - 2 4 1 6 7 *1*
   - *1* 1 2 4 6 7 : não-estável
   - 1 *1* 2 4 6 7 : estável

Algoritmos de Ordenação - Características

1. Complexidade (espacial, temporal)
2. Estabilidade
   - Mantém a posição relativa dos elementos
3. Adaptatividade
   - Aproveita a ordenação existente

Algoritmos de Ordenação - Características

1. Complexidade (espacial, temporal)
2. Estabilidade
   - Mantém a posição relativa dos elementos
3. Adaptatividade
   - Aproveita a ordenação existente
4. Memória extra
   - In-place:
     - ★ Utiliza a própria estrutura
     - ★ Utiliza memória extra: pilha de execução, variáveis auxiliares
   - Não in-place:
     - ★ Utiliza mais uma estrutura
     - ★ Cópias

Algoritmos de Ordenação - Características

1. Complexidade (espacial, temporal)
2. Estabilidade
   - Mantém a posição relativa dos elementos
3. Adaptatividade
   - Aproveita a ordenação existente
4. Memória extra
   - In-place:
     - ★ Utiliza a própria estrutura
     - ★ Utiliza memória extra: pilha de execução, variáveis auxiliares
5. Localização
   - Interna: todos os dados cabem na memória principal
   - Externa: arquivo grande; é ordenado em pedaços (chunks) que caibam na memória principal

Algoritmos de Ordenação - Elementares x Eficientes

- Elementares: custos maiores, mais simples
- Eficientes: custos menores, mais complexos (estratégias)
- Analise as constantes da função custo e o tamanho da entrada

$$f1(n) = n^2$$
$$f2(n) = x * n + y$$

- Array x Listas encadeadas
  - Métodos elementares: lidam bem com qualquer implementação
  - Métodos mais eficientes:
    - Array: mais fácil manipulação pelo acesso direto
    - Estruturas encadeadas: árvores ordenadas

Roteiro

Roteiro

Algoritmos de Ordenação Elementares
Selection Sort - selecionar e posicionar

1. Selecionar o menor item
2. Posicionar: troque com o primeiro item
3. Selecionar o segundo menor item
4. Posicionar: troque com o segundo item
5. Repita para os n elementos do vetor

Algoritmos de Ordenação Elementares
Selection Sort

- Selecionar v[j] < v[menor] ?

```
          l                                   r
          0     1     2     3     4     5
v     [ 3 | 2 | 4 | 6 | 1 | 5 ]      índice do menor = i = 0?
          i     j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Selecionar v[j] < v[menor] ?

```
          l                                   r
          0     1     2     3     4     5
v     [ 3 | 2 | 4 | 6 | 1 | 5 ]       índice do menor = j = 1
          i     j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Selecionar v[j] < v[menor] ?

```
          l                                 r
          0     1     2     3     4     5
v     [ 3 | 2 | 4 | 6 | 1 | 5 ]      índice do menor = 1?
          i           j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Selecionar v[j] < v[menor] ?

```
          l                                   r
          0     1     2     3     4     5
v     [ 3 | 2 | 4 | 6 | 1 | 5 ]       índice do menor = 1?
          i           j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Selecionar v[j] < v[menor] ?

```
          l                                   r
          0     1     2     3     4     5
v     [ 3 | 2 | 4 | 6 | 1 | 5 ]       índice do menor = 1?
          i                 j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Selecionar v[j] < v[menor] ?

```
          l                                   r
          0     1     2     3     4     5
v     [ 3 | 2 | 4 | 6 | 1 | 5 ]       índice do menor = j = 4
          i                 j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Selecionar v[j] < v[menor] ?

```
          l                                   r
          0     1     2     3     4     5
v     [ 3 | 2 | 4 | 6 | 1 | 5 ]        índice do menor = 4?
          i                                   j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Selecionar v[j] < v[menor] ?

