

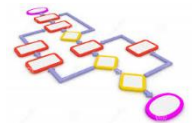


Unidade 9 – Análise de Complexidade de Algoritmos Básicos de Ordenação



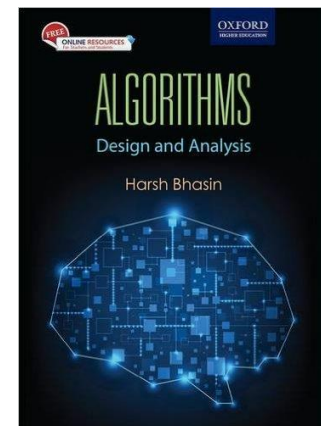
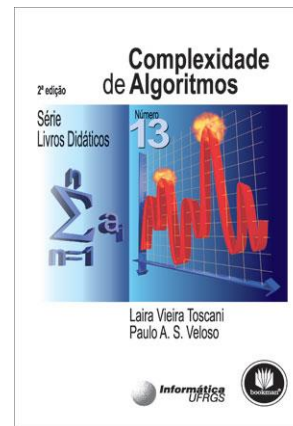
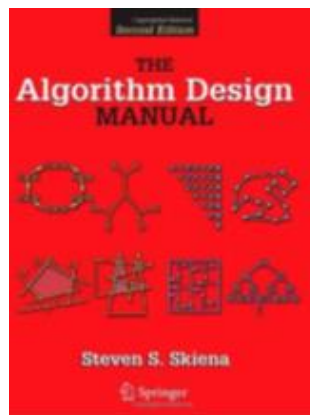
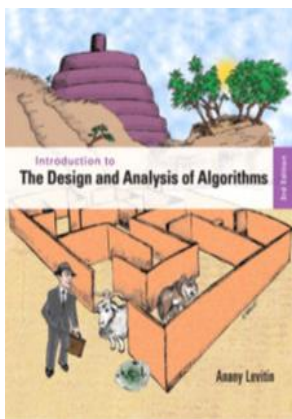
Prof. Aparecido V. de Freitas
Doutor em Engenharia
da Computação pela EPUVSP
aparecidovfreitas@gmail.com





Bibliografia

- Algorithm Design and Applications – Michael T. Goodrich, Roberto Tamassia, Wiley, 2015
- Introduction to the Design and Analysis of Algorithms – Anany Levitin, Pearson, 2012
- The Algorithm Design Manual – Steven S. Skiena, Springer, 2008
- Complexidade de Algoritmos – Série Livros Didáticos – UFRGS
- Algorithms – Design and Analysis – Harsh Bhasin – Oxford University Press - 2015



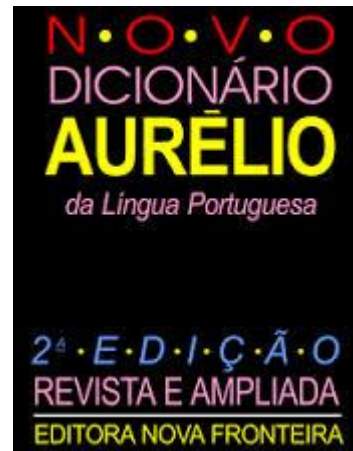


Como seria encontrar um nome em
lista telefônica desordenada ?





Ou encontrar um nome em um dicionário que não estivesse ordenado ?





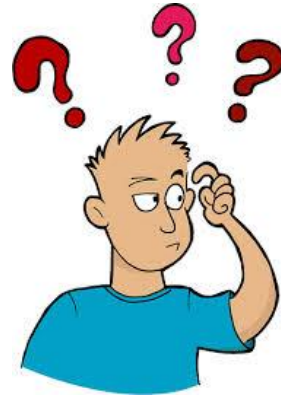
Ordenação

- A conveniência de se usar dados ordenados é inquestionável na Computação.
- Mesmo o computador é ineficiente ao buscar dados em agrupamentos não ordenados.
- Felizmente, existem bons algoritmos de ordenação.





Como se mede a eficiência de um
algoritmo de ordenação ?





Algoritmos de Ordenação

- A eficiência em geral é medida pelo número de comparações e pelo número de movimentações de dados.





