**Universidade Federal de Ouro Preto – UFOP**
**Instituto de Ciências Exatas e Biológicas – ICEB**
**Departamento de Computação – DECOM**
**Disciplina: Algoritmos e Estrutura de Dados I – CIC102 / 20092**
**Professor: David Menotti (menottid@gmail.com)**

# Trabalho Prático 5
# Ordenação – Análise de Desempenho de Algoritmos de Ordenação por Comparação e suas Variações e Otimizações

**Valor: 0,5 pts (5% da nota total)** **Data de entrega:**
**Documentação não-Latex: -0,1 pts** **19/11/2009**
**Interface gráfica: +0,1 pts**
**Impressão não frente-verso: -0,05 pts (ecologicamente correto)**

O objetivo deste trabalho é analisar algoritmos de ordenação por comparação e suas variações e otimizações. Essa análise será dividida em duas partes.

Na primeira, a análise será sobre os algoritmos de ordenação simples (de ordem de complexidade $O(n^2)$ - *i.e.*, BubbleSort, SelectSort, InsertSort e algumas variações). Você deverá implementar os três algoritmos citados e implementar variações desses algoritmos de forma que através de análises de experimentos você possa analisar:

- Se vale a pena inserir a verificação de ordenação (houve troca) no algoritmo BubbleSort;
- Se vale a pena usar o algoritmo InsertSort com elemento sentinela;
- Se vale a pena usar um algoritmo InsertSort tendo o arranjo implementado através de apontadores/cursores;
- Se vale a pena inserir uma verificação (Min == i) para evitar a troca, no método SelectSort.

A segunda parte será sobre os algoritmos de ordenação por comparação ditos eficientes, como MergeSort, HeapSort e QuickSort, que tem complexidade de tempo de $O(n \log n)$. São fornecidas implementações convencionais e versões otimizadas. Aproveite os algoritmos implementados por você e verifique:

- Até que tamanho de entrada, vale a pena usar algoritmos $O(n^2)$ com relação a algoritmos O(n log n)

## Como gerar os vetores

Considere vetores com a quantidade de elementos variando, *i.e.*, 10, 100, 1.000, 10.000, 100.000, 1.000.000, *etc*. Considere também vetores **sem valores repetidos**. Ainda, considere que todos os elementos dos vetores correspondem a valores inteiros e para gerar os vetores inicias, utilize vetores ordenados, inversamente ordenados, quase ordenados e aleatórios.

## O que analisar

A análise deve ser feita sobre o número de comparações, atribuições e tempo de execução dos algoritmos. Procure organizar inteligentemente os dados coletados em tabelas, e também construa

gráficos a partir dos dados. Então, disserte sobre os dados nas tabelas e gráficos. Grande parte da avaliação será realizada sobre a análise dos resultados, ou seja, sobre o que você dissertar.

## As implementações dos algoritmos

Neste trabalho, está sendo disponibilizado (no site da disciplina) implementações de algoritmos a serem utilizados para análise da segunda parte do trabalho. **A compilação, interpretação e uso deste código constitui parte da avaliação deste trabalho prático.** A implementação dos outros algoritmos a serem utilizados na primeira parte, fica por sua conta. Você deverá entender o padrão de programação do código fornecido e implementar os novos algoritmos seguindo esse padrão.

Salienta-se que o uso desse programa pode ser realizado em linha de comando (*prompt*), ou dentro do ambiente de programação que você usa. Todavia, antes, você deverá compilar o código para gerar o arquivo executável e configurar o arquivo *in.dat* para realizar seus testes. Também está sendo fornecido um arquivo exemplo de configuração – é apenas ilustrativo – não o pegue/tome como referência para seus testes. O resultado da execução do programa, ou seja os dados para você analisar, são gravados em *log.txt*.

**Universidade Federal de Ouro Preto – UFOP**
**Instituto de Ciências Exatas e Biológicas – ICEB**
**Departamento de Computação – DECOM**
**Disciplina: Algoritmos e Estrutura de Dados I – CIC102 / 20092**
**Professor: David Menotti (menottid@gmail.com)**

## O que deve ser entregue

- Código fonte do programa em C ou C++ (bem identada e comentada).
- Documentação do trabalho. Entre outras coisas, a documentação deve conter:
  1. Introdução: descrição do problema a ser resolvido e visão geral sobre o funcionamento do programa.
  2. Implementação: descrição sobre a implementação dos algoritmos. Deve ser detalhada a estrutura de dados utilizada (de preferência com diagramas ilustrativos), o funcionamento dos algoritmos, e decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado. Muito importante: os códigos utilizados na implementação devem ser inseridos na documentação.
  3. Análise dos testes: tabular, construir gráficos a partir dos dados gerados pelo programa e analisar densamente os resultados.
  4. Conclusão: comentários gerais sobre quais algoritmos são os mais eficientes dados vetores particulares.
  5. Bibliografia: bibliografia utilizada para o desenvolvimento do trabalho, incluindo sites da Internet se for o caso. Uma referência bibliográfica deve ser citada no texto quando da sua utilização
  6. Em Latex: Caso o trabalho não seja elaborado/escrito em latex, perde-se 0,1 pontos.
  7. Impressão: Caso o trabalho não seja impresso usando modo frente-verso, perde-se 0,05 pontos.
  8. Formato: mandatoriamente em PDF (http://www.pdf995.com/).

  Obs.: Veja modelo de como fazer o trabalho em latex:
  http://www.decom.ufop.br/prof/menotti/aedI092/tps/modelo.zip

## Como deve ser feita a entrega:

A entrega DEVE ser feita via Moodle (www.decom.ufop.br/moodle) na forma de um **único** arquivo zipado, contendo o código, os arquivos e a documentação. Também deve ser entregue a documentação impressa na próxima aula teórica após a data de entrega do trabalho.

## Comentários Gerais:

- Clareza, identação e comentários no programa também vão valer pontos;
- O trabalho é individual (grupo de UM aluno);
- Trabalhos copiados (e FONTE) terão nota zero;
- Trabalhos entregues 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.