```
          l                           r
          0    1    2    3    4    5
v     [ 3 | 2 | 4 | 6 | 1 | 5 ]      índice do menor = 4
          i                      j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Posicionar menor: troca(swap) v[i] $\leftrightarrow$ v[menor]

```
          l                                   r
          0     1     2     3     4     5
v     [ _3_ | 2 | 4 | 6 | _1_ | 5 ]       índice do menor = 4
          i                             j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Posicionar menor: troca(swap) v[i] $\leftrightarrow$ v[menor]

```
          l                                   r
          0     1     2     3     4     5
v     [ _1_ | 2 | 4 | 6 | _3_ | 5 ]       índice do menor = 4
          i                             j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Selecionar v[j] < v[menor] ?

```
          l                                   r
          0     1     2     3     4     5
v     [ 1 | 2 | 4 | 6 | 3 | 5 ]      índice do menor = i = 1?
                i     j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Selecionar v[j] < v[menor] ?

```
          l                                   r
          0     1     2     3     4     5
v     [ 1 | 2 | 4 | 6 | 3 | 5 ]       índice do menor = 1?
              i           j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Selecionar v[j] < v[menor] ?

```
          l                                   r
          0     1     2     3     4     5
v     [ 1 | 2 | 4 | 6 | 3 | 5 ]       índice do menor = 1?
              i                 j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Selecionar v[j] < v[menor] ?

```
          l                                   r
          0     1     2     3     4     5
v      [ 1 | 2 | 4 | 6 | 3 | 5 ]       índice do menor = 1?
                i                       j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Selecionar v[j] < v[menor] ?

```
          l                           r
          0    1    2    3    4    5
v     [ 1 | 2 | 4 | 6 | 3 | 5 ]      índice do menor = 1?
               i                   j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Posicionar menor: v[i] == v[menor] ? sem swap

```
          l                             r
          0    1    2    3    4    5
v     [ 1 | 2 | 4 | 6 | 3 | 5 ]      índice do menor = 1?
               i                   j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Selecionar v[j] < v[menor] ?

```
          l                                   r
          0     1     2     3     4     5
v     [ 1 | 2 | 4 | 6 | 3 | 5 ]      índice do menor = i = 2?
                    i     j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Selecionar v[j] < v[menor] ?

```
          l                                   r
          0     1     2     3     4     5
v     [ 1 | 2 | 4 | 6 | 3 | 5 ]       índice do menor = 2?
                    i           j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Selecionar v[j] < v[menor] ?

```
          l                                   r
          0     1     2     3     4     5
v     [ 1 | 2 | 4 | 6 | 3 | 5 ]       índice do menor = j = 4
                      i           j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Selecionar v[j] < v[menor] ?

```
          l                                   r
          0     1     2     3     4     5
v      [ 1 | 2 | 4 | 6 | 3 | 5 ]       índice do menor = 4?
                    i           j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Selecionar v[j] < v[menor] ?

```
          l                           r
          0    1    2    3    4    5
v     [ 1 | 2 | 4 | 6 | 3 | 5 ]      índice do menor = 4
                   i                   j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Posicionar menor: troca(swap) v[i] $\leftrightarrow$ v[menor]

```
          l                                   r
          0    1     2     3     4     5
v     [ 1 | 2 | _4_ | 6 | _3_ | 5 ]       índice do menor = 4
                    i                   j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Posicionar menor: troca(swap) v[i] $\leftrightarrow$ v[menor]

```
          l                                   r
          0    1     2     3     4     5
v     [ 1 | 2 | _3_ | 6 | _4_ | 5 ]       índice do menor = 4
                    i                 j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Selecionar v[j] < v[menor] ?

```
          l                                   r
          0     1     2     3     4     5
v      [ 1 | 2 | 3 | 6 | 4 | 5 ]       índice do menor = 3?
                        i     j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Selecionar v[j] < v[menor] ?

```
          l                                   r
          0     1     2     3     4     5
v     [ 1 | 2 | 3 | 6 | 4 | 5 ]       índice do menor = j = 4
                        i     j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Selecionar v[j] < v[menor] ?

```
          l                                   r
          0     1     2     3     4     5
v     [ 1 | 2 | 3 | 6 | 4 | 5 ]       índice do menor = 4?
                        i           j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Selecionar v[j] < v[menor] ?

```
          l                                   r
          0     1     2     3     4     5
v      [ 1 | 2 | 3 | 6 | 4 | 5 ]        índice do menor = 4?
                        i           j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Posicionar menor: troca(swap) v[i] $\leftrightarrow$ v[menor]

```
          l                                   r
          0    1    2     3      4     5
v     [ 1 | 2 | 3 | _6_ | _4_ | 5 ]       índice do menor = 4
                       i               j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Posicionar menor: troca(swap) v[i] $\leftrightarrow$ v[menor]

```
          l                                   r
          0    1    2     3      4     5
v     [ 1 | 2 | 3 | _4_ | _6_ | 5 ]       índice do menor = 4
                       i                j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Selecionar v[j] < v[menor] ?