Problema de Ordenação

Permutar (rearranjar) os elementos de um array $a[0..n-1]$ de tal modo que eles fiquem em ordem crescente, ou seja, de tal forma que se tenha:

$$a[0] \leq a[1] \leq \dots \leq a[n-1]$$



Algoritmo Bubble-Sort

- ⊕ Uma **bolha** nada mais é que um **par de números ordenados**.
Exemplo (2,7) e (7,4).
- ⊕ A primeira destas bolhas está ordenada crescentemente mas a segunda não.
- ⊕ Toda vez que se encontra uma **bolha** desordenada, deve-se invertê-la, transformando (7,4) em (4,7).
- ⊕ As **bolhas** são assim chamadas pois *sobem pela sequência* da mesma forma como bolhas de gás na água.
- ⊕ Complexidade: $O(n^2)$, com algumas variações dependendo da forma dos dados de entrada.





Sequência a ser ordenada: 4, 2, 12, 7, 4, 1

(4	2)	12	7	4	1
2	(4	12)	7	4	1
2	4	(12	7)	4	1
2	4	7	(12	4)	1
2	4	7	4	(12	1)
2	4	7	4	1	12

(2	4)	7	4	1	12
2	(4	7)	4	1	12
2	4	(7	4)	1	12
2	4	4	(7	1)	12
2	4	4	1	(7	12)
2	4	4	1	7	12



Algoritmo Bubble-Sort

```
void Bolha (int tam, int vet[]) {  
    int passo, i, aux;  
  
    // Passos de Ordenação  
    for (passo=1 ; passo < tam ; passo++) {  
  
        // Passos de Troca  
        for (i=1 ; i < tam; i++) {  
            if (vet[i-1] > vet[i]) {  
                aux          = vet[i-1];  
                vet[i-1] = vet[i];  
                vet[i]    = aux;  
            }  
        }  
    }  
}
```



Algoritmo Bubble-Sort Implementação

```
package maua;  
  
import java.util.Arrays;  
  
public class Bubble_01 {  
  
    public static void main (String [] args ) {  
  
        int[] lista = { 9, 7 , 4, 3, 1 } ;  
  
        lista = bubble(lista);  
  
        System.out.println (Arrays.toString(lista) ) ;  
    }  
}
```



Algoritmo Bubble-Sort Implementação

```
public static int[] bubble(int[] lista) {  
  
    int aux;  
    int n = lista.length;  
  
    for (int i = 1; i < n ; i++ ) {  
  
        for (int j = 1; j < n ; j ++)  
  
            if (lista[j-1] > lista[j] ) {  
                aux = lista[j-1];  
                lista[j-1] = lista[j];  
                lista[j] = aux;  
            }  
  
    }  
  
    return lista;  
  
}
```

}





Bubble Sort com Traçe



Algoritmo Bubble-Sort Implementação

```

public static int[] bubble (int[] lista) {

    int aux;

    for(int i = 1 ; i < lista.length ; i++) {

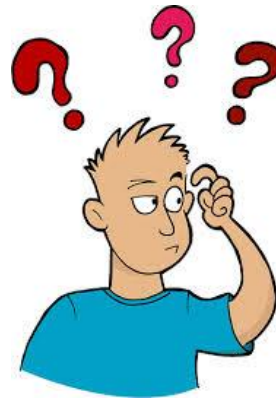
        System.out.println ("\n*****PASSO: " + i + "      *****");
        System.out.println ("\n Lista antes de aplicar o PASSO: " +
            Arrays.toString(lista) );
        for (int j=1; j < lista.length;j++)
            if (lista[j-1] > lista[j] ) {
                System.out.println("Troca de " + lista[j-1] +
                    " com " + lista[j]);
                aux = lista[j-1];
                lista[j-1]=lista[j];
                lista[j] = aux;
            }
            else System.out.println("Não trocou " + lista[j-1] +
                " com " + lista[j]);
        }
    }
    return lista;
}

```





Quantas comparações e trocas são feitas no algoritmo Bubble Sort ?

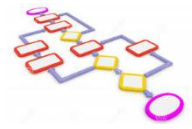




Análise de Complexidade

- ⊕ O número de comparações é o mesmo em cada caso (melhor, pior e médio) e igual ao número total de iterações do **for** interno.
- ⊕ O número de trocas, no pior caso, ocorre quando o array estiver em ordem inversa.
- ⊕ Ordem de complexidade: **$O(n^2)$**





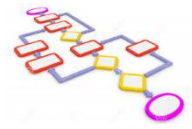
Exercício

- Ordenar a sequência 4, 2, 12, 1 pelo método Bolha.
- Mostrar todas as TROCAS EFETUADAS NA SEQUÊNCIA EM QUE ACONTECEM.
- Apresente também, passo a passo, a evolução do conteúdo do array.

