Aula 16 – Ordenação por Seleção e Bolha

Norton T. Roman & Luciano A. Digiampietri
digiampietri@usp.br
@digiampietri

2023

Projeto por Indução Fraca

Segunda Alternativa

Projeto por Indução Fraca

Segunda Alternativa

- **Base**: $n = 1$. Um único elemento está ordenado

Projeto por Indução Fraca

Segunda Alternativa

- **Base**: $n = 1$. Um único elemento está ordenado
- **H.I.**: Sei ordenar um conjunto de $n - 1 \geq 1$ valores

Projeto por Indução Fraca

Segunda Alternativa

- **Base**: $n = 1$. Um único elemento está ordenado
- **H.I.**: Sei ordenar um conjunto de $n - 1 \geq 1$ valores
- **Passo**:

Projeto por Indução Fraca

Segunda Alternativa

- **Base**: $n = 1$. Um único elemento está ordenado
- **H.I.**: Sei ordenar um conjunto de $n - 1 \geq 1$ valores
- **Passo**:
  - Seja S um conjunto de $n \geq 2$ valores, e x o menor elemento de S

Projeto por Indução Fraca

Segunda Alternativa

- **Base**: $n = 1$. Um único elemento está ordenado
- **H.I.**: Sei ordenar um conjunto de $n - 1 \geq 1$ valores
- **Passo**:
  - Seja S um conjunto de $n \geq 2$ valores, e x o menor elemento de S
  - Então x certamente é o primeiro elemento da sequência ordenada de S. Por hipótese de indução, sabemos ordenar os demais $S - x$ elementos, e assim obtemos S ordenado

Projeto por Indução Fraca

Segunda Alternativa

- **Base**: $n = 1$. Um único elemento está ordenado
- **H.I.**: Sei ordenar um conjunto de $n - 1 \geq 1$ valores
- **Passo**:
  - Seja S um conjunto de $n \geq 2$ valores, e x o menor elemento de S
  - Então x certamente é o primeiro elemento da sequência ordenada de S. Por hipótese de indução, sabemos ordenar os demais $S - x$ elementos, e assim obtemos S ordenado
- Método da Seleção (*Selection Sort*)

Projeto por Indução Fraca

Método da Seleção (recursivo)

```
Seleção(A, ini, fim):
  Entrada: Arranjo A de n valores e os índices
           de início e término da sequência a ser ordenada
  Saída: Arranjo A ordenado

  se ini < fim então:
    min = ini
    para j = ini+1 até fim faça:
      se A[j] < A[min] então: min = j
    t = A[min]
    A[min] = A[ini]
    A[ini] = t
    Seleção(A, ini+1, fim)
```

Projeto por Indução Fraca

Método da Seleção (recursivo)

- Quantas comparações no arranjo são feitas no pior caso?

```
Seleção(A, ini, fim):
  se ini < fim então:
    min = ini
    para j = ini+1 até fim faça:
      se A[j] < A[min] então: min = j
    t = A[min]
    A[min] = A[ini]
    A[ini] = t
    Seleção(A, ini+1, fim)
```

Projeto por Indução Fraca

Método da Seleção (recursivo)

- Quantas comparações no arranjo são feitas no pior caso?

```
Seleção(A, ini, fim):
  se ini < fim então:
    min = ini
    para j = ini+1 até fim faça:
      se A[j] < A[min] então: min = j
    t = A[min]
    A[min] = A[ini]
    A[ini] = t
    Seleção(A, ini+1, fim)
```

$$T(n) = \begin{cases} \\ \end{cases}$$

se $n = 1$

para $n \geq 2$

Projeto por Indução Fraca

Método da Seleção (recursivo)

- Quantas comparações no arranjo são feitas no pior caso?

```
Seleção(A, ini, fim):
  se ini < fim então:
    min = ini
    para j = ini+1 até fim faça:
      se A[j] < A[min] então: min = j
    t = A[min]
    A[min] = A[ini]
    A[ini] = t
    Seleção(A, ini+1, fim)
```

$$T(n) = \begin{cases} O(1) & \text{se } n = 1 \\ & \text{para } n \geq 2 \end{cases}$$