```
          l                                   r
          0     1     2     3     4     5
v     [ 1 | 2 | 3 | 4 | 6 | 5 ]       índice do menor = i = 4?
                                  i     j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Selecionar v[j] < v[menor] ?

```
          l                                   r
          0     1     2     3     4     5
v     [ 1 | 2 | 3 | 4 | 6 | 5 ]       índice do menor = j = 5
                                 i     j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Selecionar v[j] < v[menor] ?

```
          l                                r
          0    1    2    3    4    5
v     [ 1 | 2 | 3 | 4 | 6 | 5 ]      índice do menor = 5?
                              i         j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Posicionar menor: troca(swap) v[i] $\leftrightarrow$ v[menor]

```
          l                                   r
          0    1    2    3    4    5
v     [ 1 | 2 | 3 | 4 | _6_ | _5_ ]      índice do menor = 5
                              i     j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Posicionar menor: troca(swap) v[i] $\leftrightarrow$ v[menor]

```
          l                                   r
          0    1    2    3    4    5
v     [ 1 | 2 | 3 | 4 | _5_ | _6_ ]      índice do menor = 5
                          i          j
```

Algoritmos de Ordenação Elementares
Selection Sort

- Terminou? Vetor ordenado.

```
          l                             r
          0    1    2    3    4    5
v     [ 1 | 2 | 3 | 4 | 5 | 6 ]      índice do menor = i = 5
                                 i    j
```

Algoritmos de Ordenação Elementares
Selection Sort

```
1  void selection_sort(int v[], int l, int r){
2      int menor;
3      for(int i=l; i<r; i++){
4          menor = i;
5
6          for(int j=i+1; j<=r; j++)
7              if(v[j] < v[menor])
8                  menor = j;
9
10         if(i != menor)
11             exch(v[i], v[menor])
12     }
13 }
14
```

Algoritmos de Ordenação Elementares
Selection Sort