***** PASSO: 1 *****

- Lista antes de aplicar o PASSO: [4, 2, 12, 1]
- Troca de 4 com 2
- Não trocou 4 com 12
- Troca de 12 com 1
- Lista após o PASSO 1: [2,4,1,12]





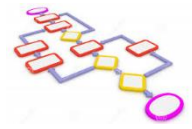
Exercício

- Ordenar a sequência 4, 2, 12, 1 pelo método Bolha.
- Mostrar todas as TROCAS EFETUADAS NA SEQUÊNCIA EM QUE ACONTECEM.
- Apresente também, passo a passo, a evolução do conteúdo do array.

***** PASSO: 2 *****

- Lista antes de aplicar o PASSO: [2, 4, 1, 12]
- Não trocou 2 com 4
- Troca de 4 com 1
- Não trocou 4 com 12
- Lista após o PASSO 2: [2,1,4,12]





Exercício

- Ordenar a sequência 4, 2, 12, 1 pelo método Bolha.
- Mostrar todas as TROCAS EFETUADAS NA SEQUÊNCIA EM QUE ACONTECEM.
- Apresente também, passo a passo, a evolução do conteúdo do array.

***** PASSO: 3 *****

Lista antes de aplicar o PASSO: [2, 1, 4, 12]

Troca de 2 com 1

Não trocou 2 com 4

Não trocou 4 com 12

Lista após o PASSO 3: [1,2,4,12]





Exercício

- Alterar o algoritmo Bubble_Sort para classificar o array em ordem DECRESCENTE.





Algoritmo Inserção Direta

- Ⓢ Usado por exemplo, para colocar em ordem um baralho de cartas
- Ⓢ Mais rápido que o método Bubble-Sort.
- Ⓢ Ordena um array utilizando um sub-array ordenado localizado em seu início, e a cada novo passo, acrescenta a este sub-array mais um elemento, até atingir o último elemento do array fazendo com que ele se torne ordenado.



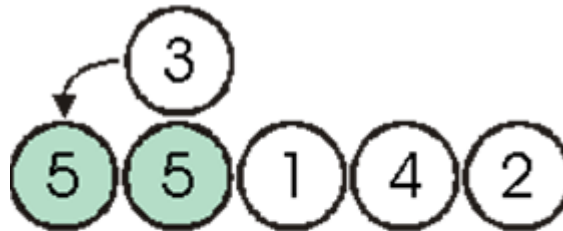
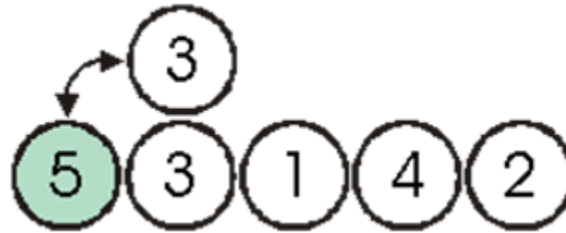


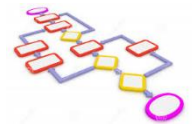
Método de Ordenação Inserção Direta



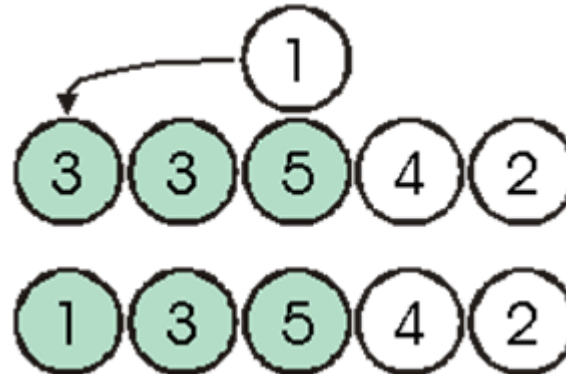
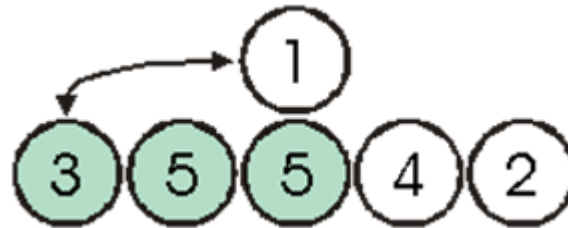
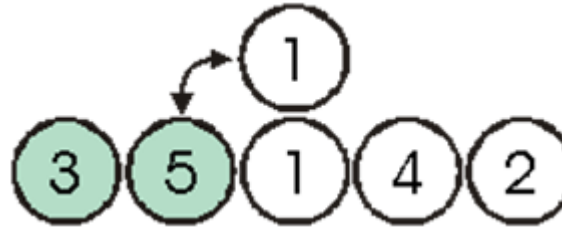