Projeto por Indução Fraca

Método da Seleção (recursivo)

- Quantas comparações no arranjo são feitas no pior caso?

```
Seleção(A, ini, fim):
  se ini < fim então:
    min = ini
    para j = ini+1 até fim faça:
      se A[j] < A[min] então: min = j
    t = A[min]
    A[min] = A[ini]
    A[ini] = t
    Seleção(A, ini+1, fim)
```

$$T(n) = \begin{cases} O(1) & \text{se } n = 1 \\ O(n) & \text{para } n \geq 2 \end{cases}$$

Projeto por Indução Fraca

Método da Seleção (recursivo)

- Quantas comparações no arranjo são feitas no pior caso?

```
Seleção(A, ini, fim):
  se ini < fim então:
    min = ini
    para j = ini+1 até fim faça:
      se A[j] < A[min] então: min = j
    t = A[min]
    A[min] = A[ini]
    A[ini] = t
    Seleção(A, ini+1, fim)
```

$$T(n) = \begin{cases} O(1) & \text{se } n = 1 \\ O(n) + T(n-1) & \text{para } n \geq 2 \end{cases}$$

Projeto por Indução Fraca

Método da Seleção (recursivo)

- Quantas comparações no arranjo são feitas no pior caso?

```
Seleção(A, ini, fim):
  se ini < fim então:
    min = ini
    para j = ini+1 até fim faça:
      se A[j] < A[min] então: min = j
    t = A[min]
    A[min] = A[ini]
    A[ini] = t
    Seleção(A, ini+1, fim)
```

$$T(n) = \begin{cases} O(1) & \text{se } n = 1 \\ O(n) + T(n-1) & \text{para } n \geq 2 \end{cases} = O(n^2)$$

Projeto por Indução Fraca

Método da Seleção (recursivo)

- Quantas comparações no arranjo são feitas no melhor caso?

```
Seleção(A, ini, fim):
  se ini < fim então:
    min = ini
    para j = ini+1 até fim faça:
      se A[j] < A[min] então: min = j
    t = A[min]
    A[min] = A[ini]
    A[ini] = t
    Seleção(A, ini+1, fim)
```

Projeto por Indução Fraca

Método da Seleção (recursivo)

- Quantas comparações no arranjo são feitas no melhor caso?

```
Seleção(A, ini, fim):
  se ini < fim então:
    min = ini
    para j = ini+1 até fim faça:
      se A[j] < A[min] então: min = j
    t = A[min]
    A[min] = A[ini]
    A[ini] = t
    Seleção(A, ini+1, fim)
```

$$T(n) = \begin{cases} \\ \end{cases}$$

se $n = 1$

para $n \geq 2$

Projeto por Indução Fraca

Método da Seleção (recursivo)

- Quantas comparações no arranjo são feitas no melhor caso?

```
Seleção(A, ini, fim):
  se ini < fim então:
    min = ini
    para j = ini+1 até fim faça:
      se A[j] < A[min] então: min = j
    t = A[min]
    A[min] = A[ini]
    A[ini] = t
    Seleção(A, ini+1, fim)
```

$$T(n) = \begin{cases} O(1) & \text{se } n = 1 \\ & \text{para } n \geq 2 \end{cases}$$

Projeto por Indução Fraca

Método da Seleção (recursivo)

- Quantas comparações no arranjo são feitas no melhor caso?

```
Seleção(A, ini, fim):
  se ini < fim então:
    min = ini
    para j = ini+1 até fim faça:
      se A[j] < A[min] então: min = j
    t = A[min]
    A[min] = A[ini]
    A[ini] = t
    Seleção(A, ini+1, fim)
```

$$T(n) = \begin{cases} O(1) & \text{se } n = 1 \\ O(n) & \text{para } n \geq 2 \end{cases}$$

Projeto por Indução Fraca

Método da Seleção (recursivo)

