INSTITUTO FEDERAL
Minas Gerais

IFMG - Campus São João Evangelista
Sistemas de Informação
Turma: SI 2021-1

Professor: Eduardo Trindade
eduardo.trindade@ifmg.edu.br
Algoritmo e Estrutura de Dados II

Trabalho Prático I - 30 pontos

Observações

- O trabalho deve ser executado e entregue **individualmente**.
- É recomendado discutir os problemas e estratégias de solução com seus colegas.
- Tente solucionar os problemas você mesmo, pois solucionar problemas e desenvolver o raciocínio lógico é componente fundamental neste curso.
- Dúvidas devem ser postadas no fórum para discussão (Trabalho Prático I) no Moodle. Sua dúvida pode ser a mesma de um colega.
- A entrega deverá ser feita **exclusivamente** pelo Moodle.
- **Atenção:** Entregar em um único arquivo .zip, (ou .rar ou .7z ou .tar.gz) contendo todo o seu trabalho. O Arquivo deve conter uma pasta com o projeto (implementação) e outra com o relatório (e demais documentos, se for o caso). O nome do arquivo .zip deve ser TP1AEDSIISeuNome.zip.
- Caso não esteja utilizando a linguagem C/C++, favor especificar.
- A data limite de entrega encontra-se no Moodle.

Objetivos

- Fixar conceitos sobre manipulação de arquivos.
- Fixar conceitos sobre algoritmos de ordenação por comparações.
- Realizar a implementação destes algoritmos utilizando vetores.
- Registrar o tempo de execução dos algoritmos (processo de ordenação e comparações), considerando diferentes tamanhos de entradas de dados.
- Analisar a complexidade dos algoritmos implementados.
- Realizar experimentação com vários tipos e tamanhos de entradas de dados, tecendo análises sobre os resultados.

Descrição

Neste trabalho você deverá implementar todos os algoritmos de ordenação apresentados em aula:

- Bubble Sort
- Shell Sort
- Selection Sort
- Insertion Sort
- Quick Sort
- Merge Sort

Seu objetivo é fazer um estudo comparativo dos métodos medindo o tempo de execução e o número de comparações realizadas para vários tamanhos de vetores de entrada. Faça um programa que calcule os tempos e as comparações e salve esses dados em um arquivo de texto.

Instâncias

Uma instância é um caso particular de um problema, com dados específicos. Cada conjunto de dados de um problema define uma instância do problema. Por exemplo, uma instância para o problema de ordenação é uma sequência de dados com elementos para serem ordenados.

Neste trabalho, os algoritmos implementados serão utilizados para ordenar uma sequência de dados contida em uma das instâncias fornecidas.

Essas instâncias possuem os seguintes tamanhos: 1.000; 10.000 e 100.000 elementos, sendo que para cada um destes tamanhos temos instâncias com as seguintes características: aleatórias; quase ordenadas; inversamente ordenadas; e ordenadas. Assim, serão fornecidas 12 instâncias conforme descrito abaixo:

- **ListaAleatoria-1000.txt** - Arquivo contendo uma sequência aleatória de 1.000 elementos do tipo inteiro.
- **ListaQuaseOrdenada-1000.txt** - Arquivo contendo uma sequência de 1.000 elementos do tipo inteiro, com apenas alguns elementos fora de ordem.
- **ListaInversamenteOrdenada-1000.txt** - Arquivo contendo uma sequência inversamente ordenada de 1.000 elementos do tipo inteiro.
- **ListaOrdenada-1000.txt** - Arquivo contendo uma sequência ordenada de 1.000 elemenntos do tipo inteiro.
- **ListaAleatoria-10000.txt** - Arquivo contendo uma sequência aleatória de 10.000 elementos do tipo inteiro.
- **ListaQuaseOrdenada-10000.txt** - Arquivo contendo uma sequência de 10.000 elementos do tipo inteiro, com apenas alguns elementos fora de ordem.
- **ListaInversamenteOrdenada-10000.txt** - Arquivo contendo uma sequência inversamente ordenada de 10.000 elementos do tipo inteiro.
- **ListaOrdenada-10000.txt** - Arquivo contendo uma sequência ordenada de 10.000 elemenntos do tipo inteiro.
- **ListaAleatoria-100000.txt** - Arquivo contendo uma sequência aleatória de 100.000 elementos do tipo inteiro.