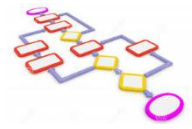
Método de Ordenação Inserção Direta



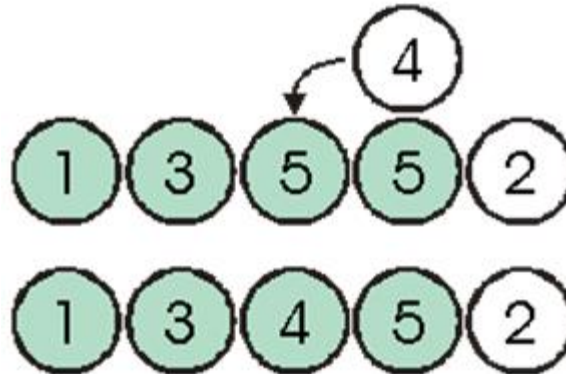
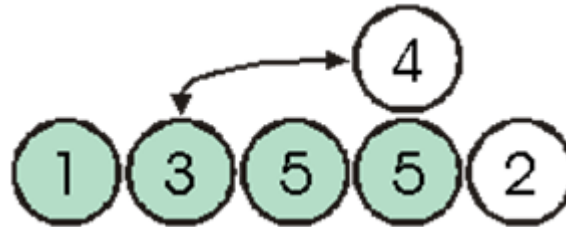
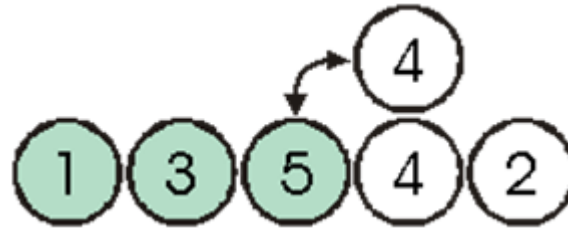


