Instituto Federal de Educação, Ciência e Tecnologia
Curso: Bacharelado em Ciência da Computação
Disciplina: Algoritmos e Estruturas de Dados I
Professor: Mário Luiz Rodrigues Oliveira
Atividade: 1º Trabalho Prático
Formiga, MG, 18 de junho de 2014

INSTRUÇÕES:

1. **Esta atividade pode ser resolvida em grupo composto por no máximo 2 integrantes.**
2. **Caso você ache que falta algum detalhe nas especificações, você deverá fazer as suposições que julgar necessárias e escrevê-las no seu relatório. Pode acontecer também que a descrição dessa atividade contenha dados e/ou especificações supérfluas para sua solução. Utilize sua capacidade de julgamento para separar o supérfluo do necessário.**
3. **Como produtos da atividade serão gerados dois artefatos: códigos fontes da implementação e documentação da atividade.**
4. **Cada arquivo-fonte deve ter um cabeçalho constando no mínimo as seguintes informações: nome(s) e matrícula do(s) autor(es) do trabalho.**
5. **O arquivo contendo a documentação da atividade (relatório) deve ser devidamente identificado com o(s) nome(s) e matrícula do(s) autor(es) do trabalho. O arquivo contendo o relatório deve, obrigatoriamente, estar no formado PDF.**
6. **Devem ser entregues os arquivos contendo os códigos-fontes e o arquivo contendo a documentação da atividade (relatório). Compacte todos os artefatos gerados num único arquivo no formato RAR. Envie apenas um arquivo por grupo.**
7. **O trabalho deve ser implementado usando-se, exclusivamente, a linguagem de programação C .**
8. **O prazo final para entrega desta atividade é até as 23:59:00 do dia 14/07/2014 via portal acadêmico acessado pela URL:https://meu.ifmg.edu.br/.**
9. **O envio é de total responsabilidade do aluno. Não serão aceitos trabalhos enviados fora do prazo estabelecido.**
10. **Trabalhos plagiados serão desconsiderados, sendo atribuída nota 0 (zero) a todos os envolvidos.**
11. **O valor desta atividade é 10 pontos.**

1. Objetivo e Descrição do Trabalho

O objetivo deste trabalho é realizar um estudo comparativo e empírico de diversos algoritmos de ordenação interna. "A ordenação interna é utilizada quando todos os registros cabem na memória principal" (Ziviani, 2011). O estudo deve comparar, obrigatoriamente, os seguintes algoritmos de ordenação interna: *bubble sort*, *selection sort*, *insertion sort*, *shellsort*, *heapsort* e *quicksort*. O método de escolha do pivô é de fundamental importância no algoritmo de ordenação *quicksort*: Neste trabalho devem ser implementadas 5 alternativas para a escolha do pivô: o primeiro elemento do vetor, o último elemento do vetor, o elemento do meio do vetor, mediana de três elementos e uma posição do vetor escolhida aleatoriamente. Uma importante melhoria para o desempenho do algoritmo de ordenação *quicksort* é evitar chamadas recursivas para vetores de tamanho pequeno. Para tanto pode-se combinar o algoritmo *quicksort* com o algoritmo de ordenação *insertion sort*. Nessa abordagem, os dados são inicialmente ordenados com o *quicksort* e vetores menores que **M** elementos são ordenados com o algoritmo *insertion sort.* Essa melhoria deve ser estudada nesse trabalho com o objetivo de determinar o valor adequado para o parâmetro **M**. De forma a comparar todos os algoritmos citados, implemente-os na forma de um programa. Para o algoritmo *bubblesort* considere a implementação sugerida em sala de aula e para os demais algoritmos considere as implementações propostas por Ziviani (Ziviani, 2011).

O programa implementado deve atender as seguintes especificações:

- o código fonte do programa deve ser portável, ou seja, o mesmo código fonte deve ser compilado corretamente e gerar código executável paras as seguintes plataformas: sistemas operacionais Windows e GNU/Linux e arquiteturas *Intel* x86/x86-64 e *AMD*64/x86-64
- o código fonte do programa deve ser compatível com o seguinte compilador: Free Pascal (FPC verão 2.6.2)

Para comparar as implementações dos algoritmos de ordenação utilize os seguintes critérios: tempo médio de execução, quantidade média de comparações realizadas pelo algoritmo e número médio de trocas (movimentações) de elementos executadas pelo algoritmo.

Adicionalmente, gere massas de dados que permitam testar os algoritmos e realizar as medições solicitadas. Os dados usados nos testes devem ser valores inteiros aleatórios. Considere, ainda, a geração de massa de dados de diferentes tamanhos, a saber: 500, 2.000, 5.000, 10.000, 30.000, 50.000, 100.000, 150.000, 200.000, 250.000 e 300.000 valores inteiros.

