Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciências da Computação

Professores: Luiz Chaimowicz e Raquel Prates
Monitores: Alysson R. da Silva e Tiago Alves
Estruturas de Dados 2021.2
Belo Horizonte, 26 de Julho de 2021

Trabalho Prático II: Otimização dos Servidores

Trabalho Individual. Valor: 10 pontos **Entrega: 15 de Agosto de 2021**

1 Introdução

Através dos serviços da Rellocator CO., no ano de 2076, uma pessoa pode fazer upload de sua mente para corpos sintéticos. No entanto, um dos grandes problemas era o fato de que 27% das consciências que faziam parte de um processo de upload eram roubadas por hackers. Dessa forma, a Rellocator CO. contratou uma equipe para desenvolver vários protótipos de um sistema de controle e detecção de anomalias no fluxo dos dados referentes as consciências a serem transferidas através de diversos servidores. Porém, houve uma redução de apenas 17% nos roubos das mesmas durante um processo de transferência. Dessa forma, o time de cientistas da Rellocator CO. deseja investigar outra medida de segurança para prevenir futuros roubos.

A nova medida de segurança visa ordenar as consciências em uma configuração para que os servidores possam trabalhar de forma mais otimizada. Dessa forma, o tempo de envio das consciências será reduzido. Ao reduzir o tempo de envio, espera-se que também seja reduzido o tempo de visibilidade das mesmas por hackers e reduzir ainda mais o número de roubos.

Dentre as características dos dados a serem ordenados, deve-se notar o fato de que os mesmos são compostos por dois campos. O primeiro campo representa o nome pessoa/consciência a ser realocada e o segundo campo representa a consciência a ser transferida. Por esse motivo, a Rellocator CO. gostaria de investigar duas hipóteses relativas a estabilidade e tempo de execução ao aplicar determinados métodos para organizar e otimizar as informações.

A primeira hipótese a ser investigada pela Rellocator CO. pretende verificar se os campos realmente irão manter a configuração desejada após ordenados utilizando algoritmos estáveis. Já a segunda hipótese, visa avaliar se o tempo de execução dos métodos de ordenação irão causar impacto no desempenho dos servidores de envio.

Para conseguir responder as duas hipóteses, a Rellocator CO. contratou vários programadores (incluindo você). Consequentemente, seu objetivo neste trabalho é implementar os métodos de ordenação e seguir uma metodologia de testes rigorosa estabelecida pela Rellocator CO.. Para facilitar o seu trabalho, a companhia disponibilizou dados utilizados apenas para homologação que são provenientes de um ambiente de simulação. Por fim, você deve apresentar todos os resultados (sem alteração) como especificado pela Rellocator CO. e entregar o seu parecer técnico.

2 Especificações

Você deve ordenar o conjunto de informações disponibilizados. As informações são compostas por dois campos, '`NOME DADOS`', onde '`NOME`' é o nome da pessoa e '`DADOS`' é o dado em binário relacionado à mesma. A ordenação deve ser feita em duas etapas. A primeira etapa ordena as informações considerando o campo '`DADOS`'. Posteriormente, com os dados já ordenados, deve-se ordená-los novamente considerando o campo '`NOME`'. Ao fim da ordenação, nomes iguais deverão estar agrupados e ordenados. Além disso, os binários de cada agrupamento também deverão estar ordenados.

2.1 Configuração Experimental

Dentre as possíveis abordagens para ordenar as informações em cada campo, foram considerados os algoritmos descritos na Tabela 1,

Campo	Algoritmos	
NOME	Quicksort	Mergesort
DADOS	Heapsort	Radix Exchange sort

Tabela 1: Algoritmos considerados viáveis para a ordenação.

com o intuito de avaliar o desempenho de qual combinação dos mesmos é mais apropriada em termos de **Tempo de Execução** e **Estabilidade**. Foram propostas quatro configurações de teste que são apresentadas na Tabela 2.

Configuração	NOME	DADOS
1	Quicksort	Heapsort
2	Quicksort	Radix Exchange sort
3	Mergesort	Heapsort
4	Mergesort	Radix Exchange sort

Tabela 2: Combinação de algoritmos que deve ser utilizada durante a avaliação.