Método de Ordenação Inserção Direta



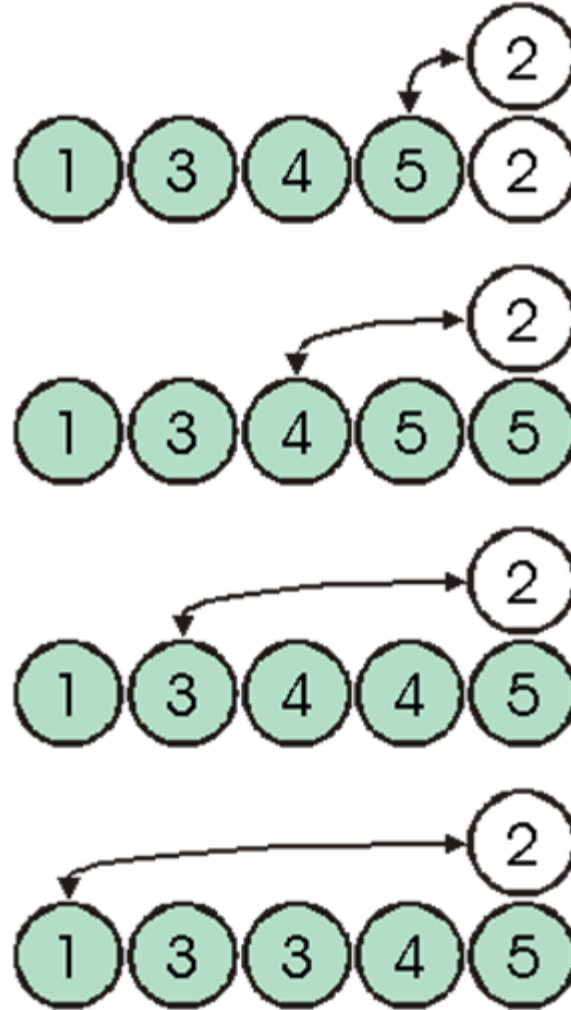


Método de Ordenação Inserção Direta



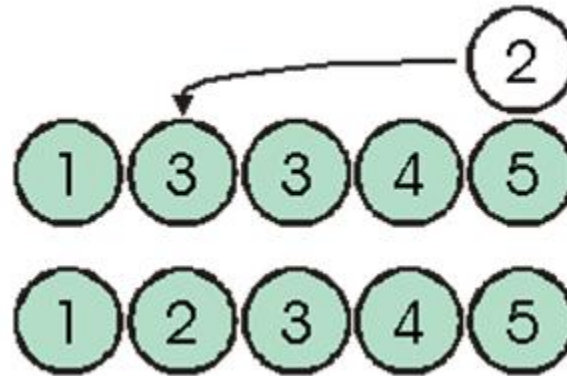


Método de Ordenação Inserção Direta





Método de Ordenação Inserção Direta



Array Ordenado



Algoritmo – Inserção Direta

```
package maua;  
  
import java.util.Arrays;  
  
public class SortInsertion {  
  
    public static void main (String[] args ) {  
  
        int[] lista = { 2,1,8,5,0} ;  
  
        lista = sortInsertion(lista);  
  
        System.out.println(Arrays.toString(lista));  
  
    }  
}
```



Algoritmo – Inserção Direta

```
public static int[] sortInsertion(int[] lista)  {  
  
    int i, j, aux;  
  
    for (i = 1; i < lista.length; i++)  {  
  
        aux = lista[i];  
  
        j = i;  
  
        while (j > 0 && lista[j-1] > aux) {  
  
            lista[j] = lista[j-1];  
  
            j--;  
        }  
  
        lista[j] = aux;  
    }  
  
    return lista;  
}
```





Inserção Direta com Traçe





```
package maua;

import java.util.Arrays;

public class Insertion {

    public static void main (String [] args ) {

        int[] lista = { 4, 2, 12, 1 } ;

        insertionSort(lista);

        System.out.println ("Lista Ordenada: " +

                            Arrays.toString(lista) ) ;

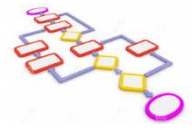
    }
```





```
public static void insertionSort(int[] lista)  {  
  
    for (int i = 1; i < lista.length; i++) {  
  
        System.out.println("\n" +  
                            "          ***** PASSO: " + i + " *****");  
  
        System.out.println("Lista antes de aplicar o PASSO: " +  
                            Arrays.toString(lista));  
  
        int a = lista[i]; // guardo o elemento a ser inserido
```





```
// passos para encontrar o elemento deve ser inserido

for (int j = i - 1; j >= 0 && lista[j] > a; j--) {

    lista[j + 1] = lista[j]; // move ate encontrar posicao correta

    System.out.println("Movimentações dentro do passo: " +

        Arrays.toString(lista));

    lista[j] = a; // insere na posicao correta

}
System.out.println("Lista dentro do PASSO: " +

    Arrays.toString(lista) + "\n");

}
}
}
```



Exercício

- Ordenar a sequência **4, 2, 12, 1** pelo método **Inserção Direta**.
- **Mostrar todas as MOVIMENTAÇÕES EFETUADAS NA SEQUÊNCIA EM QUE ACONTECEM.**
- Apresente também, passo a passo, a evolução do conteúdo do array.

***** PASSO: 1 *****

Lista antes de aplicar o PASSO: [4, 2, 12, 1]

Movimentações dentro do PASSO: [4, 4, 12, 1]

Lista dentro do PASSO: [2, 4, 12, 1]





Exercício

- Ordenar a sequência **4, 2, 12, 1** pelo método **Inserção Direta**.
- **Mostrar todas as** MOVIMENTAÇÕES EFETUADAS NA SEQUÊNCIA EM QUE ACONTECEM.
- Apresente também, passo a passo, a evolução do conteúdo do array.

***** PASSO: 2 *****

Lista antes de aplicar o PASSO: [2, 4, 12, 1]

Lista dentro do PASSO: [2, 4, 12, 1]



Exercício

- Ordenar a sequência **4, 2, 12, 1** pelo método **Inserção Direta**.
- **Mostrar todas as MOVIMENTAÇÕES EFETUADAS NA SEQUÊNCIA EM QUE ACONTECEM.**
- Apresente também, passo a passo, a evolução do conteúdo do array.

