MINISTÉRIO DA EDUCAÇÃO E DO DESPORTO
Universidade Federal de Ouro Preto - UFOP
Instituto de Ciências Exatas e Biológicas - ICEB
Programa de Pós-Graduação em Ciência da Computação
Disciplina: Projeto e Análise de Algoritmos
Professor: David Menotti Gomes

Trabalho Prático 0 - *Warm up*

Implementação e Análise de Algoritmos de Ordenação por Comparações de Chaves usando Arranjos e Apontadores

Valor: 0,8 pontos (8% da nota total) Data de Entrega: 21/03/2011

Objetivos

Entre outros:

- Fixar e/ou aprender sintaxe de programação em C/C++;
- Relembrar conceitos sobre algoritmos de ordenação por comparações de chaves;
- Realizar a implementação destes algoritmos usando apontadores;
- Analisar a complexidade dos algoritmos implementados;
- Realizar experimentação com vários tipos e tamanhos de entradas de dados, tecendo análises sobre os resultados.

Descrição

Neste trabalho, você deve implementar algoritmos não-eficientes (`BubbleSort`, `InsertionSort` e `SelectionSort`) e eficientes (`QuickSort`, `HeapSort` e `MergeSort`) para ordenar sequências de elementos[1]. Você deve apresentar implementações dos algoritmos citados onde a sequência de dados a ser ordenada pode estar representada por arranjos (vetores) ou apontadores. Lembre-se que implementações dos algoritmos de ordenação por arranjos são facilmente encontrados na literatura, e devem ser utilizadas. Todavia, o domínio e conhecimento das implementações podem ser objeto de avaliação em entrevista.

[1]Uma sequência difere de um conjunto no fato de que em conjunto não há ordem entre os elementos, enquanto que em uma sequência há.

Cada elemento da sequência de dados a ser ordenada deve conter um campo `chave` e pelo menos um campo "periférico". A especificação e desenvolvimento dos tipos abstratos de dados (as `struct`'s em C) a serem utilizados no trabalho são de sua competência e fazem parte da avaliação.

Como gerar os vetores/arranjos para ordenação

Considere sequências de dados com a quantidade de elementos variando, *i.e.*, 10, 100, 1000, 10000, 100000, 1000000, *etc.* Considere também sequências **sem valores repetidos**. Ainda, considere que todos os elementos das sequências correspondem a valores inteiros e para gerar as sequências inicias utilize sequências **ordenadas**, **inversamente ordenadas**, **quase ordenadas** e **aleatórias**.

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 dedicada a análise dos resultados, ou seja, sobre o que você dissertar.

Para comparação, relacione somente os algoritmos não-eficientes e eficientes entre si. Por exemplo, não relacione, analise ou compare, o algoritmo `BubbleSort` com o `QuickSort`.

O que deve ser entregue

- Código fonte do programa em C ou C++ (bem identado e comentado).
- 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 do programa. Devem ser detalhadas as estruturas de dados utilizadas (de preferência com diagramas ilustrativos), o funcionamento das principais funções e procedimentos utilizados, o formato de entrada e saída de dados, bem como decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado. **Muito importante**: os códigos utilizados nas implementações devem ser inseridos na documentação.
3. Análise de Complexidade: estudo da complexidade de tempo e espaço das funções implementadas e do programa como um todo (notação O).
4. Listagem de testes executados: os testes executados devem ser apresentados e analisados e discutidos, quando convier.

5. Conclusão: comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação.

6. Bibliografia: bibliografia utilizada para o desenvolvimento do trabalho, incluindo sítio da Internet se for o caso. Uma referência bibliográfica deve ser citada no texto quando da sua utilização.

7. Em LaTeX: A documentação deve ser elaborada **obrigatoriamente** em LaTeX. Veja modelo de como fazer o trabalho em latex: `http://www.decom.ufop.br/menotti/paa111/tps/modelo.zip`

8. Formato final: mandatoriamente em PDF (`http://www.pdf995.com/`).

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 fonte, arquivos diversos e a documentação. Também deve ser entregue a documentação impressa na próxima aula teórica após a data de entrega do trabalho.

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, identação e comentários no programa também serão avaliados;

- O trabalho é individual (grupo de UM aluno);

- Trabalhos copiados (e FONTE) terão nota zero. Devido a recorrentes problemas com cópias de trabalhos (plágios), os autores de trabalhos copiados também terão a maior nota dentre os testes teóricos levada a zero, como forma de punição e coação ao plágio acadêmico;

- 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.