Com o intuito de realizar os testes e as medições solicitadas você pode incluir as seguintes funcionalidades no seu programa:

- um gerador de números aleatórios para gerar vetores de números inteiros com 500, 2.000, 5.000, 10.000, 30.000, 50.000, 100.000, 150.000, 200.000, 250.000 e 300.000 elementos;
- contadores (que devem ser atualizados pelos procedimentos/funções de ordenação) para armazenar o número de comparações e de trocas de elementos executados pelos algoritmos;
- estruturas de dados que permitam armazenar o tempo de execução de cada um dos 6 algoritmos de ordenação indicados usando o relógio da máquina (computador), e
- uma função que verifique se a massa de dados foi corretamente ordenada pelos algoritmos comparados.

Afim de obter os valores médios de cada medida de comparação, execute o programa 100 vezes para cada massa de dados. Assim, o programa deve ser executado 100 vezes com vetores de 500 elementos e realizar as medições solicitadas para todos os algoritmos de ordenação indicados. Posteriormente deve-se executar o programa 100 vezes com vetores de 2.000 elementos e realizar as medições solicitadas para todos os algoritmos de ordenação indicados, e assim sucessivamente até realizar as medições com vetores de 300.000 elementos. A cada nova execução devem ser usados vetores com conteúdos diferentes[1]. No entanto, numa mesma iteração os algoritmos devem ser comparados e analisados com vetores do mesmo tamanho e conteúdo.

2. Artefatos

Esta seção descreve o que deve ser gerado como produto final do trabalho. Ao final do trabalho deve ser gerado além das implementações, um relatório documentando seu programa, com as seguintes informações:

1. introdução: apresente o problema a ser resolvido. Descreva sucintamente a solução proposta e dê uma visão geral do que será apresentado no relatório;
2. implementação: descrição sobre as decisões de projeto e implementação do programa. Essa parte da documentação deve incluir uma descrição das estruturas de dados usadas no

1 A rigor não é possível afirmar que o conteúdo dos vetores é sempre diferente, pois é gerado de forma aleatória. Para os nossos propósitos importa que o conteúdo do vetor seja gerado aleatoriamente.

programa; funcionamento das principais funções e procedimentos utilizados; o formato de entrada e saída dos dados, como executar o programa e as decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado;

3. algoritmos de ordenação: explicação sucinta e com exemplos de cada um dos 6 algoritmos de ordenação comparados. Apresente a complexidade do caso médio, melhor e pior caso dos algoritmos implementados;

4. análise das medidas e comparações realizadas: descreva a metodologia usada para realizar o estudo, incluindo uma descrição do sistema computacional e da massa de dados utilizada nos testes e comparações; a linguagem de programação e o compilador usado na implementação dos algoritmos. Apresente e discuta comparativamente os resultados obtidos (tempo médio de execução, quantidade média de trocas e movimentações para cada algoritmo comparado). Faça uso de tabelas e gráficos para facilitar suas explicações. Dê a sua interpretação para os resultados obtidos, comparando-os com resultados analíticos (teóricos). Compare, também, os resultados com as conclusões e recomendações oriundas do trabalho realizado por Ziviani (Ziviani, 2011). Procure responder aos seguintes questionamentos:

   - a quantidade média de comparações e movimentações são métricas representativas do desempenho dos algoritmos analisados?
   - qual a relação entre a quantidade média de comparações e movimentações e o tempo de execução dos algoritmos analisados?
   - em qual ou quais situações são indicados os algoritmos com complexidade quadrática?
   - em qual ou quais situações são indicados os algoritmos com complexidade O(**n*log n**)?
   - os resultados obtidos empiricamente estão em conformidade com os resultados obtidos analiticamente? Explique e justifique seus resultados, principalmente se os resultados empíricos não confirmarem os resultados analíticos.
   - qual a melhor estratégia para a escolha do pivô no algoritmo de ordenação *quicksort*?
   - qual o melhor valor para o parâmetro **M** usado na combinação do algoritmo *quicksort* com o algoritmo *insertion sort*? Para determinar o melhor valor de **M** considere apenas os testes com vetores de tamanhos: 500, 2.000, 10.000, 50.000, 200.000 e 300.000.

5. conclusão: explicite as recomendações do grupo indicando em quais situações cada um dos algoritmos de ordenação comparados e avaliados devem ser utilizados. Inclua alguns comentários/avaliação sobre o trabalho considerando: a experiência adquirida, a contribuição para o aprendizado da disciplina, as principais dificuldades encontradas ao implementá-lo e como tais dificuldades foram superadas;
6. bibliografia: cite as fontes consultadas na resolução do trabalho;
7. listagem dos códigos-fontes do programa.

Todos os artefatos (códigos fontes e relatório) devem ser entregue conforme as instruções contidas nesse documento.

3. Critérios de Correção

Conforme descrito no plano de aula, os critérios de avaliação do trabalho são:

- somente serão corrigidos os trabalhos com códigos fontes portáveis e sem de erros de compilação;
- apresentação (30%);
- análise código fonte: modularização, uso adequado de comentários, legibilidade, corretude e edentação do código, e (40%)
- documentação (30%).

4. Bibliografia

Ziviani, Nívio. Projeto de Algoritmos: com implementações em Pascal e C. 3ª edição revista e ampliada. São Paulo: Cengage Learning, 2011.