***** PASSO: 3 *****

Lista antes de aplicar o PASSO: [2, 4, 12, 1]

Movimentações dentro do PASSO: [2, 4, 12, 12]

Movimentações dentro do PASSO: [2, 4, 4, 12]

Movimentações dentro do PASSO: [2, 2, 4, 12]

Lista dentro do PASSO: [1, 2, 4, 12]

Lista ordenada: [1, 2, 4, 12]





Análise de Complexidade

- ⊕ Uma vantagem do algoritmo por inserção é que ele ordena o vetor somente quando realmente é necessário.
- ⊕ O melhor caso é quando os dados já estão em ordem. O pior caso é quando os dados estão em ordem inversa.
- ⊕ Ordem de complexidade: **$O(n^2)$**





Algoritmo por Seleção

- ⌚ Encontre o menor elemento no array;
- ⌚ Troque-o com o elemento da primeira posição;
- ⌚ Encontre o segundo menor elemento no array;
- ⌚ Troque-o com o elemento da segunda posição;
- ⌚ Continue até que o array esteja ordenado;





Algoritmo por Seleção – Pseudocódigo

```
Selectionsort(array[], n)
```

```
    for (i=0; i < n-1 ; i++)
```

```
        selecione o menor elemento  
        entre array[i], ..., array[n-1];
```

```
        troque-o com array[i];
```



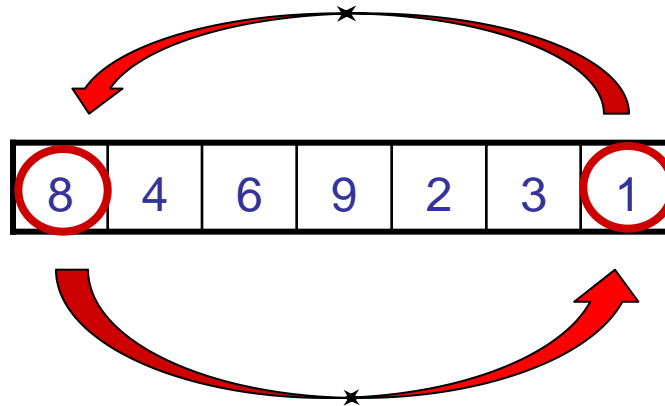


Algoritmo por Seleção



8	4	6	9	2	3	1
---	---	---	---	---	---	---

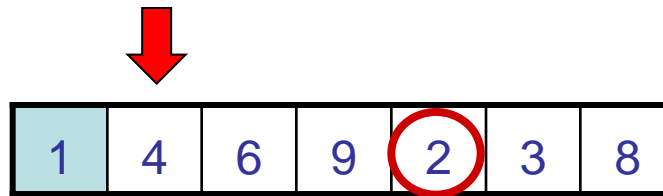
1



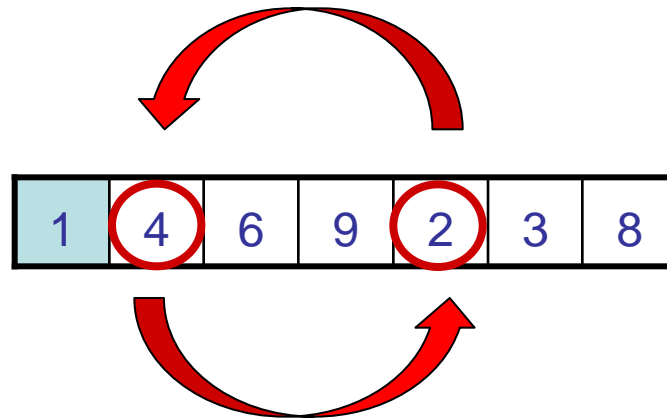
1	4	6	9	2	3	8
---	---	---	---	---	---	---