- Quantas comparações no arranjo são feitas no melhor caso?

```
Seleção(A, ini, fim):
  se ini < fim então:
    min = ini
    para j = ini+1 até fim faça:
      se A[j] < A[min] então: min = j
    t = A[min]
    A[min] = A[ini]
    A[ini] = t
    Seleção(A, ini+1, fim)
```

$$T(n) = \begin{cases} O(1) & \text{se } n = 1 \\ O(n) + T(n-1) & \text{para } n \geq 2 \end{cases}$$

Projeto por Indução Fraca

Método da Seleção (recursivo)

- Quantas comparações no arranjo são feitas no melhor caso?

```
Seleção(A, ini, fim):
  se ini < fim então:
    min = ini
    para j = ini+1 até fim faça:
      se A[j] < A[min] então: min = j
    t = A[min]
    A[min] = A[ini]
    A[ini] = t
    Seleção(A, ini+1, fim)
```

$$T(n) = \begin{cases} O(1) & \text{se } n = 1 \\ O(n) + T(n-1) & \text{para } n \geq 2 \end{cases} = O(n^2)$$

Projeto por Indução Fraca

Método da Seleção (iterativo)

```
Seleção(A, n):
  Entrada: Arranjo A de n valores
  Saída: Arranjo A ordenado

  para i = 0 até n-2 faça:
    min = i
    para j = i+1 até n-1 faça:
      se A[j] < A[min] então: min = j
    t = A[min]
    A[min] = A[i]
    A[i] = t
```

Projeto por Indução Fraca

Método da Seleção (iterativo)

- Usando a notação O, quantas comparações no arranjo são feitas no pior caso?

```
Seleção(A, n):
  Entrada: Arranjo A de n valores
  Saída: Arranjo A ordenado

  para i = 0 até n-2 faça:
    min = i
    para j = i+1 até n-1 faça:
      se A[j] < A[min] então: min = j
    t = A[min]
    A[min] = A[i]
    A[i] = t
```

Projeto por Indução Fraca

Método da Seleção (iterativo)

- Usando a notação O, quantas comparações no arranjo são feitas no pior caso?
- $O(n)$

```
Seleção(A, n):
  Entrada: Arranjo A de n valores
  Saída: Arranjo A ordenado

  para i = 0 até n-2 faça:
    min = i
    para j = i+1 até n-1 faça:
      se A[j] < A[min] então: min = j
    t = A[min]
    A[min] = A[i]
    A[i] = t
```

Projeto por Indução Fraca

Método da Seleção (iterativo)

- Usando a notação O, quantas comparações no arranjo são feitas no pior caso?
- $O(n) \times O(n)$

```
Seleção(A, n):
  Entrada: Arranjo A de n valores
  Saída: Arranjo A ordenado

  para i = 0 até n-2 faça:
    min = i
    para j = i+1 até n-1 faça:
      se A[j] < A[min] então: min = j
    t = A[min]
    A[min] = A[i]
    A[i] = t
```

Projeto por Indução Fraca

Método da Seleção (iterativo)

- Usando a notação O, quantas comparações no arranjo são feitas no pior caso?

```
Seleção(A, n):
  Entrada: Arranjo A de n valores
  Saída: Arranjo A ordenado

  para i = 0 até n-2 faça:
    min = i
    para j = i+1 até n-1 faça:
      se A[j] < A[min] então: min = j
    t = A[min]
    A[min] = A[i]
    A[i] = t
```

- $O(n) \times O(n) = O(n^2)$

Projeto por Indução Fraca

Método da Seleção (iterativo)

- E quantas trocas de elementos são feitas no arranjo no pior caso?

```
Seleção(A, n):
  Entrada: Arranjo A de n valores
  Saída: Arranjo A ordenado

  para i = 0 até n-2 faça:
    min = i
    para j = i+1 até n-1 faça:
      se A[j] < A[min] então: min = j
    t = A[min]
    A[min] = A[i]
    A[i] = t
```

Projeto por Indução Fraca

Método da Seleção (iterativo)

