Pontifícia Universidade Católica de Goiás
Departamento de Computação
CMP1054 - Estrutura de Dados II
Max Gontijo de Oliveira
1º Trabalho - Algoritmos de ordenação simples

1 Resumo

O objetivo deste trabalho é permitir ao aluno implementar os algoritmos de ordenação vistos e realizar uma avaliação empírica de desempenho.

2 Detalhes

O trabalho consiste em implementar e comparar o tempo de execução (em milissegundos) dos algoritmos de ordenação vistos em sala: ***selection sort***, ***insertion sort*** e ***bubble sort***. Os mesmos devem ser utilizados para ordenar diversos vetores (apenas de números inteiros). Mais especificamente, você deverá realizar testes com vetores de tamanhos 100, 1.000, 10.000, 100.000 e 1.000.000 de elementos.

Três diferentes tipos de vetores devem ser utilizados: aleatórios, ordenados em ordem crescente, e ordenados em ordem decrescente. Para os vetores aleatórios, faça uma cópia do vetor original que servirá de backup - usada para restaurar o vetor. Desta forma, será executada a ordenação sob as mesmas condições para os três algoritmos, mas a cada nova solicitação de execução o seu programa deverá gerar um novo vetor com elementos aleatórios que deve ser usado pelos três algoritmos de ordenação. Além disso, você pode repetir os testes várias vezes de forma a obter médias do tempo de execução.

Também faz parte do trabalho a construção de tabelas e/ou gráficos comparando o desempenho de cada algoritmo e os resultados devem apresentar a relação Tamanho X Tempo. Bem como, descobrir como medir o tempo de execução e gerar os números aleatórios na linguagem.

2.1 Implementação

O seu programa deverá permitir ao usuário selecionar o tipo do vetor a ser ordenado (aleatório, crescente ou decrescente) e o tamanho do vetor - executando a ordenação para os três algoritmos. Também deverá mostrar na tela o nome do algoritmo utilizado e o tempo de execução gasto para ordenar o vetor. Por exemplo:

```
selection sort: 2109794.434 ms
insertion sort: 2627440.718 ms
bubble sort   : 4756249.523 ms
```

2.1.1 Considerações

As seguintes considerações devem ser seguidas:

- O tempo de execução gasto pelo algoritmo deverá ser determinado a partir da chamada à função de ordenação até o seu término;
- Todos os testes devem ser realizados na mesma máquina e, para garantir resultados válidos, não pode haver programas sendo executados em paralelo ou qualquer outra atividade que possa interferir no desempenho da máquina;
- Todos os algoritmos de ordenação deverão estar no mesmo arquivo sort.h/sort.hpp, 1ou seja, criar uma biblioteca de ordenação;
- Use comentários relevantes em seu código, pois serão avaliados.

3 Entrega

O trabalho deverá ser entregue em duas etapas: a primeira é a entrega dos arquivos de código fonte e o trabalho escrito via e-mail para o endereço `max.pucgo@maxoliveira.com.br` com o seguinte assunto:

`[CMP1099] Trabalho 1 - Algoritmos de ordenação simples`

A segunda é a apresentação do trabalho em sala de aula diretamente para o professor.

3.1 Código fonte

O arquivo do trabalho escrito (preferencialmente em PDF) e os arquivos de código fonte deverão ser compactados em um único arquivo (ZIP, RAR, 7Z, etc.) e enviados até o dia **19/02/2015** às **18:45**.

3.2 Apresentação

Todos os alunos deverão apresentar o trabalho ao professor em sala de aula juntamente com os demais trabalhos aplicados na N1. Nenhum trabalho deverá ser finalizado no dia/hora da apresentação. A data de apresentação será definida em sala de aula.

A apresentação será realizada individualmente e irá considerar a capacidade do aluno de explicar todo o código desenvolvido ou partes específicas determinadas pelo professor. Além disso, poderão ser solicitadas alterações no dia da apresentação.

4 Avaliação

A avaliação será realizada em duas etapas: uma referente à entrega do programa e do trabalho escrito; a outra referente à apresentação.

A entrega vai considerar o respeito ao prazo, o funcionamento do programa no dia da apresentação e a qualidade das informações de comparação entre os algoritmos apresentadas no trabalho escrito.

A apresentação vai considerar a explicação do aluno acerca do código desenvolvido e a capacidade de o aluno responder perguntas sobre o código ou sobre os algoritmos. O professor poderá ainda solicitar alterações no código que o aluno deverá ser capaz de realizar.

Bônus: será concedido na nota do trabalho 0,5 pontos adicionais para as implementações que forem construídas utilizando alguma interface rica (QT, GTK, OpenGL, etc.) e que estejam funcionando.