Além das configurações para avaliação descritas na Tabela 2, também foi criado um arquivo para homologação de qual configuração será utilizada no sistema real. O arquivo utilizado para avaliar as configurações contém 200 mil linhas. A avaliação das configurações descritas na Tabela 2, deve ser feita variando o tamanho **N** da entrada deste arquivo. O valor de **N** é a quantidade de linhas a serem lidas e consideradas durante as ordenações. A quantidade de linhas lidas, ou a variável **N**, deve seguir a seguinte sequência,

- 1.000, 10.000, 50.000, 100.000 e 200.000.

ou seja, você deve testar **CADA UMA** das configurações de algoritmos da Tabela 2, lendo inicialmente os 1.000 (mil) primeiros elementos; Depois, em um outro teste, os 10.000 (dez mil) primeiros elementos incluindo os anteriores; depois, em mais um teste os 50.000 (cinquenta mil) primeiros elementos, e assim por diante, até que se alcance N = 200.000 (duzentos mil).

3 Apresentação dos Resultados

É obrigatório que os resultados obtidos após a execução dos experimentos descritos na Seção 2.1 sejam apresentados em forma de 1 (um) gráfico de barras simples (Column Chart em inglês), onde o eixo das abscissas (eixo x) representa o tamanho da entrada e o eixo das ordenadas (eixo y) representa o tempo de execução em segundos. O gráfico de barras deve conter os tempos de execução para todas as configurações da Tabela 2, para cada tamanho '`N`' do arquivo de entrada. Deve-se agrupar cada configuração com o respectivo tamanho de entrada '`N`'. A Figura 1 apresenta um exemplo para o padrão que **DEVE** ser seguido.

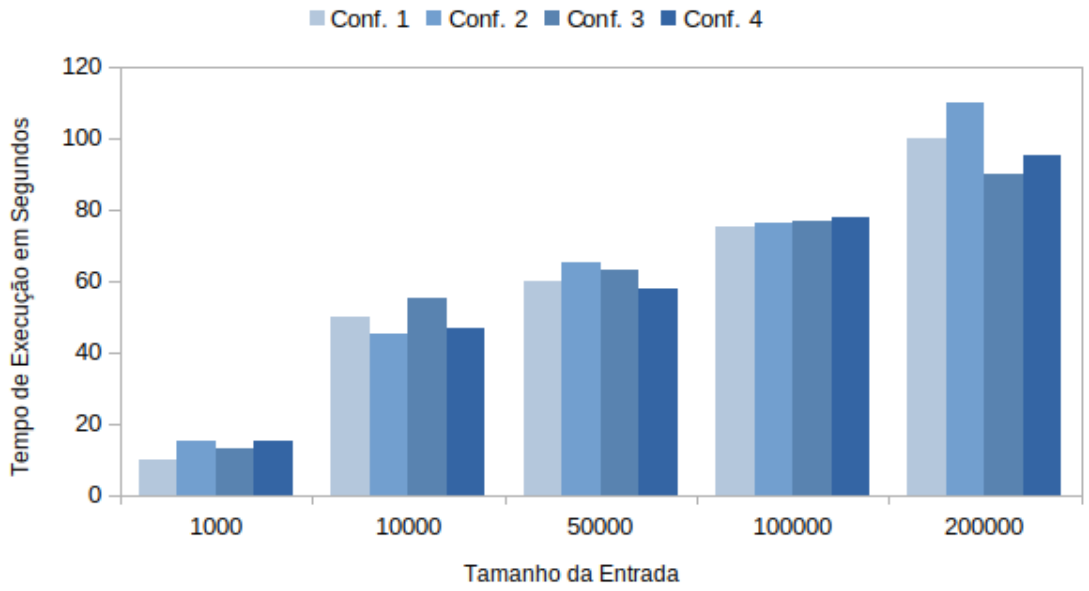

Figura 1: Ilustração do padrão que deve ser seguido para apresentação dos resultados obtidos.

Além desse gráfico **obrigatório**, você deve apresentar quaisquer outros gráficos que representem os resultados obtidos através da execução de experimentos sugeridos por você. A escolha de possíveis novos experimentos é livre e não há restrições.

Deve ser feita também uma análise acerca da estabilidade das configurações sugeridas, deixando claro quais os algoritmos em cada uma das configurações que não são estáveis, se esse for o caso.