```
1  void selection_sort(int v[], int l, int r){
2      int menor;
3      //n
4      for(int i=l; i<r; i++){
5          menor = i;
6
7          //(n-1), (n-2), (n-3), .. , 0
8          //PA ((n+0)n)/2 = (n^2)/2
9          for(int j=i+1; j<=r; j++)
10             if(v[j] < v[menor])
11                 menor = j;
12
13         if(i != menor)
14             exch(v[i], v[menor]) //n
15     }
16     //f(n) = (n^2)/2 + n
17 }
18
```

Algoritmos de Ordenação Elementares
Selection Sort

- Complexidade assintótica?
- Adaptatividade?
- Estabilidade?
- *In-place*?

Algoritmos de Ordenação Elementares
Selection Sort

- Complexidade assintótica?
  - Cerca de $\frac{N^2}{2}$ comparações e N trocas: $O(N^2)$
- Adaptatividade?
- Estabilidade?
- *In-place*?

Algoritmos de Ordenação Elementares
Selection Sort

- Complexidade assintótica?
  - Cerca de $\frac{N^2}{2}$ comparações e N trocas: $O(N^2)$
- Adaptatividade?
  - Se o primeiro item já for o menor, implica que não é necessário percorrer o vetor na primeira passada?!
- Estabilidade?
- *In-place*?

Algoritmos de Ordenação Elementares
Selection Sort

- Complexidade assintótica?
  - Cerca de $\frac{N^2}{2}$ comparações e N trocas: $O(N^2)$
- Adaptatividade?
  - Se o primeiro item já for o menor, implica que não é necessário percorrer o vetor na primeira passada?!
  - Não, portanto, não é adaptativo.
- Estabilidade?
- *In-place*?

Algoritmos de Ordenação Elementares
Selection Sort

- Complexidade assintótica?
  - Cerca de $\frac{N^2}{2}$ comparações e N trocas: $O(N^2)$
- Adaptatividade?
  - Se o primeiro item já for o menor, implica que não é necessário percorrer o vetor na primeira passada?!
  - Não, portanto, não é adaptativo.
- Estabilidade?
  - 4 3 4' 1 $\rightarrow$ mantém a ordem relativa?
- *In-place*?

Algoritmos de Ordenação Elementares
Selection Sort

- Complexidade assintótica?
  - Cerca de $\frac{N^2}{2}$ comparações e N trocas: $O(N^2)$
- Adaptatividade?
  - Se o primeiro item já for o menor, implica que não é necessário percorrer o vetor na primeira passada?!
  - Não, portanto, não é adaptativo.
- Estabilidade?
  - 4 3 4' 1 $\rightarrow$ mantém a ordem relativa?
  - Tem trocas com saltos?
- *In-place*?

Algoritmos de Ordenação Elementares
Selection Sort

- Complexidade assintótica?
  - Cerca de $\frac{N^2}{2}$ comparações e N trocas: $O(N^2)$
- Adaptatividade?
  - Se o primeiro item já for o menor, implica que não é necessário percorrer o vetor na primeira passada?!
  - Não, portanto, não é adaptativo.
- Estabilidade?
  - 4 3 4' 1 $\rightarrow$ mantém a ordem relativa?
  - Tem trocas com saltos?
    - ★ 1 3 4' 4
- *In-place*?

Algoritmos de Ordenação Elementares
Selection Sort

- Complexidade assintótica?
  - Cerca de $\frac{N^2}{2}$ comparações e N trocas: $O(N^2)$
- Adaptatividade?
  - Se o primeiro item já for o menor, implica que não é necessário percorrer o vetor na primeira passada?!
  - Não, portanto, não é adaptativo.
- Estabilidade?
  - 4 3 4' 1 $\rightarrow$ mantém a ordem relativa?
  - Tem trocas com saltos?
    - ⋆ 1 3 4' 4
  - Não mantém a ordem: não estável.
- *In-place*?

Algoritmos de Ordenação Elementares
Selection Sort

- Complexidade assintótica?
  - Cerca de $\frac{N^2}{2}$ comparações e N trocas: $O(N^2)$
- Adaptatividade?
  - Se o primeiro item já for o menor, implica que não é necessário percorrer o vetor na primeira passada?!
  - Não, portanto, não é adaptativo.
- Estabilidade?
  - 4 3 4' 1 $\rightarrow$ mantém a ordem relativa?
  - Tem trocas com saltos?
    - ⋆ 1 3 4' 4
  - Não mantém a ordem: não estável.
- *In-place*?
  - Utiliza memória extra significativa?

Algoritmos de Ordenação Elementares
Selection Sort

- Complexidade assintótica?
  - Cerca de $\frac{N^2}{2}$ comparações e N trocas: $O(N^2)$
- Adaptatividade?
  - Se o primeiro item já for o menor, implica que não é necessário percorrer o vetor na primeira passada?!
  - Não, portanto, não é adaptativo.
- Estabilidade?
  - 4 3 4' 1 $\rightarrow$ mantém a ordem relativa?
  - Tem trocas com saltos?
    - ⋆ 1 3 4' 4
  - Não mantém a ordem: não estável.
- *In-place*?
  - Utiliza memória extra significativa?
  - Copia os conteúdos para outra estrutura de dados?

Algoritmos de Ordenação Elementares
Selection Sort

- Complexidade assintótica?
  - Cerca de $\frac{N^2}{2}$ comparações e N trocas: $O(N^2)$
- Adaptatividade?
  - Se o primeiro item já for o menor, implica que não é necessário percorrer o vetor na primeira passada?!
  - Não, portanto, não é adaptativo.
- Estabilidade?
  - 4 3 4' 1 $\rightarrow$ mantém a ordem relativa?
  - Tem trocas com saltos?
    - ⋆ 1 3 4' 4
  - Não mantém a ordem: não estável.
- *In-place*?
  - Utiliza memória extra significativa?
  - Copia os conteúdos para outra estrutura de dados?
  - Não, portanto, é *in-place*.

Algoritmos de Ordenação Elementares
Selection Sort

- Selection Sort estável??

- Selection Sort com listas encadeadas??

Algoritmos de Ordenação Elementares
Selection Sort

- Selection Sort estável??
  - Não realizar o swap

- Selection Sort com listas encadeadas??

Algoritmos de Ordenação Elementares
Selection Sort

- Selection Sort estável??
  - Não realizar o swap
  - Ideia: "abrir" um espaço na posição, "empurrando" os itens para frente
  - Boa solução?
- Selection Sort com listas encadeadas??

Algoritmos de Ordenação Elementares
Selection Sort

- Selection Sort estável??
  - Não realizar o swap
  - Ideia: "abrir" um espaço na posição, "empurrando" os itens para frente
  - Boa solução?
- Selection Sort com listas encadeadas??
  - Percorre a lista sequencialmente

Roteiro

Algoritmos de Ordenação Elementares
Bubble Sort - flutue o maior

1. Do início, flutuar o item
2. Ao achar uma "bolha" maior, esta passa a flutuar
3. No fim, o maior (ou menor) está no topo: topo–;
4. Volte para o item 1

Algoritmos de Ordenação Elementares
Bubble Sort - flutue o maior

- Comparar adjacentes v[j] > v[j+1] ?

