Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
DCC205 - Estrutura de Dados

Trabalho Prático 3 – Servidor de emails otimizado

Valor: 15 pontos

Data de entrega: 19/07/2022

Objetivo

O objetivo deste trabalho é a implementação do simulador de um servidor de emails. No entanto, além do gerenciamento adequado da memória do sistema, agora também há foco na otimização da pesquisa por usuários e mensagens. O sistema simulado terá suporte à entrega, consulta e remoção de mensagens para usuários. O foco da simulação é verificar o funcionamento correto do sistema ao executar as diversas operações do servidor (descritas a seguir) em diferentes situações. A análise de complexidade das funções implementadas será avaliada na documentação, e será necessário fazer uma comparação descrevendo as vantagens e desvantagens (se houver) de se utilizar métodos de pesquisa e ordenação mais sofisticados.

Descrição

Suponhamos que você tenha desenvolvido um novo servidor de emails para a Google, pois o servidor anterior possuía algumas falhas e consumo exagerado de memória. A Google gostou muito de suas habilidades como desenvolvedor e quer agora que você melhore ainda mais o servidor!

A Google quer que o servidor dê suporte à entrega e consulta de mensagens para usuários cadastrados de maneira eficiente. Para isso, deverá ser utilizada uma **tabela hash** para armazenar as caixas de mensagens dos usuários. Cada entrada da tabela hash apontará para uma **árvore binária** que armazena os emails.

A seguir estão descritas as operações que o servidor deverá suportar:

Entregar email: Esta operação recebe um novo email destinado a determinado usuário. O ID do usuário é utilizado para determinar em qual posição da tabela hash a mensagem para ele deve ser armazenada. Uma vez determinada a posição, o email é armazenado na árvore binária correspondente, utilizando-se o ID do email como chave.

Consultar email: Esta operação recebe o identificador de um usuário e de uma mensagem. Ela deve buscar a entrada da tabela hash que corresponde ao usuário e, em seguida, recuperar a mensagem na árvore binária correspondente.

Apagar email: Esta operação recebe o identificador de um usuário e de uma mensagem. Ela deve buscar a mensagem na árvore de emails do usuário e, caso um email com o identificador fornecido exista, removê-lo.

O que deve ser feito?

Você deverá implementar um simulador do sistema descrito acima. O simulador deverá suportar as operações de entrega, consulta e remoção de mensagens. Como especificado, a principal estrutura do servidor de emails é uma **tabela hash**. Cada entrada desta tabela aponta para uma **árvore binária**, e cada nó desta árvore armazena um email.

Dado o ID de um usuário, todas as mensagens enviadas para ele devem ser armazenadas na **mesma árvore binária**. Seja M o tamanho da tabela hash e U o identificador de um determinado usuário. A árvore binária onde todas as mensagens para o usuário U devem ser armazenadas é dada pela fórmula:

$$b = U \,\%\, M$$

Ou seja, o resto da divisão inteira de U por M fornece o índice b da entrada da tabela hash que aponta para a árvore binária procurada ($0 \leq b < M$). Uma vez encontrada a árvore, mensagens para o usuário U devem ser sempre armazenadas ou procuradas nela. Seja E o identificador de um email para U. A mensagem deve ser armazenada ou buscada na árvore binária de U segundo sua chave E.

Dois ou mais usuários podem ser mapeados para a mesma árvore binária. Por exemplo, suponha que M = 4. Nesse caso, os usuários com IDs iguais a 3, 7 e 11 são todos mapeados para a árvore binária apontada pela entrada $b = 3$ da tabela hash. As mensagens de **todos** esses usuários devem ser armazenadas nesta mesma árvore binária. No entanto, duas mensagens, sejam para o mesmo usuário ou para usuários diferentes, **nunca** terão o mesmo identificador E e, dessa forma, será sempre possível armazená-las ou buscá-las na árvore binária.

Para cada operação processada, será gerada uma saída. Cada operação pode gerar um conjunto definido de saídas, especificado abaixo. O formato da saída deve corresponder **exatamente** ao formato especificado, pois será através dela que a correção do simulador será feita.