4 Execução, Entradas e Saídas

O executável do seu programa deve receber três argumentos de linha de comando. O primeiro argumento representa o arquivo para homologação. Já o segundo argumento, representa qual configuração de teste deve ser executada. O terceiro argumento representa a quantidade de linhas a serem consideradas durante o teste. Por exemplo, considere a seguinte chamada ao seu programa através de um terminal,

```
run.out homologacao.txt 1 50000
```

onde, '`run.out`' é o executável do seu programa, '`homologacao.txt`' é o arquivo que contém as 200 mil linhas para avaliação das configurações descritas na Tabela 2, o argumento '`1`' é a configuração a ser executada (Quicksort para o campo '`NOME`' e Heapsort para o campo '`DADOS`'), e '`50000`' representa a quantidade de linhas a serem avaliadas.

4.1 Arquivo de entrada

O arquivo de entrada contém 200 mil linhas, onde cada linha é composta pelos campos '`NOME`' e '`DADOS`' separados por um espaço em branco. Por exemplo, a Entrada 1 ilustra as 13 primeiras linhas de um exemplo de arquivo de entrada, onde a primeira linha contém o campo '`NOME`' = '`MARCOS`' e o campo '`DADOS`' = '`01010101`'.

```
MARCOS 01010101
CHAIMOWICZ 01011011
LUIZ 00001111
RAQUEL 10110101
LUIZ 11001100
MEIRA 01111001
PRATES 01010001
CHAIMO 10101010
LUIZ 11010011
PRATES 11110000
MANUEL 01101001
LUIZ 00110011
RAQUEL 00001111
```

Entrada 1: Exemplo de arquivo de entrada que contém as informações à serem ordenadas.

4.2 Saída

Utilize a **saída padrão** para imprimir todas as informações. Após ordenar os elementos, o seu programa deve imprimir na saída as informações ordenadas conforme o que foi especificado na Seção 2. As informações devem ser impressas uma por linha seguindo o padrão '`NOME DADOS`'. A seguir é apresentado um exemplo de saída válida para entrada apresentada na Seção 4.

```
CHAIMO 10101010
CHAIMOWICZ 01011011
LUIZ 00001111
LUIZ 00110011
LUIZ 11001100
LUIZ 11010011
MANUEL 01101001
MARCOS 01010101
MEIRA 01111001
PRATES 01010001
PRATES 11110000
RAQUEL 00001111
RAQUEL 10110101
```

Saída 1: Exemplo de saída para a entrada apresentada na Seção 4.

5 Entregáveis

Você deve utilizar a linguagem C ou C++ para o desenvolvimento do seu sistema. O uso de estruturas pré-implementadas pelas bibliotecas-padrão da linguagem ou terceiros é **terminantemente vetado**. Caso seja necessário, use as estruturas feitas no Trabalho Prático anterior para criar **suas próprias implementações** para todas as classes, estruturas, e algoritmos.

Você **DEVE utilizar** a estrutura de projeto abaixo junto ao '`Makefile`' disponibilizado no *Moodle*:

```
- TP
    |- src
    |- bin
    |- obj
    |- include
    Makefile
```

A pasta '`TP`' é a raiz do projeto; a pasta '`bin`' deve estar vazia; '`src`' deve armazenar arquivos de código ('`*.c`', '`*.cpp`', ou '`*.cc`'); a pasta '`include`', os cabeçalhos (*headers*) do projeto, com extensão '`*.h`', por fim a pasta '`obj`' deve estar vazia. O **Makefile** disponibilizado no **Moodle** deve estar na **raiz do projeto**.

5.1 Documentação

A documentação do trabalho deve ser entregue em formato **pdf** e também **DEVE** seguir o modelo de relatório que será postado no Moodle. Além disso, a documentação deve conter **TODOS** os itens descritos a seguir **NA ORDEM** em que são descritos,