- E quantas trocas de elementos são feitas no arranjo no pior caso?
- $O(n)$

```
Seleção(A, n):
  Entrada: Arranjo A de n valores
  Saída: Arranjo A ordenado

  para i = 0 até n-2 faça:
    min = i
    para j = i+1 até n-1 faça:
      se A[j] < A[min] então: min = j
    t = A[min]
    A[min] = A[i]
    A[i] = t
```

Projeto por Indução Fraca

Método da Seleção: recursivo × iterativo

```
Seleção(A, ini, fim):
  se ini < fim então:
    min = ini
    para j = ini+1 até fim faça:
      se A[j] < A[min] então:
        min = j
    t = A[min]
    A[min] = A[ini]
    A[ini] = t
    Seleção(A, ini+1, fim)
```

```
Seleção(A, n):
  para i = 0 até n-2 faça:
    min = i
    para j = i+1 até n-1 faça:
      se A[j] < A[min] então:
        min = j
    t = A[min]
    A[min] = A[i]
    A[i] = t
```

Projeto por Indução Fraca

Método da Seleção: recursivo × iterativo

```
void selection(int A[],
               int ini, int fim){
  int min, j, t;
  if (ini < fim){
    min = ini;
    for (j = ini+1; j<=fim; j++)
      if (A[j] < A[min]) min=j;
    t = A[min];
    A[min] = A[ini];
    A[ini] = t;
    selection(A, ini+1, fim);
  }
}
```

```
void selection(int A[], int n){
  int min, i, j, t;
  for(i=0; i<n-1; i++){
    min = i;
    for (j=i+1; j<n; j++)
      if (A[j] < A[min]) min=j;
    t = A[min];
    A[min] = A[i];
    A[i] = t;
  }
}
```

Projeto por Indução Fraca

Método da Seleção: recursivo × recursivo (alt.)

```
void selection(int A[],
               int ini, int fim){
  int min, j, t;
  if (ini < fim){
    min = ini;
    for (j = ini+1; j<=fim; j++)
      if (A[j] < A[min]) min=j;
    t = A[min];
    A[min] = A[ini];
    A[ini] = t;
    selection(A, ini+1, fim);
  }
}
```

```
void selection(int A[], int n){
  int max, j, t;
  if (n > 1){
    max = 0;
    for (j = 1; j<n; j++)
      if (A[j] > A[max]) max=j;
    t = A[max];
    A[max] = A[n-1];
    A[n-1] = t;
    selection(A, n-1);
  }
}
```

Projeto por Indução Fraca

Inserção × Seleção

- Inserção:
- Seleção:

Projeto por Indução Fraca

Inserção × Seleção

- Inserção:
  - Comparações: $O(n^2)$
- Seleção:

Projeto por Indução Fraca

Inserção $\times$ Seleção

- Inserção:
  - Comparações: $O(n^2)$
- Seleção:
  - Comparações: $O(n^2)$

Projeto por Indução Fraca

Inserção × Seleção

- Inserção:
  - Comparações: $O(n^2)$
  - Trocas: $O(n^2)$
- Seleção:
  - Comparações: $O(n^2)$

Projeto por Indução Fraca

Inserção $\times$ Seleção

- Inserção:
  - Comparações: $O(n^2)$
  - Trocas: $O(n^2)$
- Seleção:
  - Comparações: $O(n^2)$
  - Trocas: $O(n)$

Projeto por Indução Fraca

Inserção × Seleção

- Inserção:
  - Comparações: $O(n^2)$
  - Trocas: $O(n^2)$
- Seleção:
  - Comparações: $O(n^2)$
  - Trocas: $O(n)$

- Apesar dos algoritmos de ordenação por Inserção e Seleção terem a mesma complexidade assintótica em comparações, em situações onde a operação de troca é muito custosa é preferível utilizar a Seleção