A entrada do programa de simulação será um arquivo de texto contendo, em cada linha, uma operação. A simulação deve terminar quando o final do arquivo for alcançado. Exceto pela primeira linha do arquivo de entrada, que contém um inteiro M (o tamanho a ser utilizado para a tabela

hash), cada linha contém uma operação a ser simulada. Para cada operação, deverá ser impressa uma linha na saída padrão (na tela), conforme especificado abaixo. A seguir, estão especificadas as operações, conforme presentes no arquivo de entrada:

ENTREGA U E N MSG

- Operação de solicitação de entrega de uma nova mensagem. O parâmetro U é o identificador do destinatário da mensagem, um número inteiro tal que $0 \leq U \leq 10^6$. O valor de U deve ser utilizado para calcular a entrada da tabela hash que aponta para a árvore binária na qual a mensagem deve ser armazenada, conforme já explicado.

- O parâmetro E é o identificador do email, um número inteiro tal que $0 \leq E \leq 10^6$. O valor de E deve ser utilizado como chave para armazenar a mensagem na posição adequada da árvore binária, de tal forma que ela possa ser pesquisada depois.

- O parâmetro N indica o número de palavras da mensagem, um número inteiro tal que $0 \leq N \leq 200$. Cada palavra da mensagem pode ser lida individualmente, e toda palavra terá ao menos 1 e no máximo 40 caracteres. Eventuais caracteres de pontuação são parte da palavra adjacente, ou seja, espaço em branco é o único separador de palavras.

- MSG é a mensagem a ser recebida: um conjunto de N palavras separadas por espaço em branco.

- Exemplo: ENTREGA 5 103 6 Bom dia, meu amigo! Tudo bom?

- Saída esperada (U, E e b devem ser substituídos pelos valores correspondentes):
  OK: MENSAGEM 103 PARA 5 ARMAZENADA EM 23

CONSULTA U E

- Operação de consulta de email. Os parâmetros U e E, já explicados, são, respectivamente, os identificadores do usuário e da mensagem para os quais a consulta foi feita.

- Esta operação deve buscar, na caixa de entrada do usuário U, a mensagem E. Mais uma vez, o valor de U deve ser utilizado para encontrar a árvore onde são armazenados os emails para o usuário, e o valor de E deve ser utilizado como chave para buscar a mensagem na árvore binária correspondente.

- CONSULTA U E: MSG - imprime a primeira mensagem de chave E encontrada na caixa de entrada do usuário U.

- Exemplo: CONSULTA 5 103

- Saída esperada (U e E devem ser substituídos pelos valores correspondentes):
  Por exemplo, para o usuário do exemplo anterior:

  CONSULTA 5 103: Bom dia, meu amigo! Tudo bom?

  Já para um caso onde a mensagem não esteja armazenada a saída seria:

  CONSULTA U E: MENSAGEM INEXISTENTE

APAGA U E

- Apaga um email. Os parâmetros U e E, já explicados, são, respectivamente, os identificadores do usuário e da mensagem para os quais a consulta foi feita.

- Esta operação deve buscar, na caixa de entrada do usuário U, a mensagem E. Caso seja encontrada, a mensagem (o nó da árvore correspondente ao email) deve ser removida. Mais uma vez, o valor de U deve ser utilizado para encontrar a árvore onde são armazenados os emails para o usuário, e o valor de E deve ser utilizado como chave para buscar a mensagem na árvore binária correspondente.

- Exemplo: APAGA 5 103

- Saída esperada (U e E devem ser substituídos pelos valores correspondentes):

  Caso a mensagem seja encontrada e devidamente removida.
  OK: MENSAGEM APAGADA

  Caso uma mensagem com chave E não seja encontrada na caixa de entrada do usuário U.
  ERRO: MENSAGEM INEXISTENTE

Quais TADs implementar?

Você deverá implementar pelo menos três Tipos Abstratos de Dados (TADs): o primeiro para o email (ou mensagem), o segundo para a caixa de entrada (árvore binária) e o terceiro para o servidor de emails (tabela hash). A *função hash* apresentada deve ser a utilizada pelo servidor de emails. **O tamanho M da tabela hash é um parâmetro do programa e será fornecido na primeira linha do arquivo de entrada.**

Parâmetros

O seu programa deve aceitar dois parâmetros:

-i: informa o nome do arquivo de entrada
-o: informa o nome do arquivo de saída

Exemplo:

```
bin/tp3 -i entrada.txt -o saida.txt
```

Avaliação Experimental

Você deve considerar pelo menos as seguintes dimensões na sua avaliação experimental:

- Número de usuários
- Número de mensagens
- Tamanho das mensagens
- Distribuição de frequência de operações

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 que **você** implementou nos Trabalhos Práticos anteriores para criar **suas próprias implementações** para todas as classes, estruturas de dados, e algoritmos.

