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 / 20091
Professor: David Menotti (menottid@gmail.com)

Trabalho Prático 5

Ordenação – Análise de Desempenho de Algoritmos Eficientes de Ordenação por Comparação (MergeSort, HeapSort e QuickSort) e BucketSort e RadixSort

Valor: 0,5 pts (5% da nota total) **Data de entrega:**
Documentação não-Latex: -0,1 pts **23/06/2009**
Impressão não Frente-Verso: -0,05 pts (não ecologicamente correto)
Interface gráfica: +0,1 pts
Entrega até 17/06/2009: +0,2pts

O objetivo deste trabalho é analisar algoritmos eficientes comumente usados de ordenação por comparação, como MergeSort, HeapSort e QuickSort, que tem complexidade de tempo de $O(n \log n)$, com algoritmos de ordenação especiais, como o BucketSort e RadixSort, que tem complexidade linear, *i.e.*, $O(n)$.

Estes algoritmos são ditos especiais pois necessitam de vetores com configuração restrita e algum conhecimento *a priori* sobre os dados do vetor. No caso do **BucketSort**, é desejável que o domínio do vetor (o valor mínimo e máximo) seja bem menor que o número de elementos. No caso do **RadixSort**, é necessário que se conheça a quantidade de dígitos do maior elemento do vetor.

Como gerar os vetores

Para este trabalho, quando testar/analisar o algoritmo **BucketSort** considere que o tamanho de cada *bucket* seja 1.

Considere também vetores que tenham muitos números repetidos. Seja **FatorRepetição** a relação entre o número de elementos do vetor e o domínio do vetor (diferença entre o valor máximo e mínimo) maior que 1. Realize testes para diversos tamanhos de vetores: 10.000, 100.000, 1.000.000, *etc*. e também utilize diversos fatores de repetição (**FatorRepetição**): 1, 2, 4, 8, 16, 32, 64, 128, *etc*.

Considere ainda vetores com a quantidade de elementos variando, *i.e.*, 10.000, 100.000, 1.000.000, *etc*. Os vetores devem ter valores repetidos e a quantidade de dígitos do maior elemento do vetor variando entre 3, 4, 5, 6, 7, 8, *etc*, no caso de avaliar o algoritmo **RadixSort**.

Os limites superiores para os tamanhos dos vetores deverão ser determinados pela plataforma (computador/sistema operacional) que você está usando, ou seja, aumente o tamanho até que sua plataforma consiga armazenar os vetores e as estruturas auxiliares em memória principal e o tempo de ordenação seja inferior a 10 minutos.

Também 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.

As implementações dos algoritmos

Neste trabalho, está sendo disponibilizado (no site da disciplina) as implementações dos algoritmos a serem utilizados para análise. Tais implementações são cortezia do aluno Kayran dos Santos. **A compilação, interpretação e uso deste código constitui parte da avaliação deste trabalho prático.**

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*.

O que deve ser entregue

- Documentação do trabalho. Entre outras coisas, a documentação deve conter:
  1. Introdução: descrição da análise a ser realizada e visão geral sobre os algoritmos analisados.
  2. Análise dos testes: tabular, construir gráficos a partir dos dados gerados pelo programa e analisar densamente os resultados.
  3. Conclusão: comentários gerais sobre quais algoritmos são os mais eficientes dados vetores particulares.
  4. Bibliografia: bibliografia utilizada para o desenvolvimento do trabalho, incluindo sites da Internet se for o caso
  5. Em Latex: Caso o trabalho não seja elaborado/escrito em latex, perde-se 0,1 pontos.
  6. Impressão: Caso o trabalho não seja impresso usando os dois lados da folha (impressão frente-verso), perde-se 0,05 pontos extra – um incentivo à "preservar o planeta".
  7. Formato: mandatoriamente em PDF (http://www.pdf995.com/).

Obs2: Veja modelo de como fazer o trabalho em latex:
http://www.decom.ufop.br/prof/menotti/aedI091/tps/modelo.zip

(caso alguém desenvolva um modelo similar em word, favor me enviar)

Como deve ser feita a entrega:

A entrega DEVE ser feita pelo Moodle na forma de um **único** arquivo zipado, contendo os arquivos usados e a documentação. Também deve ser entregue a documentação impressa na próxima aula (teórica ou prática) após a data de entrega do trabalho.

Comentários Gerais:

- Comece a fazer este trabalho logo, enquanto o conteúdo está fresco na memória e o prazo para terminá-lo está tão longe quanto jamais poderá estar;
- O trabalho é individual (grupo de UM aluno);
- 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.