PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA POLITÉCNICA E DE ARTES
ESTRUTURA DE DADOS ORIENTADA A OBJETOS
ADS1232
PROF. MSC. ANIBAL SANTOS JUKEMURA

PROGRAMAÇÃO ORIENTADA A OBJETOS [JAVA]

Agenda

- Classe Random
- Ordenação
- Exemplos
- Exercícios

Agenda

- Classe Random: gerar números aleatórios
  - Exemplo:

```
import java.util.Random;

public class Random2 {

        public static void main(String[] args) {

                //instância um objeto da classe Random usando o construtor básico
                Random gerador = new Random();

                //imprime sequência de 10 números inteiros aleatórios entre 0 e 25
            for (int i = 0; i < 10; i++) {
                System.out.println(gerador.nextInt(26));
                 }
        }
}
```

Façamos então, um exemplo em sala com ArrayList

PUC GOIÁS

O que é Ordenação de uma estrutura de dados?

- Ordenar corresponde ao processo de rearranjar um conjunto de objetos em uma ordem específica.

- Objetivo da ordenação: – facilitar a recuperação posterior de elementos do conjunto ordenado.

- Exemplos:
  - Listas telefônicas
  - Dicionários
  - Índices de livros
  - Tabelas e arquivos

PUC GOIÁS

Discutiremos dois algoritmos dos vários existentes

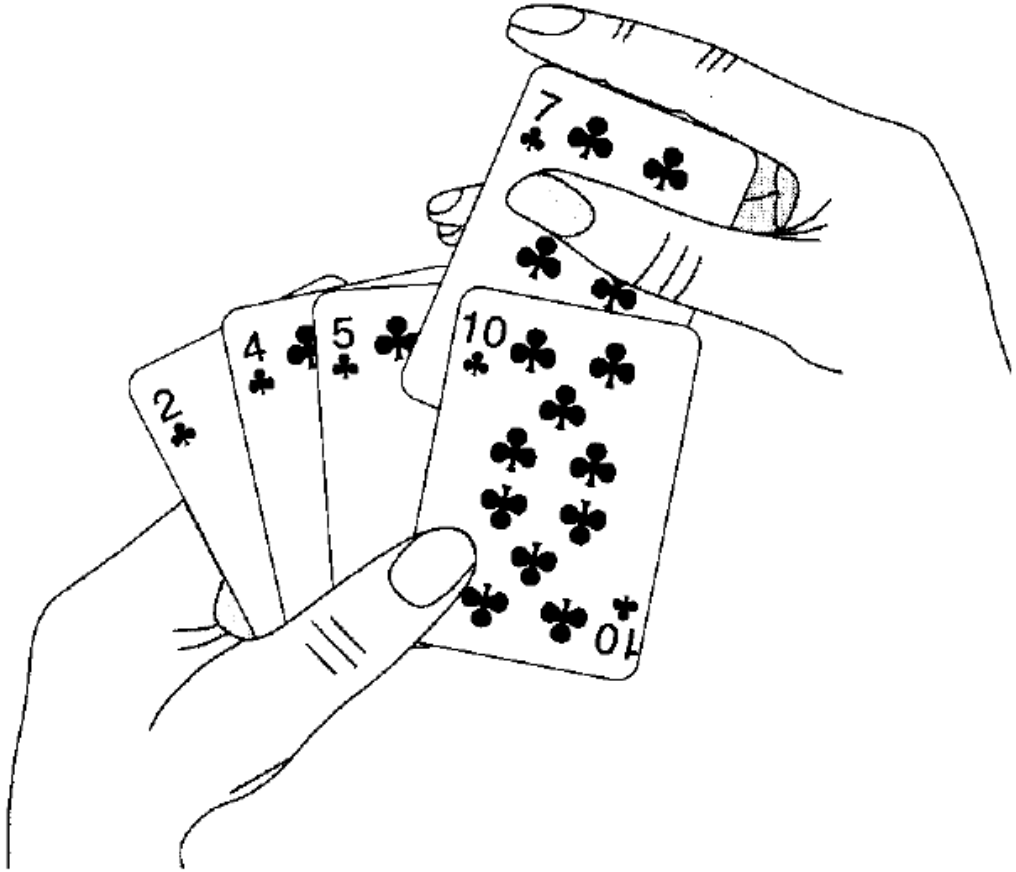
- Selection Sort
- Insertion Sort

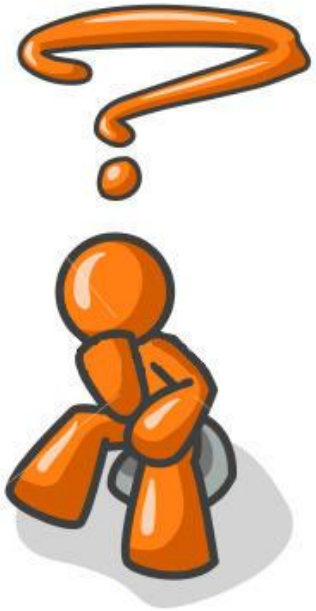
Mas antes. Como você faria?

12	7	10	5	4	15	9	8

PUC GOIÁS

Pense em um jogo de cartas, por exemplo:

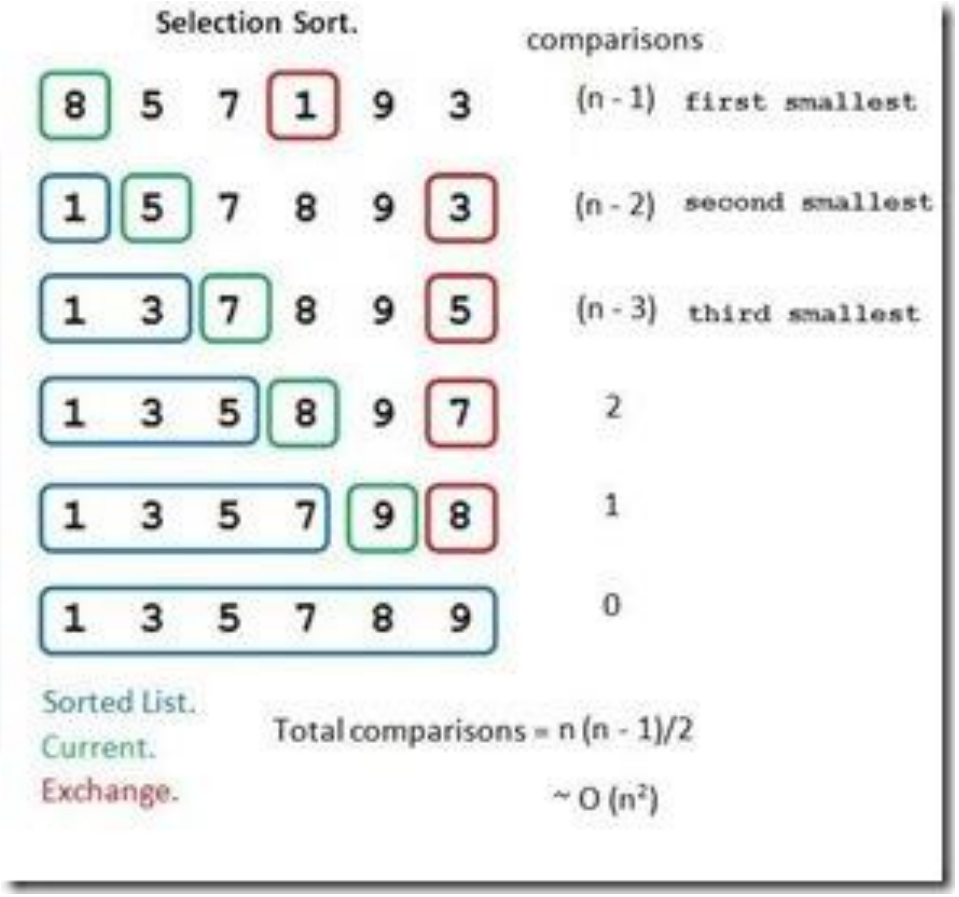




PUC GOIÁS

Selection Sort:

A ideia é sempre procurar o menor elemento da lista e inseri-lo no início do lista. Procuramos o menor valor da lista e colocamos ele na menor posição atual da lista. E assim vamos indo até termos toda lista ordenada.
Partindo sempre a partir do último elemento reordenado (a partir do *i*), o programa procura o menor elemento na lista e o substitui pelo elemento *i* atual.