```
          l                           r
          0    1    2    3    4    5
v     [ 3 | 2 | 4 | 6 | 1 | 5 ]
          j   j+1
```

Algoritmos de Ordenação Elementares
Bubble Sort - flutue o maior

- Comparar adjacentes v[j] > v[j+1] ? Flutua (swap)

```
          l                           r
          0    1    2    3    4    5
v     [ 2 | 3 | 4 | 6 | 1 | 5 ]
          j   j+1
```

Algoritmos de Ordenação Elementares
Bubble Sort - flutue o maior

- Comparar adjacentes v[j] > v[j+1] ?

```
          l                             r
          0    1    2    3    4    5
v     [ 2 | 3 | 4 | 6 | 1 | 5 ]
               j   j+1
```

Algoritmos de Ordenação Elementares
Bubble Sort - flutue o maior

- Comparar adjacentes v[j] > v[j+1] ?

```
          l                                 r
          0     1     2     3     4     5
v      [ 2 | 3 | 4 | 6 | 1 | 5 ]
                    j   j+1
```

Algoritmos de Ordenação Elementares
Bubble Sort - flutue o maior

- Comparar adjacentes v[j] > v[j+1] ?

```
          l                                 r
          0     1     2     3     4     5
v      [ 2 | 3 | 4 | 6 | 1 | 5 ]
                        j     j+1
```

Algoritmos de Ordenação Elementares
Bubble Sort - flutue o maior

- Comparar adjacentes v[j] > v[j+1] ? Flutua (swap)

```
          l                                 r
          0     1     2     3     4     5
v     [ 2 | 3 | 4 | 1 | 6 | 5 ]
                        j     j+1
```

Algoritmos de Ordenação Elementares
Bubble Sort - flutue o maior

- Comparar adjacentes v[j] > v[j+1] ?

```
          l                                 r
          0     1     2     3     4     5
v      [ 2 | 3 | 4 | 1 | 6 | 5 ]
                              j    j+1
```

Algoritmos de Ordenação Elementares
Bubble Sort - flutue o maior

- Comparar adjacentes v[j] > v[j+1] ? Flutua (swap)
- A cada flutuação, um elemento é posicionado corretamente (topo)

```
          l                                r
          0    1    2    3    4    5
v     [ 2 | 3 | 4 | 1 | 5 | _6_ ]
                          j    j+1
```

Algoritmos de Ordenação Elementares
Bubble Sort - flutue o maior

- Comparar adjacentes v[j] > v[j+1] ?

```
          l                       r
          0    1    2    3    4    5
v     [ 2 | 3 | 4 | 1 | 5 | 6 ]
          j   j+1
```

Algoritmos de Ordenação Elementares
Bubble Sort - flutue o maior

- Comparar adjacentes v[j] > v[j+1] ?

```
          l                       r
          0    1    2    3    4    5
v     [ 2 | 3 | 4 | 1 | 5 | 6 ]
               j   j+1
```

Algoritmos de Ordenação Elementares
Bubble Sort - flutue o maior

- Comparar adjacentes v[j] > v[j+1] ?