- Título, nome, e matrícula.
- **Introdução:** Contém a apresentação do contexto, problema, e qual solução será avaliada.
- **Método:** Descrição da implementação que detalhe as estruturas, classes e métodos implementados. Para cada método de ordenação, você deve apresentar uma descrição, com suas palavras, em alto nível, explicando o método e decisões tomadas relativas à sua implementação.
- **Análise de Complexidade:** Contém a análise da complexidade de tempo e espaço dos procedimentos implementados, formalizada pela notação assintótica.
- **Configuração Experimental:** Descrição de quais experimentos foram realizados explicitando o objetivo de cada um.
- **Resultados:** Contém a apresentação dos resultados. Cada gráfico apresentado deve ser precedido por um parágrafo explicitando os valores máximos, mínimos, e médios de cada experimento. A análise relacionada à estabilidade das configurações deve estar presente aqui.
- **Conclusões:** A Conclusão deve conter uma frase inicial sobre o que foi feito no trabalho. Posteriormente deve-se sumarizar os resultados obtidos. Por fim, deve-se apresentar a conclusão técnica sobre o que foi observado para a sua implementação.
- **Bibliografia:** Contém fontes utilizadas para realização do trabalho. A citação deve estar em formato científico apropriado que deve ser escolhido por você.
- **Instruções para compilação e execução**: Esta seção deve ser colocada em um apêndice ao fim do documento e em uma página exclusiva (não divide página com outras seções).

5.2 Submissão

Todos os arquivos relacionados ao trabalho devem ser submetidos na atividade designada para tal no *Moodle*. A entrega deve ser feita **em um único arquivo** com extensão **.zip**, com nomenclatura '`nome_sobrenome_matricula.zip`', onde '`nome`', '`sobrenome`', e '`matricula`' devem ser substituídos por suas informações pessoais. O arquivo **.zip** deve conter a sua documentação no formato **.pdf** e a estrutura de projeto descrita na Seção 5.

6 Avaliação

O trabalho será avaliado em 10 pontos com a seguinte distribuição:

1. Execução correta dos casos de teste - **20% da nota total**
2. Apresentação da análise de complexidade das implementações - **15% da nota total**
3. Estrutura e conteúdo da documentação e apresentação dos resultados - **50% da nota total**
4. Indentação do código fonte - **5% da nota total**
5. Comentários no código fonte em forma de documentação em cima da assinatura de **TODAS** as funções criadas - **5% da nota total**
6. Cumprimento total da especificação - **5% da nota total**

Se o programa submetido não compilar[1], seu trabalho não será avaliado e sua nota será 0. Trabalhos entregues com atrasos sofrerão penalização de $2^d - 1$ pontos, com $d =$ dias de atraso.

7 Considerações Finais

1. Leia **atentamente** o documento de especificação, pois o descumprimento de quaisquer requisitos obrigatórios aqui descritos causará penalizações na nota final.
2. Certifique-se de garantir que seu arquivo foi submetido corretamente no sistema.
3. **Plágio é CRIME**. Trabalho onde o plágio for identificado serão **automaticamente anulados** e as medidas administrativas cabíveis serão tomadas. Discussões a respeito do trabalho entre colegas são permitidas. É permitido consultar fontes externas, desde que exclusivamente para fins didáticos e devidamente registradas na sessão de bibliografia da documentação. **Cópia e compartilhamento de código não são permitidos**.

[1]Entende-se por compilar aquele programa que, independente de erros no Makefile ou relacionados a problemas na configuração do ambiente, funcione e atenda aos requisitos especificados neste documento em um ambiente Linux.

8 FaQ (Frequently asked Questions)

1. Posso utilizar o tipo String? **SIM.**
2. Posso utilizar o tipo String para simular minhas estruturas de dados? **NÃO.**
3. Posso utilizar alguma biblioteca para tratar exceções? **SIM.**
4. Posso utilizar alguma biblioteca para gerenciar memória? **SIM.**
5. As análises a apresentação dos resultados são importantes na documentação? **SIM.**
6. Os meus princípios de programação ligados a C++ e relacionados a engenharia de software serão avaliados? **NÃO.**
7. Com relação ao código fonte, será avaliado apenas o que foi explicitado na Seção 6, incluindo Indentação de código e comentário para documentação das funções? **SIM.**
8. Posso utilizar alguma estrutura de dados do tipo Queue, Stack, Vector, List, e **etc**..., do C++? **NÃO.**
9. Posso fazer o trabalho em dupla ou em grupo? **NÃO.**
10. Posso trocar informações com os colegas sobre a teoria? **SIM.**
11. Posso fazer o trabalho no Windows, Linux, ou MacOS? **SIM.**
12. Posso utilizar IDEs, Visual Studio, Code Blocks, Visual Code, Eclipse? **SIM.**