Você **DEVE utilizar** a estrutura de projeto abaixo junto ao *Makefile* :

```
- 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** deve estar na **raiz do projeto**. A execução do **Makefile** deve gerar os códigos objeto *.o no diretório **obj,** e o executável do TP no diretório **bin**.

Documentação

A documentação do trabalho deve ser entregue em formato **pdf**. A documentação deve conter os itens descritos a seguir :

Título, nome, e matrícula.
Introdução: Contém a apresentação do contexto, problema, e qual solução será empregada.
Método: Descrição da implementação, detalhando as estruturas de dados, tipos abstratos de dados (ou classes) e funções (ou métodos) implementados.
Análise de Complexidade: Contém a análise da complexidade de tempo e espaço dos procedimentos implementados, formalizada pela notação assintótica.
Estratégias de Robustez: Contém a descrição, justificativa e implementação dos mecanismos de programação defensiva e tolerância a falhas implementados.
Análise Experimental: Apresenta os experimentos realizados em termos de desempenho computacional e localidade de referência, assim como as análises dos resultados.
Conclusões: A Conclusão deve conter uma frase inicial sobre o que foi feito no trabalho. Posteriormente deve-se sumarizar o que foi aprendido.
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).
***Número máximo de páginas:** 20

A documentação deve conter a descrição do seu trabalho em termos funcionais, dando foco nos algoritmos, estruturas de dados e decisões de implementação importantes durante o desenvolvimento.

Evite a descrição literal do código-fonte na documentação do trabalho.

Dica: Sua documentação deve ser clara o suficiente para que uma pessoa (da área de Computação ou não) consiga ler, entender o problema tratado e como foi feita a solução.

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 matrícula devem ser substituídos por suas informações pessoais. O arquivo **.zip** deve conter a sua documentação no formato **.pdf** e a estrutura de projeto descrita.

Avaliação

O trabalho será avaliado de acordo com:

- A Corretude na execução dos casos de teste - (50% da nota total)
- Apresentação da análise de complexidade das implementações - (15% da nota total)
- Estrutura e conteúdo exigidos para a documentação - (20% da nota total)
- Indentação, comentários do código fonte e uso de boas práticas - (10% da nota total)
- 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. Não serão aceitos trabalhos entregues com atraso.

Comentários gerais:

- Comece a fazer esse trabalho prático o quanto antes, enquanto o prazo de entrega está tão distante quanto jamais estará.

- Leia **atentamente** o documento de especificação, pois o descumprimento de quaisquer requisitos obrigatórios aqui descritos causará penalizações na nota final.

- Certifique-se de garantir que seu arquivo foi submetido corretamente no sistema.

- Plágio é CRIME. Trabalho onde o plágio for identificado serão **automaticamente anulados** e as medidas administrativas cabíveis serão tomadas (em relação a todos os envolvidos). 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**.

- Penalização por atraso: considerando o fim do semestre, **não** será permitida a entrega com atraso.

FaQ (Frequently asked Questions)

1. **Posso utilizar alguma estrutura de dados do tipo Queue, Stack, Vector, List, e etc..., do C++? NÃO**
2. Posso utilizar o tipo String? SIM.
3. Posso utilizar o tipo String para simular minhas estruturas de dados? NÃO

[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.

4. Posso utilizar alguma biblioteca para tratar exceções? SIM.
5. Posso utilizar alguma biblioteca para gerenciar memória? SIM.
6. As análises e apresentação dos resultados são importantes na documentação? SIM.
7. Os meus princípios de programação ligados a C++ e relacionados a engenharia de software serão avaliados? NÃO
8. Posso fazer o trabalho em dupla ou em grupo? NÃO
9. Posso trocar informações com os colegas sobre a teoria? SIM.
10. Posso fazer o trabalho no Windows, Linux, ou MacOS? SIM.
11. Posso utilizar IDEs, Visual Studio, Code Blocks, Visual Code, Eclipse? SIM.