```
          l                       r
          0    1    2    3    4    5
v     [ 2 | 3 | 4 | 1 | 5 | 6 ]
                   j   j+1
```

Algoritmos de Ordenação Elementares
Bubble Sort - flutue o maior

- Comparar adjacentes v[j] > v[j+1] ? Flutua (swap)

```
          l                       r
          0    1    2    3    4    5
v     [ 2 | 3 | 1 | 4 | 5 | 6 ]
                   j   j+1
```

Algoritmos de Ordenação Elementares
Bubble Sort - flutue o maior

- Comparar adjacentes v[j] > v[j+1] ?

```
          l                       r
          0    1    2    3    4    5
v     [ 2 | 3 | 1 | 4 | 5 | 6 ]
                      j   j+1
```

Algoritmos de Ordenação Elementares
Bubble Sort - flutue o maior

- Comparar adjacentes v[j] > v[j+1] ?

```
          l                       r
          0    1    2    3     4     5
v     [ 2 | 3 | 1 | 4 | _5_ | 6 ]
                      j    j+1
```

Algoritmos de Ordenação Elementares
Bubble Sort - flutue o maior

- Comparar adjacentes v[j] > v[j+1] ?

```
          l                   r
          0     1     2     3     4     5
v      [ 2 | 3 | 1 | 4 | 5 | 6 ]
          j   j+1
```

Algoritmos de Ordenação Elementares
Bubble Sort - flutue o maior

- Comparar adjacentes v[j] > v[j+1] ?

```
          l                   r
          0    1    2    3    4    5
v     [ 2 | 3 | 1 | 4 | 5 | 6 ]
               j   j+1
```

Algoritmos de Ordenação Elementares
Bubble Sort - flutue o maior

- Comparar adjacentes v[j] > v[j+1] ? Flutua (swap)

```
          l                   r
          0    1    2    3    4    5
v     [ 2 | 1 | 3 | 4 | 5 | 6 ]
               j   j+1
```

Algoritmos de Ordenação Elementares
Bubble Sort - flutue o maior

- Comparar adjacentes v[j] > v[j+1] ?

```
          l                   r
          0    1    2    3    4    5
v     [ 2 | 1 | 3 | 4 | 5 | 6 ]
                   j   j+1
```

Algoritmos de Ordenação Elementares
Bubble Sort - flutue o maior

- Comparar adjacentes v[j] > v[j+1] ?

```
          l                       r
          0     1     2     3     4     5
v     [ 2 | 1 | 3 | _4_ | 5 | 6 ]
                  j     j+1
```

Algoritmos de Ordenação Elementares
Bubble Sort - flutue o maior

- Comparar adjacentes v[j] > v[j+1] ?

```
          l           r
          0   1   2   3   4   5
v     [ 2 | 1 | 3 | 4 | 5 | 6 ]
          j  j+1
```

Algoritmos de Ordenação Elementares
Bubble Sort - flutue o maior

- Comparar adjacentes v[j] > v[j+1] ? Flutua (swap)

```
          l           r
          0   1   2   3   4   5
v     [ 1 | 2 | 3 | 4 | 5 | 6 ]
          j  j+1
```

Algoritmos de Ordenação Elementares
Bubble Sort - flutue o maior

- Comparar adjacentes v[j] > v[j+1] ?

```
          l           r
          0    1      2      3    4    5
v      [ 1  |  2  |  _3_  |  4  |  5  |  6 ]
               j     j+1
```

Algoritmos de Ordenação Elementares
Bubble Sort - flutue o maior

- Comparar adjacentes v[j] > v[j+1] ?

```
          l   r
          0   1    2   3   4   5
v     [ 1 | 2 | 3 | 4 | 5 | 6 ]
          j  j+1
```

Algoritmos de Ordenação Elementares
Bubble Sort - flutue o maior

- Comparar adjacentes v[j] > v[j+1] ?

```
          l     r
          0     1     2     3     4     5
v     [ 1 | _2_ | 3 | 4 | 5 | 6 ]
          j    j+1
```

Algoritmos de Ordenação Elementares
Bubble Sort - flutue o maior

- Comparar adjacentes v[j] > v[j+1] ?

