CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
Unidade Contagem

Linguagens e Técnicas de Programação II

TRABALHO PRÁTICO 1

Ana Beatriz Costa Viana
Ingrid Yara Alves dos Santos
Maria Luiza Terra Dutra

Contagem
2022

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
Unidade Contagem

TRABALHO PRÁTICO 1

Trabalho realizado para a disciplina de Linguagens e Técnicas de Programação II da terceira série do curso técnico integrado de Informática do Centro Federal de Educação Tecnológica de Minas Gerais.

Orientador: Alisson Rodrigo dos Santos

Contagem
2022

RESUMO

O trabalho em questão foi desenvolvido para a matéria de Linguagens e Técnicas de Programação II com objetivo demonstrar, analisar e comparar a quantidades de movimentações que são feitas nos algoritmos de ordenação: *InsertionSort, SelectionSort* e *QuickSort* com a utilização de arquivos de texto.

Assim, as alunas Ana Beatriz, Ingrid Yara e Maria Luiza da turma de Informática 3 concluíram com sucesso o desafio proposto, explorando o paradigma de programação da Orientação a Objetos, os fundamentos da linguagem de programação Java e a utilização de IDE's avançadas como o Visual Studio Code.

Palavras-chave: InsertionSort. SelectionSort. QuickSort. Orientação a Objetos. Java.

ABSTRACT

The present work was developed for the subject of Linguagens e Técnicas de Programação II with the objective of demonstrating, analyzing and comparing the amounts of movements that are made in the sorting algorithms: *InsertionSort, SelectionSort* and *QuickSort* with the use of text files.

Thus, students Ana Beatriz, Ingrid Yara and Maria Luiza from the Informatics 3 class successfully completed the proposed challenge, exploring the Object Oriented programming paradigm, the fundamentals of the Java programming language and the use of advanced IDE's such as Visual Studio Code.

Keywords: InsertionSort. SelectionSort. QuickSort. Object orientation. Java.

SUMÁRIO

1. INTRODUÇÃO

Este relatório tem como finalidade apresentar o desenvolvimento e resultados obtidos da pesquisa e codificação para o '**Trabalho Prático 1 - Ordenação**', realizado pelas discentes **Ana Beatriz Costa Viana, Ingrid Yara Alves dos Santos e Maria Luiza Terra Dutra** de T1, orientado pelo professor Alisson Rodrigo e com apoio dos colegas de INFO3.

Utilizando a linguagem **Java** e o editor e compilador **Visual Studio Code**, foram realizadas a análise e comparação da quantidade de **movimentações** de valores obtidas pelos algoritmos de ordenação ***InsertionSort, SelectionSort*** **e** ***QuickSort,*** testando diferentes arquivos de texto previamente fornecidos.

2. DESENVOLVIMENTO

Para a realizarmos o trabalho de forma satisfatória foi necessária a definição de objetivos internos para o cumprimento de todos os requisitos, sendo: a organização do programa - em que nós definimos os arquivos, suas classes e características; a pesquisa sobre os métodos propostos ao nosso grupo e, por fim, a codificação em si. Vale ressaltar que na elaboração do produto final, estes objetivos foram revisitados durante todo o desenvolvimento para que pudéssemos concluir o trabalho com êxito.

Ao decorrer das pesquisas identificamos que para o funcionamento do nosso código precisaríamos da bibliotecas do *java.io*, sendo elas: **FileReader, BufferedReader**, FileNotFoundException e IOException; também utilizamos as bibliotecas do *java.util* **ArrayList** e Scanner.

Devido a necessidade de flexibilidade e fácil entendimento do programa como um todo, organizamos os códigos em quatro (4) arquivos, sendo o primeiro o **Main** - que define qual arquivo será lido, chama a função que lê o arquivo e as funções que ordenam o arquivo. Ao final, ele apresenta os arquivos ordenados no terminal juntamente com o contador de movimentações.

