Instituto Federal de Minas Gerais - Campus Ouro Branco
Curso: Sistemas de Informação
Disciplina: Projeto e Análise de Algoritmo
Professor: Saulo Henrique Cabral Silva (saulo.cabral@ifmg.edu.br)

Trabalho Prático 2
Análise de algoritmos de ordenação

Objetivos: Consiste em rever conceitos básicos de programação, com o foco na análise dos algoritmos de ordenação de estruturas lineares

Descrição:

Os algoritmos de ordenação ou de classificação de registros, tem como objetivo, organizar os registros em ordem crescente ou decrescente, e assim facilitar a recuperação de ocorrências de determinado elemento no conjunto de dados. Neste trabalho você irá analisar o desempenho de quatro algoritmos de ordenação em diferentes cenários. O principal objetivo é identificar em quais cenários um algoritmo supera os demais.

Entrada:

Neste trabalho você deverá avaliar os seguintes algoritmos de ordenação: *insertion sort, selection sort, merge sort, heapsort*. Para avaliar o comportamento dos algoritmos, cada algoritmo deverá ser testado com 5 tamanhos de instâncias distintas (2000000, 1000000, 500000, 250000, 125000). Para cada tamanho de instância, os dados deverão ser organizados seguindo 4 distribuições distintas (ordem crescente, ordem decrescente, aleatório, 50% ordenados).

Seguindo as instruções acima, cada algoritmo será testado em 20 cenários diferentes. Utilizando o exemplo do algoritmo *insertion sort*, o mesmo será testado nas seguintes combinações de **tamanho de instância x distribuição dos dados**:

1) *2000000 de elementos x dados ordenados de forma crescente.*
2) *2000000 de elementos x dados ordenados de forma decrescente.*
3) *2000000 de elementos x dados gerados de forma aleatória.*
4) *2000000 de elementos x 50% dos dados contidos no arranjo ordenados.*
5) *1000000 de elementos x dados ordenados de forma crescente.*
6) *1000000 de elementos x dados ordenados de forma decrescente.*
7) *1000000 de elementos x dados gerados de forma aleatória.*
8) *1000000 de elementos x 50% dos dados contidos no arranjo ordenados.*
9) *500000 de elementos x dados ordenados de forma crescente.*
10) *500000 de elementos x dados ordenados de forma decrescente.*

11) *500000 de elementos x dados gerados de forma aleatória.*
12) *500000 de elementos x 50% dos dados contidos no arranjo ordenados.*
13) *250000 de elementos x dados ordenados de forma crescente.*
14) *250000 de elementos x dados ordenados de forma decrescente.*
15) *250000 de elementos x dados gerados de forma aleatória.*
16) *250000 de elementos x 50% dos dados contidos no arranjo ordenados.*
17) *125000 elementos x dados ordenados de forma crescente.*
18) *125000 elementos x dados ordenados de forma decrescente.*
19) *125000 elementos x dados gerados de forma aleatória.*
20) *125000 elementos x 50% dos dados contidos no arranjo ordenados.*

Ao término de cada cenário de teste, anote o tempo que o algoritmo de ordenação demandou para ordenar a instância passada como argumento. Cada linguagem de programação fornece uma forma de colher esse tempo. Abaixo, apresento na linguagem Java, por considerar que a maioria dos alunos matriculados nesta disciplina possuem maior prática.

```
long tempoInicio = System.currentTimeMillis();

selectSort(vet);
long tempoFim = System.currentTimeMillis();

//tempo em milissegundos
long tempoTotal = tempoFim - tempoInicio;
```

Finalizando os testes de todos os cenários com os quatro algoritmos propostos, **apresente gráficos de tempo de execução, e realize comparações com o objetivo de identificar qual algoritmo mais eficiente e qual o menos eficiente para um determinado cenário**. Em seguida discorra uma justificativa do comportamento do melhor e do pior algoritmo para cada cenários de teste.

O que deve ser entregue:

1. A linguagem utilizada para implementar cada um dos algoritmos fica à cargo do grupo (códigos devem estar bem *indentados* e comentados).
2. Documentação do trabalho. Entre outras coisas, a documentação deve conter:
   2.1. Introdução: descrição do problema a ser resolvido e visão geral sobre o funcionamento do programa.
   2.2. Análise e tempos de execução: apresentar os tempos de execução anotados para cada algoritmo proposto em todos os cenários de testes. Apresente comparações entre os algoritmos, no intuito de encontrar o algoritmo mais eficiente e o mais ineficiente para cada cenário de teste. Discorra sobre o funcionamento dos algoritmos selecionados, explicando o motivo do mesmo ser o mais eficiente, ou o mais ineficiente.
   2.3. Conclusão: comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação e análise.
   2.4. Bibliografia: bibliografia utilizada para o desenvolvimento do trabalho, incluindo sites da Internet se for o caso
3. Formato: mandatoriamente em PDF (http://www.pdf995.com/).

Obs1: Apesar desse trabalho ser bem simples, a documentação pedida segue o formato da documentação que deverá ser entregue nos próximos trabalhos.

Obs2: Consulte as dicas do Prof. Nívio Ziviani de como deve ser feita uma boa implementação e documentação de um trabalho prático: http://sauloifmg.com.br/roteirotp.pdf

Como deve ser feita a entrega:

A entrega **DEVE** ser feita via plataforma moodle na forma de um único arquivo *zipado*, contendo o código, os arquivos, o executável e a documentação.

Comentários Gerais:

- Comece a fazer este trabalho logo, enquanto o problema está fresco na memória e o prazo para terminá-lo está tão longe quanto jamais poderá estar;
- Clareza, *indentação* e comentários no programa também vão valer pontos;
- O trabalho pode ser feito em duplas (**grupos de MÁXIMO 2 alunos**);
- Trabalhos copiados (e **FONTE**) terão nota ZERO;
- Trabalhos entregue em atraso serão aceitos, todavia a nota atribuída ao trabalho será zero
- Evite discussões inócuas com o professor em tentar postergar a data de entrega do referido trabalho.