```
          l r
           0     1    2    3    4    5
v      [ _1_ | 2 | 3 | 4 | 5 | 6 ]
           j    j+1
```

Algoritmos de Ordenação Elementares
Bubble Sort

```
1  void bubble_sort(int v[], int l, int r){
2
3      for(; r>l; r--) {
4          for(int j=l; j<r; j++) {
5              if(v[j] > v[j+1]) {
6                  exch(v[j], v[j+1])
7              }
8          }
9      }
10
11 }
```

Algoritmos de Ordenação Elementares
Bubble Sort

```
1  void bubble_sort(int v[], int l, int r){
2
3      //n
4      for(; r>l; r--) {
5
6          //(n-1), (n-2), (n-3), .. , 0
7          //PA ((n+0)n)/2 = (n^2)/2
8          for(int j=l; j<r; j++) {
9
10             if(v[j] > v[j+1]) {
11
12                 //(n-1), (n-2), (n-3), .. , 0
13                 //PA ((n+0)n)/2 = (n^2)/2
14                 exch(v[j], v[j+1])
15             }
16         }
17     }
18     //f(n) = (n^2)/2 + (n^2)/2
19 }
```

Algoritmos de Ordenação Elementares
Bubble Sort

- Complexidade assintótica?
- Adaptatividade?
- Estabilidade?
- *In-place*?

Algoritmos de Ordenação Elementares
Bubble Sort

- Complexidade assintótica?
  - Cerca de $\frac{N^2}{2}$ comparações e $\frac{N^2}{2}$ trocas: $O(N^2)$
  - Melhor caso: $O(N)$ (como?)
- Adaptatividade?
- Estabilidade?
- *In-place*?

Algoritmos de Ordenação Elementares
Bubble Sort

- Complexidade assintótica?
  - Cerca de $\frac{N^2}{2}$ comparações e $\frac{N^2}{2}$ trocas: $O(N^2)$
  - Melhor caso: $O(N)$ (como?)
- Adaptatividade?
  - Ordenação diminui processamento?
- Estabilidade?
- *In-place*?

Algoritmos de Ordenação Elementares
Bubble Sort

- Complexidade assintótica?
  - Cerca de $\frac{N^2}{2}$ comparações e $\frac{N^2}{2}$ trocas: $O(N^2)$
  - Melhor caso: $O(N)$ (como?)
- Adaptatividade?
  - Ordenação diminui processamento?
  - Sim, portanto, é adaptativo.
- Estabilidade?
- *In-place*?

Algoritmos de Ordenação Elementares
Bubble Sort

- Complexidade assintótica?
  - Cerca de $\frac{N^2}{2}$ comparações e $\frac{N^2}{2}$ trocas: $O(N^2)$
  - Melhor caso: $O(N)$ (como?)
- Adaptatividade?
  - Ordenação diminui processamento?
  - Sim, portanto, é adaptativo.
- Estabilidade?
  - 2 4 3 4' 1 $\rightarrow$ mantém a ordem relativa?
- *In-place*?

Algoritmos de Ordenação Elementares
Bubble Sort

- Complexidade assintótica?
  - Cerca de $\frac{N^2}{2}$ comparações e $\frac{N^2}{2}$ trocas: $O(N^2)$
  - Melhor caso: $O(N)$ (como?)
- Adaptatividade?
  - Ordenação diminui processamento?
  - Sim, portanto, é adaptativo.
- Estabilidade?
  - 2 4 3 4' 1 $\rightarrow$ mantém a ordem relativa?
  - Tem trocas com saltos?
- *In-place*?

Algoritmos de Ordenação Elementares
Bubble Sort

- Complexidade assintótica?
  - Cerca de $\frac{N^2}{2}$ comparações e $\frac{N^2}{2}$ trocas: $O(N^2)$
  - Melhor caso: $O(N)$ (como?)
- Adaptatividade?
  - Ordenação diminui processamento?
  - Sim, portanto, é adaptativo.
- Estabilidade?
  - 2 4 3 4' 1 $\rightarrow$ mantém a ordem relativa?
  - Tem trocas com saltos?
    - ⋆ 2 3 4 1 4'
- *In-place*?

Algoritmos de Ordenação Elementares
Bubble Sort

- Complexidade assintótica?
  - Cerca de $\frac{N^2}{2}$ comparações e $\frac{N^2}{2}$ trocas: $O(N^2)$
  - Melhor caso: $O(N)$ (como?)
- Adaptatividade?
  - Ordenação diminui processamento?
  - Sim, portanto, é adaptativo.
- Estabilidade?
  - 2 4 3 4' 1 $\rightarrow$ mantém a ordem relativa?
  - Tem trocas com saltos?
    - ⋆ 2 3 4 1 4'
  - Mantém a ordem (não trocar os iguais): estável.
- *In-place*?

Algoritmos de Ordenação Elementares
Bubble Sort

- Complexidade assintótica?
  - Cerca de $\frac{N^2}{2}$ comparações e $\frac{N^2}{2}$ trocas: $O(N^2)$
  - Melhor caso: $O(N)$ (como?)
- Adaptatividade?
  - Ordenação diminui processamento?
  - Sim, portanto, é adaptativo.
- Estabilidade?
  - 2 4 3 4' 1 $\rightarrow$ mantém a ordem relativa?
  - Tem trocas com saltos?
    - ⋆ 2 3 4 1 4'
  - Mantém a ordem (não trocar os iguais): estável.
- *In-place*?
  - Utiliza memória extra significativa?

Algoritmos de Ordenação Elementares
Bubble Sort

- Complexidade assintótica?
  - Cerca de $\frac{N^2}{2}$ comparações e $\frac{N^2}{2}$ trocas: $O(N^2)$
  - Melhor caso: $O(N)$ (como?)
- Adaptatividade?
  - Ordenação diminui processamento?
  - Sim, portanto, é adaptativo.
- Estabilidade?
  - 2 4 3 4' 1 $\rightarrow$ mantém a ordem relativa?
  - Tem trocas com saltos?
    - ⋆ 2 3 4 1 4'
  - Mantém a ordem (não trocar os iguais): estável.
- *In-place*?
  - Utiliza memória extra significativa?
  - Copia os conteúdos para outra estrutura de dados?

Algoritmos de Ordenação Elementares
Bubble Sort

- Complexidade assintótica?
  - Cerca de $\frac{N^2}{2}$ comparações e $\frac{N^2}{2}$ trocas: $O(N^2)$
  - Melhor caso: $O(N)$ (como?)
- Adaptatividade?
  - Ordenação diminui processamento?
  - Sim, portanto, é adaptativo.
- Estabilidade?
  - 2 4 3 4' 1 $\rightarrow$ mantém a ordem relativa?
  - Tem trocas com saltos?
    - ⋆ 2 3 4 1 4'
  - Mantém a ordem (não trocar os iguais): estável.
- *In-place*?
  - Utiliza memória extra significativa?
  - Copia os conteúdos para outra estrutura de dados?
  - Não, portanto, é *in-place*.

Algoritmos de Ordenação Elementares
Bubble Sort