Algoritmo por Seleção

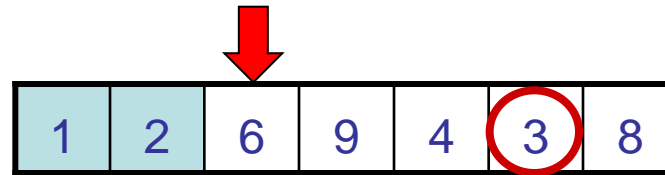


2

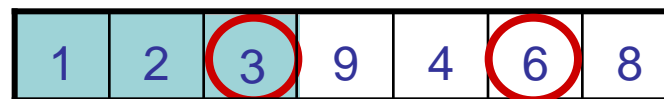
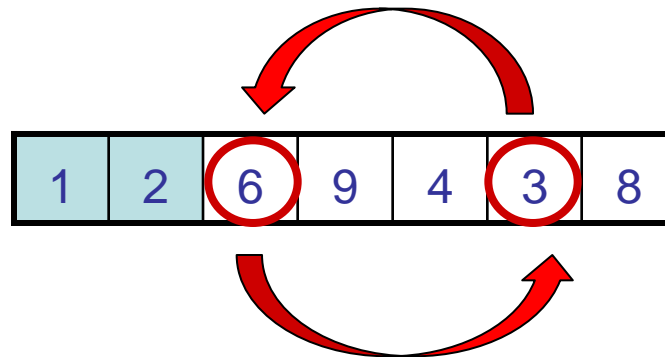




Algoritmo por Seleção

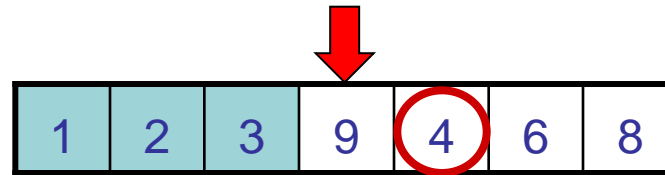


3

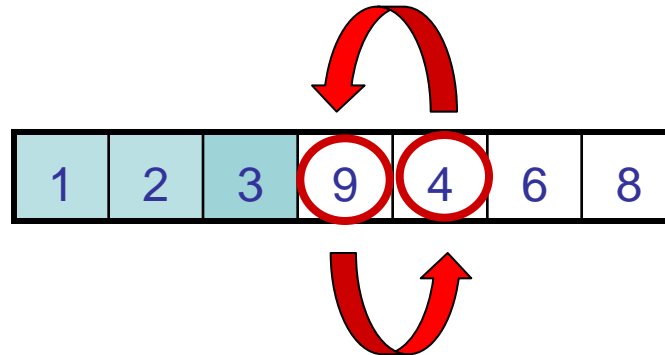




Algoritmo por Seleção

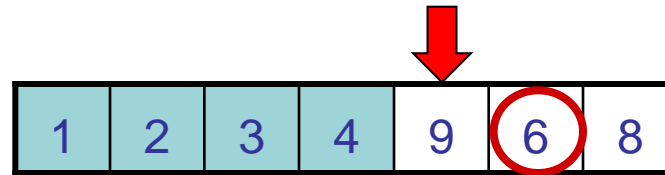


4

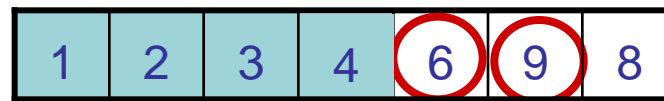
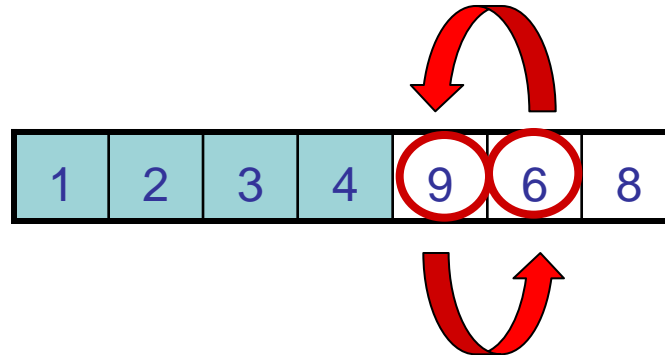