Para a **leitura e armazenamento dos dados**, criamos o arquivo/**classe Dados**, que lê os arquivos .txt e passa os nomes para um ArrayList através da função '**leArquivo (função do tipo ArrayList<String>)**'. Esta, recebe como parâmetro a **String 'nomeDoArquivo'** para que o FileReader e o BufferedReader funcionem de maneira correta; utilizamos um bloco **Try/Catch** para realizar a leitura das linhas e adicioná-las ao Arraylist. Ao final, a função retorna um **ArrayList de strings** com todos os nomes presentes no arquivo. Da mesma forma criamos outra função denominada **Escreve (função do tipo ArrayList<String>, String FileName, String nome)**, que tem como objetivo escrever as informações ordenadas dentro de um arquivo .csv, para efetuar essa ação ele recebe como parâmetro o ArrayList, o arquivo de texto que vai receber esses dados e uma String para identificar qual é o método de ordenação do arquivo, e finalizando essa parte do código, o **Try/Catch** foi utilizado desta vez com o objetivo escrever no arquivo.

Para o **contador de movimentações** criamos a **classe Contador** que possui três (3) funções: '**Incrementador (função do tipo void)**' - incrementa a variável privada 'contador' da classe toda vez que é chamada; **'getContador (função do tipo int)'** - retorna a variável 'contador'; e **'imprimeContador (função do tipo void)'** - que imprime a variável 'contador' e altera seu valor para 0.

Por fim, criamos a **classe Ordenados que é filha da classe Contador**, responsável por ordenar o ArrayList utilizando os diferentes algoritmos de ordenação. Suas funções serão melhor esclarecidas no tópico a seguir.

2.2 - Métodos de Ordenação

2.2.1 - Selection Sort

O método de ordenação Selection Sort ordena os elementos percorrendo um determinado vetor ou lista, então, seleciona o menor valor do conjunto e o coloca na primeira posição do vetor e assim segue sucessivamente até que reste o elemento de maior valor.

Para implementarmos o algoritmo nos baseamos no tópico sobre Selection Sort no site w3Schools[?] e nos códigos e explicações encontrados no tópico 'Java ArrayList String Selection Sort' do site Stack Overflow[?]. Com a utilização dessas referências pudemos gerar o código que atendesse às especificações do problema em questão.

O método de ordenação Selection Sort utiliza apenas a função '**selectionSort (função pública do tipo ArrayList<String>)**' que recebe como parâmetro o ArrayList de String '**listaDeNomes**', realiza a sua ordenação e ao final retorna este mesmo ArrayList.

2.2.2. - Insertion Sort

O método de ordenação Insertion Sort é caracterizado por ser simples e com uma implementação simples. Seu funcionamento consiste em construir um array ordenado, um item por vez. Os elementos são comparados entre si em sequência, depois são dispostos em uma ordem específica. Essa ordenação trabalha inserindo um elemento em uma determinada posição, daí o nome de ordenação por Insertion Sort.

Para implementação desse método foi necessário a utilização de duas funções públicas (insertSort e insertSorted) em onde a função insertSort chama a função insertSorted para ordenar o ArrayList (que é o conteúdo do arquivo de texto) recebido por parâmetro e incrementa a variável index que controla até quando ordenar. E ao final é retornado o ArrayList ordenado. Para a realização desse procedimento nós utilizamos como base os códigos do site W3schools e na página da web no apêndice [7].

2.2.3 - Quick Sort

O método quicksort classifica seus elementos começando com o elemento chamado pivô e, em seguida, organiza a lista de modo que todos os elementos antes do pivô sejam menores que o pivô e todos os elementos depois deles sejam maiores que o pivô. Ao final da ordenação, o pivô estará em sua posição final. Os dois grupos desordenados recursivamente sofreram o mesmo processo até que a lista esteja ordenada. Esse processo é repetido até que se tenha a lista ordenada.

O método de ordenação Quick Sort utiliza três funções: duas privadas (quickSort e separar) e uma pública (QuickSort). Essas funções recebem como parâmetro o ArrayList para ordenar e variáveis para controlar as posições e ao final é retornado o ArrayList ordenado.

3. CONCLUSÃO

Ao fazer um comparativo com os valores obtidos pelos contadores implementados, vemos que o método Insertion Sort é o menos eficiente devido a sua alta quantidade de movimentações e, tendo como base o número de comparações, o método mais eficiente é o Selection Sort. Porém, para quando se deseja ordenar uma grande quantidade de valores o ideal é usar os métodos Selection Sort e Quick Sort, pelo menor número de movimentações.

3.1 - Tabelas comparativas

Tabela 1 - Lista Crescente

Algoritmo	100 elementos	1.000 elementos	10.000 elementos
SelectionSort	0	0	0
InsertionSort	0	1	18
QuickSort	0	1	18

Fonte: Elaborado pelas autoras com base no terminal do programa desenvolvido, 2022.