```
1  void bubble_sort(int v[], int l, int r){
2      int swap = 1;
3      for(; r>l && swap; r--) {
4          swap = 0;
5          for(int j=l; j<r; j++) {
6              if(v[j] > v[j+1]) {
7                  exch(v[j], v[j+1])
8                  swap = 1;
9              }
10         }
11     }
12 }
```

Algoritmos de Ordenação Elementares
Bubble Sort

- Selection Sort x Bubble sort?
- Bubble Sort com listas encadeadas??
- Otimização?

Algoritmos de Ordenação Elementares
Bubble Sort

- Selection Sort x Bubble sort?
  - Bubble sort é pior que o selection
  - Sempre?
  - Teste com as entradas “16-aleatorio” e “17-quaseordenado” do conjunto de testes
- Bubble Sort com listas encadeadas??
- Otimização?

Algoritmos de Ordenação Elementares
Bubble Sort

- Selection Sort x Bubble sort?
  - Bubble sort é pior que o selection
  - Sempre?
  - Teste com as entradas “16-aleatorio” e “17-quaseordenado” do conjunto de testes
- Bubble Sort com listas encadeadas??
  - Percorre a lista sequencialmente
- Otimização?

Algoritmos de Ordenação Elementares
Bubble Sort

- Selection Sort x Bubble sort?
  - Bubble sort é pior que o selection
  - Sempre?
  - Teste com as entradas "16-aleatorio" e "17-quaseordenado" do conjunto de testes
- Bubble Sort com listas encadeadas??
  - Percorre a lista sequencialmente
- Otimização?
  - Shaker sort: consiste em realizar uma iteração para colocar o menor elemento em cima e na volta colocar o maior elemento no fundo