Algoritmo por Seleção



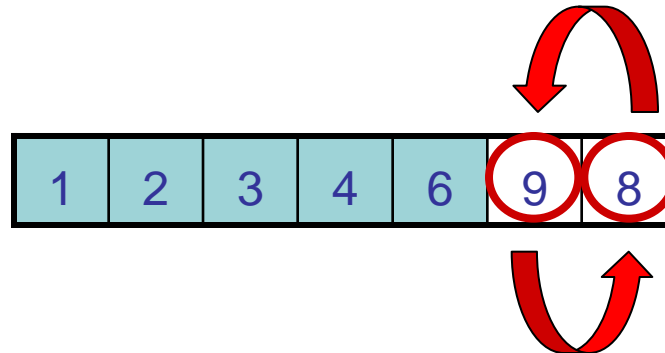
5



Algoritmo por Seleção



6





Algoritmo por Seleção – Implementação

```
Selectionsort(array[], n) {  
    for (i=0; i < n-1 ; i++) {  
  
        selecione o menor elemento  
        entre array[i],...,array[n-1];  
  
        troque-o com array[i];  
    }  
}
```





```
package maua;

import java.util.Arrays;


public class Selection {
    public static void main(String[] args) {

        int[] lista = new int[] { 5, 2, 1, 9, 7 } ;

        lista = selection_sort(lista);

        System.out.println("\n\nLista ordenada: " +
            Arrays.toString(lista) );
    }
}
```





```
public static int[] selection_sort(int[] lista) {  
  
    int menor, indiceMenor;  
  
    for (int i = 0 ; i < lista.length - 1 ; i++) {  
  
        menor = lista[i];  
  
        indiceMenor = i ;  
  
        for (int j = i+1 ; j < lista.length ; j++) {  
  
            if (lista[j] < menor) {  
  
                menor = lista[j];  
                indiceMenor = j;  
  
            }  
        }  
        lista[indiceMenor] = lista[i];  
  
        lista[i]=menor;  
    }  
    return lista;  
}
```





Selection com Trace

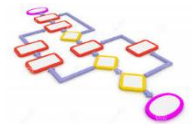




```
public static int[] selection_sort(int[] lista) {  
  
    //armazena o menor valor e o índice do menor valor  
    int menor, indiceMenor;  
  
    for (int i = 0 ; i < lista.length - 1 ; i++) {  
  
        System.out.println ("\n\n\n*****PASSO: " +  
                             (i+1) + " *****");  
        System.out.println ("Lista antes de aplicar o PASSO: " +  
                             Arrays.toString(lista) );  
  
        menor = lista[i];  
        indiceMenor = i ;  
  
        // compara com os outros valores do array  

```





```
for (int j = i+1 ; j < lista.length ; j++) {  
    if (lista[j] < menor) {  
        menor = lista[j];  
        indiceMenor = j;  
    }  
}  
  
//swap  
  
lista[indiceMenor] = lista[i];  
lista[i]=menor;  
System.out.println("Movimentações dentro do PASSO: " +  
    Arrays.toString(lista) );  
  
}  
return lista;  
}
```





Análise de Complexidade

- ⊕ O laço mais externo executa $n-1$ vezes.
- ⊕ O laço mais interno executa de $i + 1$ até $n-1$.
- ⊕ Ordem de complexidade: **$O(n^2)$**





Exercício

- Ordenar a sequência **5, 2, 1, 9, 7** pelo método **SelectionSort**.
- **Mostrar todas as MOVIMENTAÇÕES EFETUADAS NA SEQUÊNCIA EM QUE ACONTECEM.**
- Apresente também, passo a passo, a evolução do conteúdo do array.

*****PASSO: 1 *****

Lista antes de aplicar o PASSO: [5, 2, 1, 9, 7]

Movimentações dentro do PASSO: [1, 2, 5, 9, 7]





Exercício

- Ordenar a sequência **5, 2, 1, 9, 7** pelo método **SelectionSort**.
- **Mostrar todas as MOVIMENTAÇÕES EFETUADAS NA SEQUÊNCIA EM QUE ACONTECEM.**
- Apresente também, passo a passo, a evolução do conteúdo do array.

*****PASSO: 2 *****

Lista antes de aplicar o PASSO: [1, 2, 5, 9, 7]

Movimentações dentro do PASSO: [1, 2, 5, 9, 7]





Exercício

- Ordenar a sequência **5, 2, 1, 9, 7** pelo método **SelectionSort**.
- **Mostrar todas as MOVIMENTAÇÕES EFETUADAS NA SEQUÊNCIA EM QUE ACONTECEM.**
- Apresente também, passo a passo, a evolução do conteúdo do array.

*****PASSO: 3 *****

Lista antes de aplicar o PASSO: [1, 2, 5, 9, 7]

Movimentações dentro do PASSO: [1, 2, 5, 9, 7]





Exercício

- Ordenar a sequência **5, 2, 1, 9, 7** pelo método **SelectionSort**.
- **Mostrar todas as MOVIMENTAÇÕES EFETUADAS NA SEQUÊNCIA EM QUE ACONTECEM.**
- Apresente também, passo a passo, a evolução do conteúdo do array.

*****PASSO: 4 *****

Lista antes de aplicar o PASSO: [1, 2, 5, 9, 7]

Movimentações dentro do PASSO: [1, 2, 5, 7, 9]

