FCT/Unesp – Presidente Prudente
Projeto e Análise de Algoritmos
Prof. Danilo Medeiros Eler

Trabalho Prático 01
17/09/2019

Instruções de Envio

Enviar para **daniloelerunesp@gmail.com** (por favor, enviar **somente** para esse email)
No assunto do email você deve colocar: [PAA2019] Trabalho Prático 01.
No corpo do email você deve identificar o seu grupo, colocando o nome de cada integrante. Anexar o relatório com suas análises, código fonte e, se houver, os dados utilizados pelos algoritmos (no caso de ele estar lendo de arquivo).

Data máxima para envio

O trabalho deve ser enviado por email até o dia **27/10/2019**.

Especificações do trabalho prático

O trabalho pode ser desenvolvido em grupos de no máximo 3 pessoas.

Implementar os algoritmos de ordenação e fazer uma análise experimental, calculando o tempo de execução dos algoritmos. Em seguida, deve ser relatada a análise assintótica, comparando os resultados obtidos com a análise experimental. Você deve mostrar a análise assintótica (somente a complexidade de tempo) do melhor caso, caso médio e pior caso. Essa análise pode ser obtida do material de aula ou de livros.

Considere três tipos de entrada de dados, com o objetivo de identificar melhor caso, pior caso e caso médio dos algoritmos. A primeira entrada considera números gerados de forma aleatória; a segunda considera números gerados em ordem crescente; a terceira considera números gerados em ordem decrescente.

Execute os algoritmos de ordenação com diferentes tamanhos de entrada, isto é, vetores com diferentes tamanhos. Exemplos de tamanho de vetor (n): 1000, 5000, 10000, 15000, 20000, 25000,.....
Armazene o tempo gasto em cada execução e gere um gráfico para comparar o desempenho do algoritmo com os diferentes tipos de entrada e depois para comparar todos os algoritmos, com os diferentes tipos de entrada.

Se o tempo de execução for muito pequeno, você deve rodar os algoritmos mais de uma vez; lembrando que isso deve ser padrão para todas as execuções. Por exemplo, se você executar o algoritmo *quick sort* 10 vezes seguidas para computar o tempo dessas

execuções, o algoritmo *bubble sort* também deverá ser executado 10 vezes seguidas, bem como todos os outros.

O trabalho pode ser desenvolvido em qualquer linguagem e os gráficos podem ser feitos por qualquer software, isto é, não é necessário que o programa gere os gráficos.

A nota principal será baseada no relatório (gráficos e discussões). Além disso, será verificado ser os algoritmos estão corretos, consequentemente, a sua análise.

Os algoritmos que deverão ser implementados são:

Bubble Sort (versão original sem melhorias)
Bubble Sort melhorado (verifica se o vetor já está ordenado)
Quick Sort (com pivô sendo o primeiro elemento da lista – partição)
Quick Sort (com pivô sendo o elemento central da lista – partição)
Insertion Sort (inserção simples ou inserção direta)
Shell Sort
Selection Sort (Seleção Direta)
Heap Sort
Merge Sort