Tabela 2 - Lista Decrescente

Algoritmo	100 elementos	1.000 elementos	10.000 elementos
SelectionSort	50	501	5012
InsertionSort	4950	499500	49995000
QuickSort	50	500	5006

Fonte: Elaborado pelas autoras com base no terminal do programa desenvolvido, 2022.

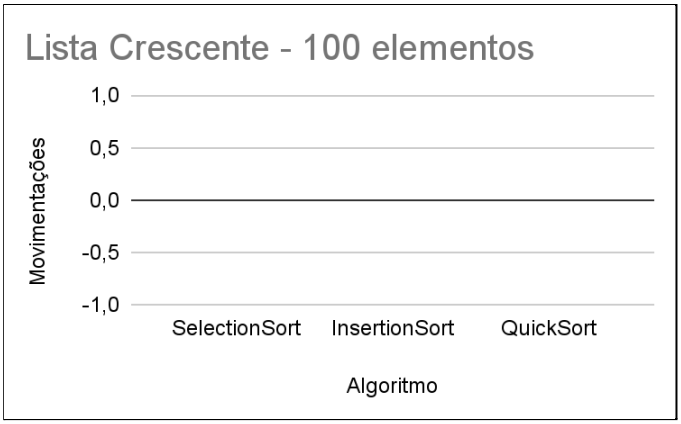
Tabela 3 - Lista Aleatória

Algoritmo	100 elementos	1.000 elementos	10.000 elementos
SelectionSort	92	992	9991
InsertionSort	2318	245111	25062018
QuickSort	132	2201	29945

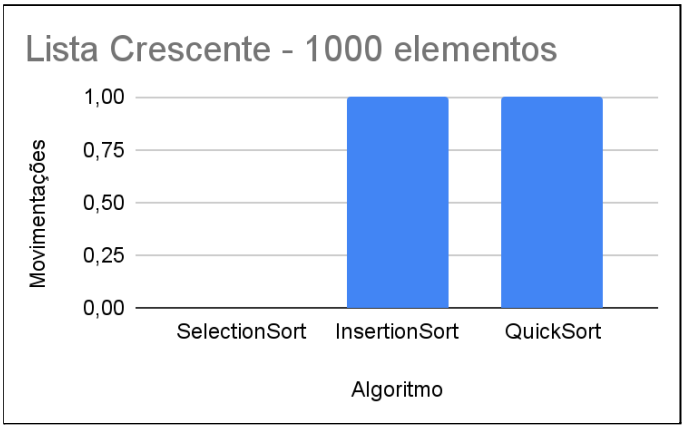
Fonte: Elaborado pelas autoras com base no terminal do programa desenvolvido, 2022.

3.2 - Gráficos de barra

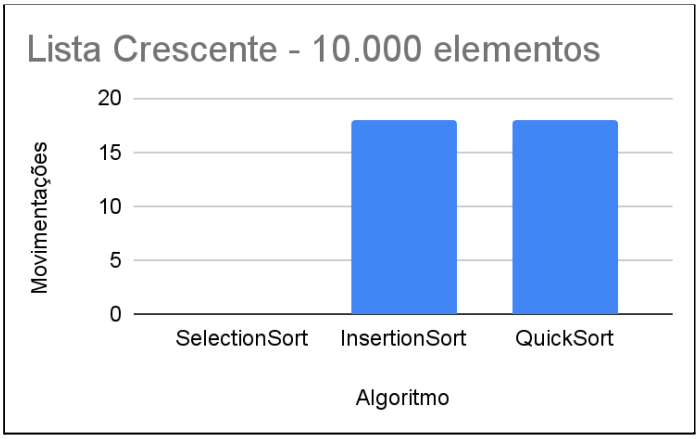
Os gráficos a seguir foram gerados com base nas tabelas.



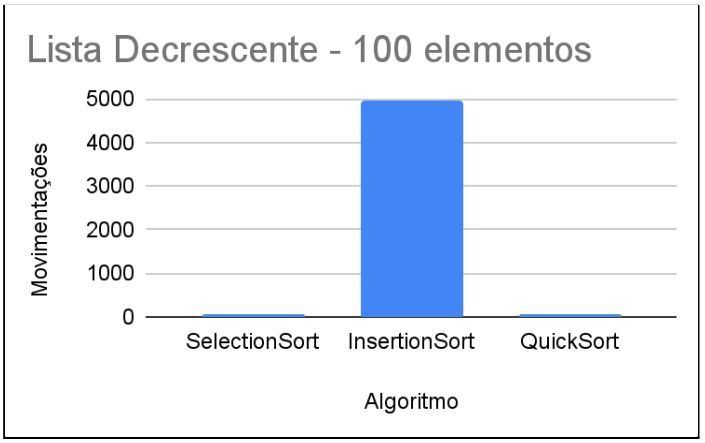
Fonte: Elaborado pelas autoras, 2022.