Projeto por Indução Simples

Terceira Alternativa

Projeto por Indução Simples

Terceira Alternativa

- **Base**: $n = 1$. Um único elemento está ordenado

Projeto por Indução Simples

Terceira Alternativa

- **Base**: $n = 1$. Um único elemento está ordenado
- **H.I.**: Sei ordenar um conjunto de $n - 1 \geq 1$ valores

Projeto por Indução Simples

Terceira Alternativa

- **Base**: $n = 1$. Um único elemento está ordenado
- **H.I.**: Sei ordenar um conjunto de $n - 1 \geq 1$ valores
- **Passo**:

Projeto por Indução Simples

Terceira Alternativa

- **Base**: $n = 1$. Um único elemento está ordenado
- **H.I.**: Sei ordenar um conjunto de $n - 1 \geq 1$ valores
- **Passo**:
  - Seja S um conjunto de $n \geq 2$ valores, e x o maior elemento de S

Projeto por Indução Simples

Terceira Alternativa

- **Base**: $n = 1$. Um único elemento está ordenado
- **H.I.**: Sei ordenar um conjunto de $n - 1 \geq 1$ valores
- **Passo**:
  - Seja S um conjunto de $n \geq 2$ valores, e x o maior elemento de S
  - Então x certamente é o último elemento da sequência ordenada de S. Por hipótese de indução, sabemos ordenar os demais $S - x$ elementos, e assim obtemos S ordenado

Projeto por Indução Simples

Terceira Alternativa

- **Base**: $n = 1$. Um único elemento está ordenado
- **H.I.**: Sei ordenar um conjunto de $n - 1 \geq 1$ valores
- **Passo**:
  - Seja S um conjunto de $n \geq 2$ valores, e x o maior elemento de S
  - Então x certamente é o último elemento da sequência ordenada de S. Por hipótese de indução, sabemos ordenar os demais $S - x$ elementos, e assim obtemos S ordenado
- Variação do Método da Seleção

Projeto por Indução Simples

- No entanto, se implementarmos de uma forma diferente a seleção e o posicionamento do maior elemento, obteremos o algoritmo da Bolha (Bubble Sort):

Projeto por Indução Simples

- No entanto, se implementarmos de uma forma diferente a seleção e o posicionamento do maior elemento, obteremos o algoritmo da Bolha (Bubble Sort):

```
Bolha(A, n):
  Entrada: Arranjo A de n valores
  Saída: Aranjo A ordenado.

para i = n - 1 até 1 faça:
  para j = 1 até i faça:
    se A[j-1] > A[j] então
      t = A[j-1]
      A[j-1] = A[j]
      A[j] = t
```

Projeto por Indução Simples

Método da Bolha (iterativo)

- Usando a notação O, quantas comparações no arranjo são feitas no pior caso?

```
Bolha(A, n):
  para i = n - 1 até 1 faça:
    para j = 1 até i faça:
      se A[j-1] > A[j] então
        t = A[j-1]
        A[j-1] = A[j]
        A[j] = t
```

Projeto por Indução Simples

Método da Bolha (iterativo)

- Usando a notação O, quantas comparações no arranjo são feitas no pior caso?
- $O(n)$

```
Bolha(A, n):
  para i = n - 1 até 1 faça:
    para j = 1 até i faça:
      se A[j-1] > A[j] então
        t = A[j-1]
        A[j-1] = A[j]
        A[j] = t
```

Projeto por Indução Simples

Método da Bolha (iterativo)

- Usando a notação O, quantas comparações no arranjo são feitas no pior caso?
- $O(n) \times O(n)$

```
Bolha(A, n):
  para i = n - 1 até 1 faça:
    para j = 1 até i faça:
      se A[j-1] > A[j] então
        t = A[j-1]
        A[j-1] = A[j]
        A[j] = t
```

Projeto por Indução Simples

Método da Bolha (iterativo)

- Usando a notação O, quantas comparações no arranjo são feitas no pior caso?
- $O(n) \times O(n) = O(n^2)$

```
Bolha(A, n):
  para i = n - 1 até 1 faça:
    para j = 1 até i faça:
      se A[j-1] > A[j] então
        t = A[j-1]
        A[j-1] = A[j]
        A[j] = t
```

