INSTITUTO POLITÉCNICO DE LEIRIA
ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO
Algoritmos e Estruturas de Dados
2º ano, 1º semestre – 2018/2019
Ficha Prática 3

Algoritmos de Ordenação

Os exercícios seguintes devem ser desenvolvidos nos packages:

- **pt.ipleiria.estg.dei.aed.ordenacao.algoritmos**
- **pt.ipleiria.estg.dei.aed.ordenacao.utilizacao**
- **pt.ipleiria.estg.dei.aed.modelo.contactos**
- **pt.ipleiria.estg.dei.aed.modelo.contactos.io**
- **pt.ipleiria.estg.dei.aed.modelo.contactos.comparadores**

1. InsertionSort

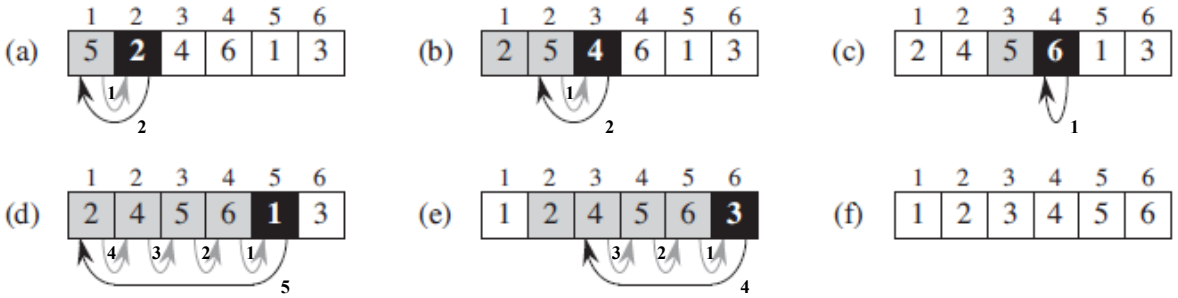

Figura 1. InsertionSort
(adaptado de http://www.java2novice.com/java-sorting-algorithms/insertion-sort/)

O algoritmo de ordenação InsertionSort (Figura 1) consiste em

1. Dividir uma sequência em duas partes:
   - Subsequência ordenada (parte esquerda, constituída inicialmente apenas pelo primeiro elemento);
   - A subsequência por ordenar (parte direita).

2. Em cada iteração, selecionar o primeiro elemento da subsequência por ordenar e percorrer os elementos da subsequência ordenada, a partir do último elemento e em direção ao primeiro. Enquanto esses elementos tiverem uma ordem superior ao elemento selecionado, coloca-os na posição seguinte (setas cinzentas na Figura 1). Quando o algoritmo encontra um elemento com ordem não superior ao elemento selecionado, encontra assim espaço para colocar o elemento selecionado na posição seguinte (setas pretas na Figura 1).

3. O algoritmo termina quando a subsequência por ordenar for vazia.

Com base na especificação acima, resolva os exercícios seguintes:

a) Recorrendo à classe **AlgoritmoOrdenacao**, implemente o algoritmo de ordenação **InsertionSort**;

b) Execute a função para sequências de 100, 1.000 e 10.000 valores inteiros aleatórios entre -100 e 100, -1.000 e 1.000 e -10.000 e 10.000 respetivamente;

c) Apresente uma janela de gráficos que demonstre a evolução do número de comparações e do tempo de execução para sequências de tamanho 5 a 40 com incrementos de 5.

2. Ordenação de Contactos

a) Implemente as entidades necessárias para a representação de contactos, no package **pt.ipleiria.estg.dei.aed.modelo.contactos**, com a seguinte informação:

- Primeiro nome (String)
- Último nome (String)
- Número de telefone (long)
- Morada (String)
- Data de nascimento
  - Dia (int)
  - Mês (int)
  - Ano (int)

b) Recorrendo à interface **Comparacao**, implemente dois critérios, no package **pt.ipleiria.estg.dei.aed.modelo.contactos.comparadores**, que permitam ordenar os contactos por:

- número de telefone, de forma descendente;
- último nome, de forma ascendente e, em caso de empate, seguidamente por data de nascimento de forma descendente.

c) Crie dois vetores com os contactos apresentados na Tabela 1 e ordene-os com o algoritmo **InsertionSort** de acordo com os critérios implementados na alínea anterior:

Primeiro Nome	Último Nome	Número de Telefone	Morada	Data de Nascimento
Ana	Silva	950000000	Rua de Leiria	01/10/1990
Ana	Rita	990000000	Travessa 25 de Abril	15/06/2000
Hugo	Santos	971234567	Avenida 1º de Maio	18/03/1994
Teresa	Silva	950000001	Rua de Leiria	02/10/1990
Eça	Queiroz	100000000	Praça do Almada	25/11/1845

Tabela 1. Contactos a inserir no projeto

d) Visualize a execução do exercício da alínea c) em *jGRASP*.

3. Comparação de Algoritmos

Para a realização deste exercício deve descarregar, a partir do Moodle, os seguintes ficheiros:

- **dados_contactos_50000.csv**
- **dados_contactos_300002.csv**
- **dados_contactos_50000_ordenados.csv**
- **dados_contactos_50000_quase_ordenados.csv**

a) Implemente uma classe **ContactosIO** no package **pt.ipleiria.estg.dei.aed.modelo.contactos.io** que permita ler uma lista de contactos, a partir de um ficheiro *csv* no formato:

- primeira linha indica o **número de contactos**
- cada uma das restantes linhas indica um contacto no formato: **Primeiro nome**, **Último nome**, **Número de telefone**, **Morada**, **Data de nascimento**;

b) Apresente uma janela de gráficos que permita comparar os algoritmos de ordenação **SelectionSort**, **BubbleSortOtimizado**, **QuickSort** e **InsertionSort**, demonstrando a evolução do número de comparações, do número de trocas e do tempo de execução para sequências de Contactos de tamanho 10 a 50 com incrementos de 10, utilizando como fonte de dados o ficheiro **dados_contactos_50000.csv**;

c) Compare as estatísticas obtidas na execução dos algoritmos de ordenação **SelectionSort**, **BubbleSortOtimizado**, **QuickSort** e **InsertionSort**, quando a sequência de contactos a ordenar está muito desordenada, quase ordenada e completamente ordenada. Como fonte de dados, utilize os ficheiros **dados_contactos_50000.csv**, **dados_contactos_50000_quase_ordenados.csv** e **dados_contactos_50000_ordenados.csv**, ordenando os contactos através do campo último nome, de forma ascendente e, em caso de empate, através do campo data de nascimento de forma descendente (critério definido na alínea b) do 2º exercício).