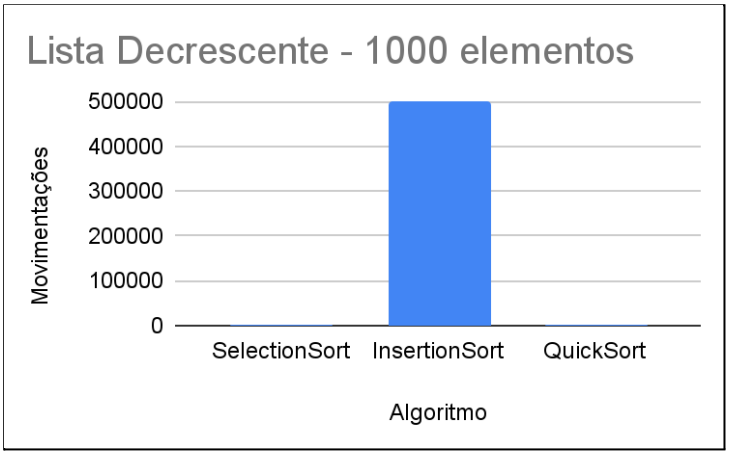
Fonte: Elaborado pelas autoras, 2022.



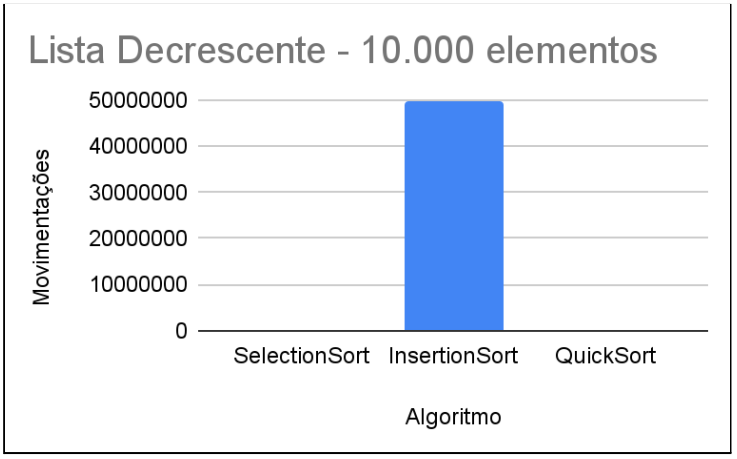
Fonte: Elaborado pelas autoras, 2022.



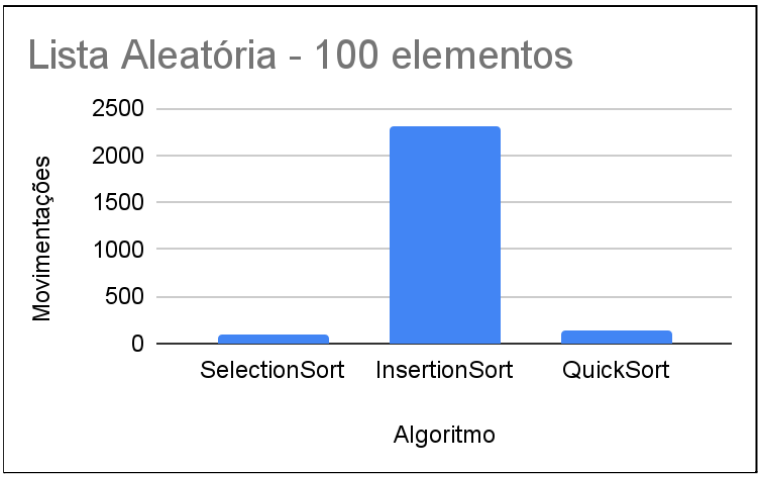
Fonte: Elaborado pelas autoras, 2022.