Projeto por Indução Simples

Método da Bolha (iterativo)

- Usando a notação O, quantas comparações no arranjo são feitas no melhor caso?

```
Bolha(A, n):
  para i = n - 1 até 1 faça:
    para j = 1 até i faça:
      se A[j-1] > A[j] então
        t = A[j-1]
        A[j-1] = A[j]
        A[j] = t
```

Projeto por Indução Simples

Método da Bolha (iterativo)

- Usando a notação O, quantas comparações no arranjo são feitas no melhor caso?
- $O(n)$

```
Bolha(A, n):
  para i = n - 1 até 1 faça:
    para j = 1 até i faça:
      se A[j-1] > A[j] então
        t = A[j-1]
        A[j-1] = A[j]
        A[j] = t
```

Projeto por Indução Simples

Método da Bolha (iterativo)

- Usando a notação O, quantas comparações no arranjo são feitas no melhor caso?
- $O(n) \times O(n)$

```
Bolha(A, n):
  para i = n - 1 até 1 faça:
    para j = 1 até i faça:
      se A[j-1] > A[j] então
        t = A[j-1]
        A[j-1] = A[j]
        A[j] = t
```

Projeto por Indução Simples

Método da Bolha (iterativo)

- Usando a notação O, quantas comparações no arranjo são feitas no melhor caso?
- $O(n) \times O(n) = O(n^2)$

```
Bolha(A, n):
  para i = n - 1 até 1 faça:
    para j = 1 até i faça:
      se A[j-1] > A[j] então
        t = A[j-1]
        A[j-1] = A[j]
        A[j] = t
```

Projeto por Indução Simples

Método da Bolha (iterativo)

- E quantas trocas de elementos são feitas no arranjo no pior caso?

```
Bolha(A, n):
  para i = n - 1 até 1 faça:
    para j = 1 até i faça:
      se A[j-1] > A[j] então
        t = A[j-1]
        A[j-1] = A[j]
        A[j] = t
```

Projeto por Indução Simples

Método da Bolha (iterativo)

- E quantas trocas de elementos são feitas no arranjo no pior caso?
- $O(n)$

```
Bolha(A, n):
  para i = n - 1 até 1 faça:
    para j = 1 até i faça:
      se A[j-1] > A[j] então
        t = A[j-1]
        A[j-1] = A[j]
        A[j] = t
```

Projeto por Indução Simples

Método da Bolha (iterativo)

- E quantas trocas de elementos são feitas no arranjo no pior caso?
- $O(n) \times O(n)$

```
Bolha(A, n):
  para i = n - 1 até 1 faça:
    para j = 1 até i faça:
      se A[j-1] > A[j] então
        t = A[j-1]
        A[j-1] = A[j]
        A[j] = t
```

Projeto por Indução Simples

Método da Bolha (iterativo)

- E quantas trocas de elementos são feitas no arranjo no pior caso?

```
Bolha(A, n):
  para i = n - 1 até 1 faça:
    para j = 1 até i faça:
      se A[j-1] > A[j] então
        t = A[j-1]
        A[j-1] = A[j]
        A[j] = t
```

- $O(n) \times O(n) = O(n^2)$

Projeto por Indução Fraca

Método da Bolha: iterativo × recursivo

```
void bubble(int A[], int n){
  int i, j, t;
  for (i = n - 1; i>0; i--)
    for(j = 1; j<=i; j++)
      if (A[j-1] > A[j]){
        t = A[j-1];
        A[j-1] = A[j];
        A[j] = t;
      }
}
```

```
void bubble(int A[], int n){
  int j, t;
  if (n>1){
    for(j = 1; j < n; j++)
      if (A[j-1] > A[j]){
        t = A[j-1];
        A[j-1] = A[j];
        A[j] = t;
      }
    bubble(A, n-1);
  }
}
```

Projeto por Indução Fraca

Inserção × Seleção × Bolha

	Comparações			Trocas		
Algoritmo	melhor	médio	pior	melhor	médio	pior
Inserção						
Seleção						
Bolha						

[a] É possível adaptar a implementação para não fazer nenhuma troca.

