Trabalho 1 - Ordenação - CIX56

André Ricardo Abed Grégio

2º Semestre - 2023

1 Descrição

O objetivo do trabalho é implementar as versões **recursivas** dos seguintes algoritmos utilizando a linguagem C: *Busca Sequencial*, *Busca Binária*, *Insertion Sort*, *Selection Sort*, *Merge Sort*, *Quick Sort* e *Heap Sort*.

Junto com a implementação, deve ser entregue um relatório em PDF com experimentos relatando:

1. Quantidade de comparações feitas entre elementos do vetor;
2. Tempo de execução em segundos.

A definição dos testes a serem feitos e discutidos no relatório ficam a cargo do aluno.

2 Template

A implementação deve usar o template fornecido pelo professor.

O arquivo *ordenacao.h* contém os protótipos (assinaturas) das funções que obrigatoriamente devem ser implementadas, no arquivo *ordenacao.c*.

Não é permitida mudança nos protótipos de funções disponibilizados no arquivo *ordenacao.h*. É permitida a criação de funções auxiliares se necessário. Também é permitida a criação de arquivos de header (.h) e de implementação (.c) complementares se necessário.

É obrigatória a inclusão de um arquivo *main.c* com alguns testes nos algoritmos implementados. **A definição dos testes fica a cargo do aluno**.

O arquivo deve compilar sem erros ou avisos através do comando *make*.

O nome do binário deve ser *trab*. Ajuste isso no *makefile*. O binário deve executar em sistemas Linux executando o programa sem a passagem de nenhum parâmetro (e.g., *./trab*).

3 Arquivos a serem entregues

Devem ser entregues:

- arquivos de código fonte .c e .h do programa;
- makefile;
- relatório .pdf.

Não inclua arquivos irrelevantes para a compilação do projeto (por exemplo, binários compilados ou arquivos objeto .o).

4 Relatório

Realize testes, e discorra sobre esses testes no relatório. Relacione, por exemplo, o tempo de execução em seu computador para diferentes algoritmos, e também o número de comparações realizadas pelos algoritmos para diferentes tamanhos de entradas.

Você pode incluir gráficos ou tabelas no relatório para torná-lo mais informativo e interessante.

O relatório deve ter no máximo duas páginas se utilizado espaçamento simples e coluna dupla, ou no máximo três páginas para espaçamento 1,5 ou duplo e formato de uma coluna.

Veja um exemplo de como escrever um relatório em <https://pt.overleaf.com/read/kfzrvbppnpth>.

5 Entrega

O trabalho deve ser feito individualmente.

Os arquivos devem ser empacotados em um arquivo `grr.tar.gz`, onde `grr` é o grr do aluno. Ao descompactar este arquivo deverá ser criado um diretório de nome `grr` que conterá todos os demais arquivos.

O trabalho deve ser entregue via Moodle. A data limite para o envio está estipulada no link de entrega do Moodle.

Não serão aceitas entregas em atraso, exceto para os casos explicitamente amparados pelas resoluções da UFPR.

6 Dicas

6.1 Contando o tempo

Para contar o tempo gasto por determinada função em C, inclua a biblioteca *time.h*. Veja a seguir um exemplo para contar o tempo em segundos necessário para se executar uma função.

```
#include <time.h>
```

```
//...
clock_t start, end;//variáveis do tipo clock_t
double total;

start = clock();//start recebe o "ciclo" corrente
minhaFuncao();//chama a função que desejamos medir o tempo
end = clock();//end recebe o "ciclo" corrente
//o tempo total é a diferença dividida pelos ciclos por segundo
total = ((double)end - start)/CLOCKS_PER_SEC;
//total agora possui o tempo em segundos
printf("Tempo total: %f", total);
```

6.2 Função clock

A especificação em C da função clock()[1] é clara em dizer que a função deve retornar o tempo de CPU (*CPU time*)[2], mas algumas fontes citam que o Windows ignora a especificação e conta o tempo de parede (*Wall clock time*)[3]. Para simplificar, será aceito tanto o *CPU time* quanto o *Wall Clock time*, mas deixe claro em seu relatório qual sistema operacional você usou.

Para tornar os testes mais confiáveis, execute os programas com "tudo fechado" em sua máquina. Por exemplo, não abra o navegador web enquanto executar os testes, já que o navegador pode competir por recursos, e gerar diferenças nos tempos, especialmente para o *wall clock time*.

7 Distribuição da Nota

A nota do trabalho é composta por 70% referente a completude e qualidade de implementação, e 30% referente ao relatório. Esses valores podem variar para casos específicos. Por exemplo, caso um aluno entregue apenas o relatório e não a implementação, o relatório não terá direito aos 30% de nota.

Alguns descontos padrão, considerando uma nota entre 0 e 100 pontos para o trabalho:

- Plágio: perda total da pontuação para todos os envolvidos. Isso é válido mesmo para casos onde o plágio se refere a apenas um trecho do código.
- Algoritmos não recursivos, ou implementados em linguagem diferente de C, serão desconsiderados.
- Falta de algum arquivo requisitado: desconto de 10 a 100 pontos.
- Inclusão de arquivos desnecessários (lixo): desconto de 5 a 20 pontos.
- Erros e avisos de compilação: desconto de 5 a 100 pontos.
- Nomes de arquivo incorretos: 10 pontos por arquivo.
- Arquivo com formato incorreto: 10 pontos por arquivo.
- Vazamentos de memória: 10 a 50 pontos.

[1] Ver Seção 7.23.2.1 de <http://www.open-std.org/jtc1/sc22/WG14/www/docs/n1256.pdf>
[2] <https://en.wikipedia.org/wiki/CPU_time>
[3] <https://en.wikipedia.org/wiki/Elapsed_real_time>

8 Demais Regras

- Dúvidas ou casos não especificados neste documento podem ser discutidos com o professor até a **data de entrega do trabalho**.
- Os trabalhos não serão aceitos após a data/hora limite.