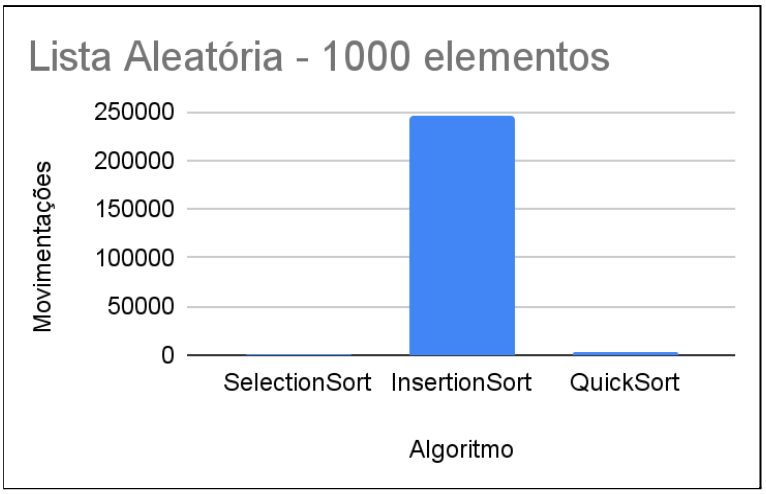
Fonte: Elaborado pelas autoras, 2022.



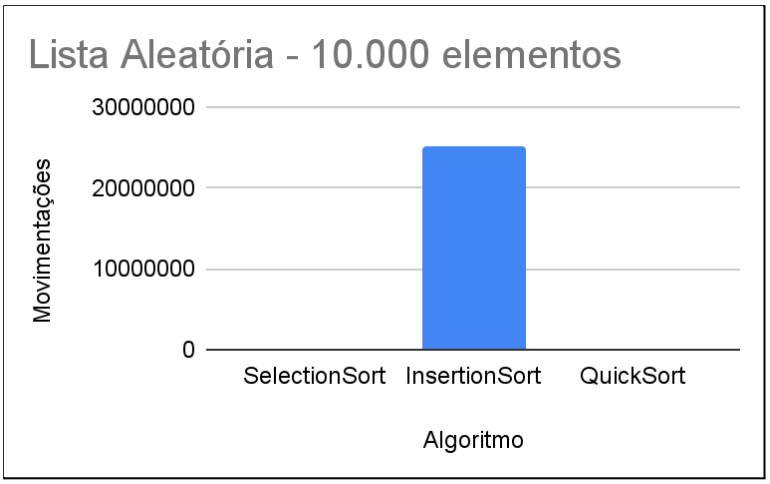
Fonte: Elaborado pelas autoras, 2022.



Fonte: Elaborado pelas autoras, 2022.



Fonte: Elaborado pelas autoras, 2022.



Fonte: Elaborado pelas autoras, 2022.

4. REFERÊNCIAS

JAVA SELECTION SORT ALGORITHM EXAMPLE. W3Schools, 2022. Disponível em: https://www.w3schools.blog/java-selection-sort-algorithm-example. Acesso em 10 de abril de 2022.

JAVA ARRAYLIST STRING SELECTION SORT. Stack OverFlow, 2017. Disponível em: https://stackoverflow.com/questions/47219202/java-arraylist-string-selection-sort. Acesso em: 16 de abril de 2022.

JAVA INSERTION SORT ALGORITHM. W3Schools, 2022. Disponível em: https://www.w3schools.blog/java-insertion-sort-algorithm-example. Acesso em: 14 may. 2022.

INSERTION SORT. Geeks for geeks, 2022. Disponível em: https://www.geeksforgeeks.org/insertion-sort/. Acesso em: 14 may. 2022.

INSERTION SORT (WITH CODE IN PYTHON/C++/JAVA). Programiz, 2022. Disponível em: https://www.programiz.com/dsa/insertion-sort. Acesso em: 14 may. 2022.

INSERTION SORT ALGORITHM. Insertion sort algorithm. InterviewBit, 2022. Disponível em: https://www.interviewbit.com/tutorial/insertion-sort-algorithm/. Acesso em: 14 may. 2022.

Apêndice [7] (URL sem título). Disponível em: https://www.ccs.neu.edu/home/vkp/csu213-sp05/Lectures/InsertionSort.java. Acesso em: 14 may. 2022.

DE ALMEIDA HONORATO, B. Algoritmos de Ordenação: Análise e Comparação. Devmidia, 2022. Disponível em: https://www.devmedia.com.br/algoritmos-de-ordenacao-analise-e-comparacao/28261. Acesso em: 14 may. 2022.