Trabalho 1 – Algoritmos de Ordenação

1. Objetivo

O objetivo deste trabalho é implementar os seguintes algoritmos de ordenação, comparando-os entre si:

- Bubblesort
- Insertionsort
- Selectionsort
- Quicksort
- Mergesort

Para cada um dos algoritmos, você deverá medir o tempo de ordenação de um vetor. Este vetor será de tamanhos incrementais (1000, 5000, 10000, 15000, 20000…) até o valor máximo de 100000.

Para cada algoritmo de ordenação (e consequentemente para cada tamanho de vetor), deverá ser utilizado 3 tipos de entrada: crescente, decrescente e aleatória.

Para os vetores aleatórios, repita os testes pelo menos 10 vezes, de forma a obter médias do tempo de execução, Esta repetição **não é necessária** para as entradas crescente e decrescente.

2. Implementação

O arquivo “main.c” anexado a descrição do trabalho exemplifica uma execução do Bubblesort de N(1000) até N(100000), salvando os resultados em um arquivo que pode ser facilmente aberto como uma planilha. Sugiro utilizar o mesmo para análise dos algoritmos e modificar para as entradas crescentes e decrescentes.

O arquivo “funcoes.h” possui o protótipo das funções que deverão ser implementadas para o trabalho. Caso necessário, o aluno pode incluir funções adicionais às exigidas (documentar a necessidade).

O arquivo “funcoes.c”, ATUALMENTE EM BRANCO, é o arquivo onde deverá ser criado as funções pedidas.

Lembrando, para compilar um programa com múltiplos arquivos:

gcc main.c funcoes.c -o main

3. O que deverá ser entregue

• Código fonte do programa em C (identado e comentado).

• 5 planilhas (1 para cada algoritmo), cada uma com 3 abas, contendo os valores das execuções dos algoritmos com entrada crescente, decrescente e aleatória.

• Documentação do Trabalho, que deve conter:

1. Descrição do funcionamento dos algoritmos (Atenção: descrições copiadas da internet terão valor NULO).

2. Gráfico de LINHAS com as comparações do próprio algoritmo Ex: comparação do Bubblesort com Entrada Crescente X Entrada Decrescente X Média das Entradas Aleatórias

3. Gráfico de LINHAS comparando a média dos casos aleatórios de todos os algoritmos. Ex: Comparação do Bubblesort X Insertionsort X Quicksort

4. Considerações

• O trabalho poderá ser feito individualmente ou em dupla. Os dois integrantes da dupla devem saber o trabalho por completo, caso contrário haverá redução de nota.

• O tempo de execução gasto pelo algoritmo deverá ser determinado a partir da chamada à função de ordenação até o seu término (ver “main.c”);

• Uma execução completa de um dos algoritmos tende a levar um tempo considerável. Por exemplo, a execução crescente, decrescente e as 10 execuções aleatórias do Bubblesort para os 21 tamanhos requisitados do trabalho leva cerca de 60 minutos. Portanto, recomenda-se testar os algoritmos inicialmente com os casos pequenos e depois deixar rodando no computador em um momento que não esteja usando para evitar interferência nos resultados.

• O algoritmo Quicksort utiliza um método de particionamento que, escolhido um elemento pivô, esse método gera uma partição de elementos cuja chave seja maior que a chave do pivô e outra partição cujos elementos possuem a chave menores ou iguais a chave do pivô. Existem várias maneiras da escolha deste pivô, mas neste trabalho será utilizado como pivô sempre o primeiro elemento da partição.

• **Data de entrega**: **09 de Junho.** Os trabalhos deverão ser entregues durante a aula.