**LABORATÓRIO DE PROGRAMAÇÃO AVANÇADA**

**DÉCIMO TRABALHO PRÁTICO**

**COMPARAÇÃO DE MÉTODOS DE ORDENAÇÃO**

## OBJETIVO

Relembrar os conceitos de ordenação através da implementação de seis métodos e da comparação do desempenho entre eles.

## PARTE 1

Deve ser enviado um relatório contendo os dados abaixo.

| | | **Aleatórios** | | | | |
|---|---|---|---|---|---|---|
| Métrica | n | 1K | 5K | 10K | 20K | 50K |
| Número de comparações | Bolha | | | | | |
| | Seleção | | | | | |
| | Inserção | | | | | |
| | Heapsort | | | | | |
| | Mergesort | | | | | |
| | Quicksort | | | | | |
| Número de trocas | Bolha | | | | | |
| | Seleção | | | | | |
| | Inserção | | | | | |
| | Heapsort | | | | | |
| | Mergesort | | | | | |
| | Quicksort | | | | | |
| Tempo | Bolha | | | | | |
| | Seleção | | | | | |
| | Inserção | | | | | |
| | Heapsort | | | | | |
| | Mergesort | | | | | |
| | Quicksort | | | | | |

Como pode-se perceber, os seguintes métodos de ordenação devem ser implementados: Bolha, Inserção, Seleção, Heapsort, Mergesort, e Quicksort. A comparação entre os métodos de ordenação será feita através dos seguintes parâmetros: (a) número de comparações; (b) número de trocas; e (c) tempo de execução. As ordenações devem ser feitas em diferentes tamanhos de vetores, isto é, 1000, 5000, 10000, 20000 e 50000. Estes vetores deverão ser gerados ALEATORIAMENTE. Usar a mesma massa de dados para TODOS os algoritmos de ordenação.

**IMPORTANTE**: Para minimizar possibilidades de que sejam gerados efeitos locais e, consequentemente, evitar viés no experimento, os dados que vão para o relatório acima são o VALOR MÉDIO de 10 (dez) execuções de cada algoritmo. Em cada uma das dez execuções, deve-se gerar novos números aleatórios. Não esqueça de usar a mesma massa de dados para TODOS os algoritmos de ordenação.

**Resumindo**: gera-se um vetor de **1000** elementos **aleatórios** e aplica-se os algoritmos Bolha, Inserção, Seleção, Heapsort, Mergesort e Quicksort sobre esse vetor e armazena-se o número de comparações, o número de trocas e o tempo de execução de cada algoritmo. Gera-se **novo**

**vetor** de 1000 elementos **aleatórios** e aplica-se novamente os seis algoritmos de ordenação e armazena-se novamente os três parâmetros de comparação. Repete-se mais **oito** vezes. Ao final, tira-se a **MÉDIA** do número de comparações, do número de trocas e do tempo de execução dos seis algoritmos. Preenche-se na tabela acima na coluna 1K.

**OBSERVAÇÃO**: Apesar de estar 1K, não significa 1024, mas 1000.

Repete-se o mesmo procedimento acima para as quantidades 5000, 10000, 20000 e 50000.

## PARTE 2

O objetivo é plotar três gráficos das medições feitas, um para cada parâmetro: (a) número de comparações; (b) número de trocas; e (c) tempo de execução.

Para cada gráfico, deve-se incluir os dados dos seis métodos de ordenação, isto é, Bolha, Inserção, Seleção, Heapsort, Mergesort e Quicksort. Não esqueça de adicionar legendas para os métodos de ordenação.

Nos três gráficos, o eixo das abcissas (X) deve ser os tamanhos dos vetores: 1000, 5000, 10000, 20000 e 50000. No eixo das ordenadas (Y) deve ser um dos 3 parâmetros da comparação (comparações, trocas e tempo).

Resumindo, deverão ser entregues 3 gráficos, um para o número de comparações, um para o número de trocas, e outro para o tempo de execução. Sugiro que os gráficos sejam em linha.

**OBSERVAÇÃO**: como os dados do bolha devem ser bem maiores do que os dos outros, recomendo que, se for o caso, vocês coloquem o eixo dos Y (ordenadas) em ESACALA LOGARÍTMICA. Vi que o Excel, Calc do LibreOffice e GNU Plot tem essa opção. Vocês podem usar **qualquer software para fazer os gráficos.**

## PARTE 3

Além da tabela e dos gráficos, incluir respostas para as seguintes perguntas:

- Para cada método de ordenação, qual foi seu melhor e pior caso observado?
- Qual função melhor descreve o desempenho de cada método (n, $n^2$, $\log_2 n$, $n.\log_2 n$)?
- Em quais métodos (e quantidade de dados nos vetores), o número de comparações é um bom substituto para o tempo de execução (ou seja, as duas métricas dão resultados relativos parecidos)?
- De forma geral, qual(is) o(s) melhor(es) algoritmo(s) dentre os testados?

## OBSERVAÇÕES GERAIS

- Se você implementar em C, use a função **`gettimeofday()`** para calcular tempos, e **`srand()`** e **`rand()`** para gerar os números aleatórios (no caso do Linux)
- Em Java tem o método **`System.currentTimeMillis()`** e a classe **`Random`** (disponível na biblioteca **`java.util`**)
- Mantenha sempre as mesmas unidades para comparação, por exemplo, tempo sempre em milissegundos
- Este trabalho deve ser entregue até o dia **04/08/2016 (quinta)** até meia-noite. Após este prazo começa a contar o desconto progressivo, isto é, 1 ponto no primeiro dia, 3 pontos no segundo dia e 6 pontos no terceiro dia.
- Envie para o professor (xbarretox@gmail.com) e o monitor (jbc@icomp.ufam.edu.br)
- No email, favor colocar no assunto: [LPAV-TP10]