PUC GOIÁS

Selection Sort: Algoritmo Genérico

```
1. para i ← 1 até tamanho-1, faça
2.  minimo ← i
3.  para j ← i+1 até tamanho, faça
4.      se vetor[j] < vetor[minimo], então
5.          minimo ← j
6.      fim-se
7.  fim-para
8.  temp ← vetor[i]
9.  vetor[i] ← vetor[minimo]
10. vetor[minimo] ← temp
11. fim-para
```

Vamos implementá-lo?

PUC GOIÁS

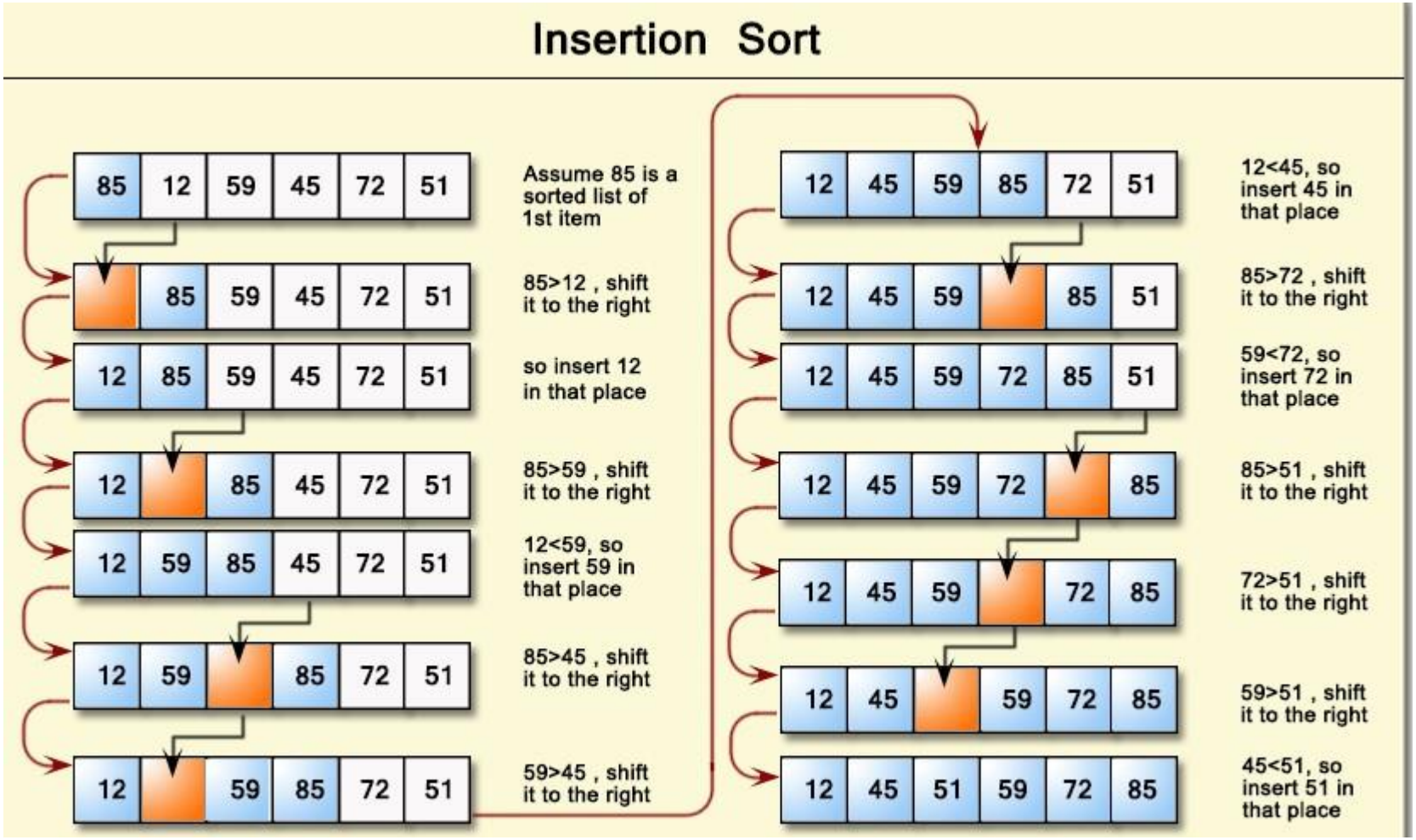
Insertion Sort:

A ideia:

- Considera que o primeiro elemento está ordenado (ou seja, na posição correta).
- A partir do segundo elemento, insere os demais elementos na posição correta entre aqueles já ordenados.
- O elemento é inserido na posição adequada movendo-se todos os elementos maiores para direita (posição seguinte do vetor).

PUC GOIÁS

Insertion Sort:



PUC GOIÁS

Insertion Sort: Algoritmo Genérico

```
ORDENA-POR-INSERÇÃO(A, n)
1  para j ← 1 até n faça
2      chave ← A[j]
3      ▷ Insere A[j] no subvetor ordenado A[1..j − 1]
4      i ← j − 1
5      enquanto i ≥ 0 e A[i] > chave faça
6          A[i + 1] ← A[i]
7          i ← i − 1
8      A[i + 1] ← chave
```

Vamos implementá-lo?

PUC GOIÁS

Complicado ? Use a classe Collections

```
import java.util.Collections;



.
.
.
.
.
Collections.sort(lista);
```

Vamos implementá-lo?


```java
import java.util.Random;
import java.util.ArrayList;
import java.util.Collections;

public class RamdomClassExemplo {

    public static void selectionSort(ArrayList<Integer> list)
    {
        int i, j, temp, minimo;
        for (i = 0; i < list.size(); i++)
        {
            minimo = i;
            for (j = i + 1; j <list.size(); j++)
            {
              if (list.get(j) < list.get(minimo))
                minimo = j;
            }
            temp = list.get(i);
            list.set(i, list.get(minimo));
            list.set(minimo, temp);
        }
    }
```

PUC GOIÁS

```
24      public static void insertionSort(ArrayList<Integer> list)
25      {
26          int i = 0, j, chave = 0;
27          for (j = 1; j < list.size(); j++)
28          {
29              chave = list.get(j);
30              i = j - 1;
31              while (i >= 0 && list.get(i) > chave)
32              {
33                  list.set(i + 1, list.get(i));
34                  i--;
35              }
36              list.set(i + 1, chave);
37          }
38      }
```

PUC GOIÁS

```
40      public static void main(String[] args) {
41          int tamanho = 20;
42          Random fator = new Random();
43          ArrayList<Integer> lista = new ArrayList<Integer>(tamanho);
44
45          for (int i = 0; i < tamanho; i++)
46          {
47              lista.add(fator.nextInt(101));
48          }
49          System.out.println(lista);
50          selectionSort(lista);
51          insertionSort(lista);
52          //Collections.sort(lista);
53          System.out.println(lista);
54      }
55  }
```

PUC GOIÁS

Exercícios:

Exercício 01 – Crie uma lista com a classe ArrayList de 10 números inteiros. Após isso, liste apenas os números pares em uma nova lista chamada “pares”. Após isso, liste apenas os números ímpares em uma nova lista chamada “impares”. Agora ordene ambas as listas e imprime-as novamente.

Listas

Referência Bibliográfica Principal

• DEVMEDIA. Disponível em https://www.devmedia.com.br . Acessado em Fevereiro de 2018.

• CORMEN, Thomas H. et al. Algoritmos: teoria e prática. Capítulo 02. 3. ed. Rio de Janeiro: Campus, 2012.