- **ListaQuaseOrdenada-100000.txt** - Arquivo contendo uma sequência de 100.000 elementos do tipo inteiro, com apenas alguns elementos fora de ordem.
- **ListaInversamenteOrdenada-100000.txt** - Arquivo contendo uma sequência inversamente ordenada de 100.000 elementos do tipo inteiro.
- **ListaOrdenada-100000.txt** - Arquivo contendo uma sequência ordenada de 100.000 elemenntos do tipo inteiro.

Implementação

A implementação deve seguir as seguintes regras:

(1) O programa poderá ser implementado em qualquer linguagem de programação de sua preferência, desde que devidamente comentado e indentado.

(2) Seu programa deverá exibir um menu inicial que pergunte ao usuário qual método será utilizado para ordenação.

(3) Após a escolha do método, apresente um novo menu com uma lista de instâncias para o usuário. Deve-se escolher aqui uma das doze instâncias fornecidas para ordenação. Cada instância corresponde a um arquivo instância fornecido e um tamanho N de dados.

(4) Definido o método e a instância, direcione o usuário para a visualização da ordenação do vetor.

(5) Mostre o número de comparações realizadas e o tempo de execução do algoritmo (considere o processo de ordenação e comparações para cálculo). Dica: para cálculo do tempo de execução, sugere-se o uso da função clock() da biblioteca time.h.

(6) Armazene em um arquivo .txt os dados que julgar necessários para a etapa seguinte de Análise (ex: comparações, tempo, método, tamanho da instância, etc.).

Seja criativo para elaborar seu programa e utilize melhorias se achar necessário.

Análise

Pelo menos duas análises deverão ser realizadas:

1 **Complexidade**: definição da ordem de complexidade do algoritmo (O(n)), considerando que a instrução mais relevante é a comparação.

2 **Tempo de execução**: registrar o tempo que cada algoritmo demorou para resolver cada instância do problema. Para obter uma medida estatisticamente mais confiável, sugere-se a medição de um mesmo método para uma determinada instância várias vezes (10x) e calcule o tempo médio dividindo o tempo total pelo número de vezes que o algoritmo foi executado.

   Utilize o arquivo de texto gerado em um programa como o Excel, Matlab ou Octave para gerar gráficos dos seus resultados. As Figuras a seguir mostram exemplos de resultados obtidos para diversos métodos de ordenação.