Projeto por Indução Fraca

Inserção × Seleção × Bolha

	Comparações			Trocas		
Algoritmo	melhor	médio	pior	melhor	médio	pior
Inserção			$O(n^2)$			
Seleção			$O(n^2)$			
Bolha			$O(n^2)$			

[a] É possível adaptar a implementação para não fazer nenhuma troca.

Projeto por Indução Fraca

Inserção × Seleção × Bolha

	Comparações			Trocas		
Algoritmo	melhor	médio	pior	melhor	médio	pior
Inserção			$O(n^2)$	$O(n)^a$		
Seleção			$O(n^2)$	$O(n)^a$		
Bolha			$O(n^2)$	$O(1)$		

[a] É possível adaptar a implementação para não fazer nenhuma troca.

Projeto por Indução Fraca

Inserção × Seleção × Bolha

	Comparações			Trocas		
Algoritmo	melhor	médio	pior	melhor	médio	pior
Inserção	$O(n)$		$O(n^2)$	$O(n)^a$		
Seleção	$O(n^2)$		$O(n^2)$	$O(n)^a$		
Bolha	$O(n^2)$		$O(n^2)$	$O(1)$		

[a] É possível adaptar a implementação para não fazer nenhuma troca.

Projeto por Indução Fraca

Inserção × Seleção × Bolha

	Comparações			Trocas		
Algoritmo	melhor	médio	pior	melhor	médio	pior
Inserção	$O(n)$		$O(n^2)$	$O(n)^a$		$O(n^2)$
Seleção	$O(n^2)$		$O(n^2)$	$O(n)^a$		$O(n)$
Bolha	$O(n^2)$		$O(n^2)$	$O(1)$		$O(n^2)$

[a] É possível adaptar a implementação para não fazer nenhuma troca.

Projeto por Indução Fraca

Inserção × Seleção × Bolha

	Comparações			Trocas		
Algoritmo	melhor	médio	pior	melhor	médio	pior
Inserção	$O(n)$	$O(n^2)$	$O(n^2)$	$O(n)^a$	$O(n^2)$	$O(n^2)$
Seleção	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n)^a$	$O(n)$	$O(n)$
Bolha	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	$O(n^2)$	$O(n^2)$

[a] É possível adaptar a implementação para não fazer nenhuma troca.

Projeto por Indução Fraca

Inserção × Seleção × Bolha

	Comparações			Trocas		
Algoritmo	melhor	médio	pior	melhor	médio	pior
Inserção	$O(n)$	$O(n^2)$	$O(n^2)$	$O(n)^a$	$O(n^2)$	$O(n^2)$
Seleção	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n)^a$	$O(n)$	$O(n)$
Bolha	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	$O(n^2)$	$O(n^2)$

[a] É possível adaptar a implementação para não fazer nenhuma troca.

Projeto por Indução Fraca

Inserção × Seleção × Bolha

	Comparações			Trocas		
Algoritmo	melhor	médio	pior	melhor	médio	pior
Inserção	$O(n)$	$O(n^2)$	$O(n^2)$	$O(n)^a$	$O(n^2)$	$O(n^2)$
Seleção	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n)^a$	$O(n)$	$O(n)$
Bolha	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	$O(n^2)$	$O(n^2)$

[a] É possível adaptar a implementação para não fazer nenhuma troca.

Vídeo comparando diversos algoritmos de ordenação (incluindo os três vistos até o momento):
`https://www.youtube.com/watch?v=ZZuD6iUe3Pc`

Referências

- Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford. Introduction to Algorithms. 2a ed. MIT Press, 2001.

- Material baseado em slides dos professores Cid de Souza, Cândida da Silva e Delano Beder

Aula 16 – Ordenação por Seleção e Bolha

Norton T. Roman & Luciano A. Digiampietri
digiampietri@usp.br
@digiampietri

2023