Universidade Federal de Juiz de Fora – UFJF
Departamento de Ciência da Computação
Disciplina: Análise e Projeto de Algoritmos Prof. Stênio Sã. Semestre:2017-1

Especificação do Trabalho (Fase 1)

O desenvolvimento de Hardware com capacidade de processamento cada vez maior pode nos levar a supor que a abordagem usada para o desenvolvimento de algoritmos eficientes para um dado problema possa em breve tornar-se um aspecto irrelevante, o que atribuiria ao estudo de análise de desempenho de algoritmos uma importância meramente teórica. Entretanto, esta é uma visão por demais estática, já que ignora o fato de que o volume de dados produzido pelo homem cresce em uma escala muito maior que o poder de processamento que a indústria de Hardware oferece a cada geração de processadores.

Neste contexto, o estudo de técnicas de análise de algoritmos tem um papel importante não apenas no que se refere à teoria em si, mas pelo que pode oferecer de recurso para a escolha da abordagem utilizada pelo desenvolvedor da solução requerida. A verificação da complexidade de tempo (e não menos importante, da complexidade de espaço) de um algoritmo configura-se como uma ferramenta poderosa no estudo comparativo entre duas ou mais soluções para um mesmo problema. Além disso, da mesma forma que se pode estabelecer níveis de complexidade de algoritmos para um dado problema, pode-se estabelecer níveis de complexidades de problemas, de tal forma que seja possível estabelecer uma hierarquia de complexidade dos problemas segundo a complexidade dos mesmos.

Objetivo:

O principal objetivo desta atividade avaliativa é levar o aluno a vivenciar, na prática, os conceitos e os métodos envolvidos no processo de análise e síntese de algoritmos. Além disso, objetiva-se que o aluno reforce os conhecimentos sobre desenvolvimento de programas, levando-o a realizar experimentação com diferentes tipos e tamanhos de entradas de dados, ao mesmo tempo em que desenvolve a análises dos resultados.

Descrição

Neste trabalho, o grupo deverá realizar um estudo comparativo do comportamento de seis algoritmos de ordenação interna entre si e em relação ao comportamento assintótico dos mesmos. O grupo deve implementar algoritmos não eficientes (*BubbleSort, InsertionSort e SelectionSort*) e eficientes (*QuickSort, HeapSort* e *MergeSort*) para ordenar sequências de dados armazenados em memória interna. Deve ainda apresentar implementações dos algoritmos citados, onde a sequência de chaves a ser ordenada deve 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 será objeto de avaliação.

Cada elemento da sequência a ser ordenada deve conter um campo chave **e pelo menos um campo de informação**. A especificação e desenvolvimento dos tipos abstratos de dados (as *structs* em C) a serem utilizados no trabalho são de responsabilidade do grupo e fazem parte da avaliação. Você pode, por exemplo, utilizar uma estrutura em que, além da chave, tenha nome e salario.

Geração dos dados de entrada para os experimentos:

Considere sequências de dados com a quantidade de elementos variando de 10, 100, 1000, 10000, 100000, 1000000, etc. Considere também sequências sem valores repetidos. E ainda, considere que todos os elementos das sequências correspondem a valores inteiros. Para fins de análise de melhor e de pior caso, gerar as sequências ordenadas, inversamente ordenadas, quase ordenadas e aleatórias.

Análise dos algoritmos:

A análise deve ser feita **sobre o número de comparações, atribuições e tempo de execução** dos algoritmos. Organize os dados coletados em tabelas, e também construa gráficos a partir dos dados. A partir das tabelas e gráficos, desenvolva o texto sobre a análise dos algoritmos. Grande parte da avaliação será dedicada a análise dos resultados, ou seja, sobre o que o grupo escreveu sobre a análise dos resultados. 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, C++ ... (bem endentado e comentado).

. Documentação do trabalho. Entre outras coisas, a documentação deve conter:

a. Introdução: descrição do problema a ser resolvido e visão geral sobre o funcionamento do programa.

b. 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 neste documento. Muito importante: os códigos utilizados nas implementações devem ser inseridos na documentação como anexos.

c. Análise de Complexidade: estudo da complexidade de tempo e espaço das funções implementadas e do programa como um todo (utilize a notação O).

d. Listagem de testes executados: os testes executados devem ser apresentados, analisados e discutidos, quando convier.

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

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

Em LATEX: A documentação deve ser elaborada obrigatoriamente em LATEX. No modelo encaminhado juntamente com esse documento

Formato final: obrigatoriamente em PDF.

Entrega: Cada grupo X deve enviar para o e–mail ssoares@ice.ufjf.br com o assunto "Trabalho 1 – APA 2017 – Grupo X" até o meio dia do dia 24 de abril de 2017.

Referência

[1] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. Algoritmos: Teoria e Prática. Campus, 2002.
[2] Knuth, D. E. The Art of Computer Programing: Sorting and Searching, vol. 3. Addison-Wesley, 1973.
[3] Wirth, N. Algoritmos e Estruturas de Dados. Guanabara Koogan, 1989.

Boa prova!