Figura 1: Métodos de Ordenação - Resultados Matlab

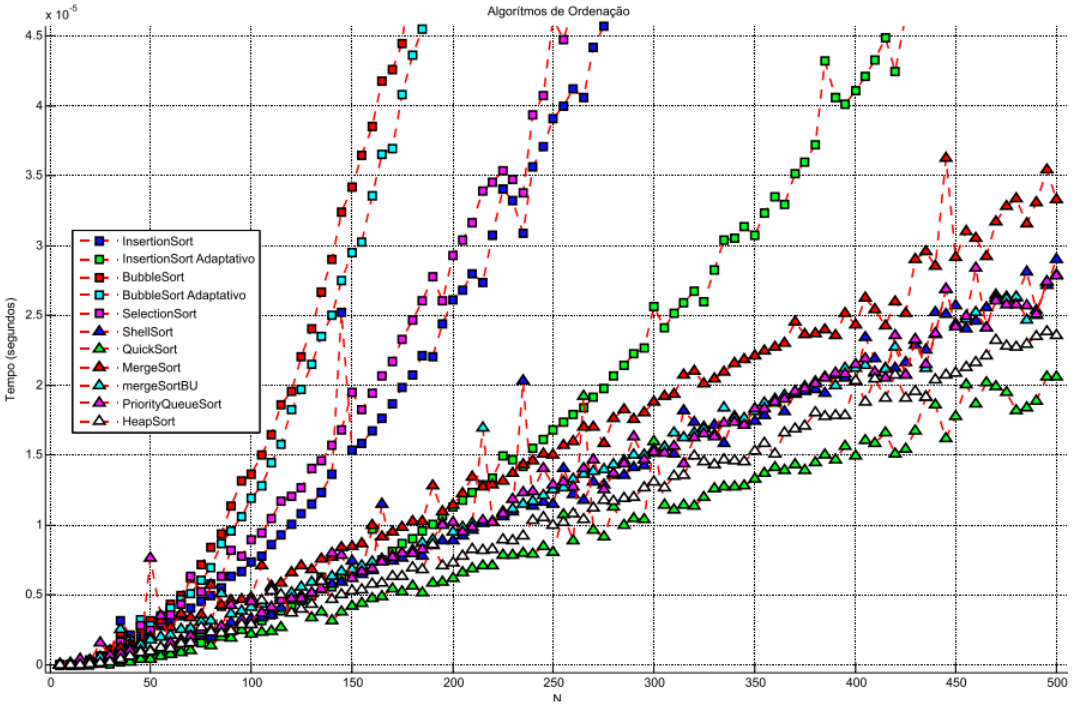

Figura 2: Resultados expostos em Tabela e Gráfico Excel

Lista Inversamente Ordenada

Número de Elementos	Número de Comparações	
	SelectionSort	QuickSort
1.000	499.500	2.032
10.000	49.995.000	22.714
100.000	4.999.950.000	246.604

Entrega

O que deverá ser entregue:

1 **Código fonte bem indentado e comentado**: Código fonte contendo todas as especificações da etapa de Implementação e funções bem definidas e tratadas.

2 **Documentação do Trabalho**: Relatório completo do trabalho em formato PDF contendo:

(a) **Capa:** Capa formatada e identificada com os dados necessários e recomendados pela ABNT, identificação aluno e título do trabalho.

(b) **Sumário:** enumeração das divisões, seções, capítulos e outras partes do trabalho, seguindo a mesma ordem e grafia em que o conteúdo nele se sucede.

(c) **Introdução:** descrição do problema a ser resolvido e visão geral sobre o funcionamento do programa.

(d) **Implementação:** descrição sobre a implementação do programa e algoritmos utilizados. Devem ser detalhadas as estruturas de dados utilizadas, o funcionamento das principais funções e procedimentos utilizados, o formato de entrada e saída de dados, bem como decisões tomadas relativas aos casos e detalhes de especificação que possam estar omissos no enunciado.

(e) **Análise de Complexidade:** descrição e análise da complexidade dos algoritmos implementados (notação Big O) e do tempo de execução para cada método implementado.

(f) **Resultados:** descrição detalhada dos testes executados, apresentando análises comparativas e críticas. Apresente os principais resultados em tabelas e/ou gráficos.

(g) **Conclusão:** comentários gerais sobre o trabalho e considerações finais sobre a comparação entre algoritmos. Principais dificuldades encontradas em sua implementação ou compreensão.

(h) **Bibliografia:** referência teórica utilizada para o desenvolvimento do trabalho, incluindo sites da *Internet*, se for o caso. A referência deve ser citada no texto quando for utilizada.

(i) **Apêndice:** código fonte com as principais funções utilizadas e detalhadas no trabalho.

Comentários Gerais

- Clareza, indentação e comentários no programa serão bem avaliados.
- O trabalho é individual. Trabalhos identificados como cópias terão a nota dividida ou zerada entre os alunos.
- Criatividade é ser inovador, diferente, ter ideias revolucionárias e então mudar o mundo. Comece sendo criativo nas pequenas coisas e tarefas.