Trabalho Prático 0

Operações com Matrizes Alocadas Dinamicamente

Antony Guilherme Costa do Amaral

Universidade Federal de Minas Gerais (UFMG)

Belo Horizonte – MG – Brasil

antonyguilherme@ufmg.br

Matrícula: 2020026796

Introdução

Esta documentação possui como objetivo explicitar o funcionamento do programa que realiza o escalonamento de URL's. Este se trata da divisão e ordenação para o futuro acesso às URL's, com o intuito de economizar processamento e memória. Dito isso, as principais operações são **adicionar**, **escalonar** e **limpar**. Sendo estas avaliadas dinamicamente, ou seja, sem nenhum tamanho pré-determinado.

Implementação

O programa foi escrito utilizando a linguagem C++, compilada pelo compilador G++ da GNU Compiller Collection. Além disso, o programa foi testado em um ambiente Linux (Ubuntu-WSL) tendo disponível 8GB de RAM DDR4 e um processador AMD Ryzen 5.

Estruturas de Dados

A implementação do programa teve como base o tipo abstrato de dados **Escalonador** o qual gerenciava as principais operações do programa. A partir dele são gerenciados também os tipos abstratos de **Host** e **URL** que por sua vez gerenciam as **Listas** e **Filas** do programa. Estas possuem um papel fundamental, visto que armazenam os principais dados do programa. Vale a pena destacar que todos esses tipos foram implementados por meio de classes.

- ### Escalonador

O Escalonador armazena os **Hosts** das **URL's** informadas, e além disso, é responsável por adicionar **URL's** e criar o arquivo de saída.

- ### Host

O **Host** armazena as **URL's** de seu domínio e realiza o controle sobre a ordem na qual elas devem ser acessadas. Ademais, ele também é responsável por armazenar seu domínio.

- **URL**

A **URL** é responsável por armazenar seu endereço e calcular sua profundidade, ou seja, o quão distante do endereço inicial ela está.

- **Lista**

A Lista é um meio de armazenamento dos dados de uma forma sequencial. Para isso, cada item da Lista possui apontadores para o próximo item e o item anterior. Dessa forma é possível navegar pela lista e realizar as operações de exclusão e adição de elementos em qualquer posição.

- **Fila**

A Fila é um meio de armazenamento de dados de uma forma sequencial. Para isso, cada item da Fila possui apontadores para o próximo item e o item anterior. Contudo, adições são feitas no fim da Fila e remoções no início.

Organização do Código

O código é organizado em 5 arquivos principais sendo estes **escalonador.hpp, host.hpp, url.hpp, fila.hpp, e lista.hpp.** Estes são responsáveis por armazenar suas respectivas classes de funcionalidades listadas acima.

Operações Do Escalonador

- **Adicionar URL's (ADD_URLS <quantidade>)**

A operação adicionar URL's é responsável, como o próprio nome já diz, por adicionar as **URL's** ao **Escalonador**. Dito isso, é válido ressaltar que a quantidade de **URL's**, que serão consumidas pelo programa, deverá ser especificada. Ademais, a fonte desses itens deverá ser um arquivo.

- **Escalonar URL's (ESCALONA_TUDO, ESCALONA <quantidade> e ESCALONA_HOST <host>)**

A função de escalonar URL's possui três possibilidades diferentes sendo elas: **Escalonar Todas as URL's , Escalonar por Quantidade e Escalonar por Host.** Com isso, é importante destacar como o processo de escalonamento ocorre. Portanto, este acontece da seguinte maneira: As **URL's** são coletadas e separas por **Hosts** e ordenadas por profundidade, como especificado anteriormente. Após a isso, elas são escalonadas na ordem na qual seus **Host** foram encontrados e sendo que as **URL's** de menor profundidade são acessadas primeiro no contexto de seu **Host**.

- **Ver URL’s do Hosts e Listar Hosts (VER_HOST <host> e LISTA_HOSTS)**

Essas duas operações possuem como respectivas responsabilidades listar as **URL’s** do **Host** informado e listar todos **Hosts** encontrados até aquele momento.

- **Limpa Host e Limpa Tudo (LIMPA_HOST <host> e LIMPA_TUDO)**

Essas duas operações são responsáveis por limpar todas as **URL’s** do **Host** informado e limpar todos os **Hosts** e suas **URL’s**, respectivamente.

Instruções Para Compilação e Execução

Para realizar a compilação basta acessar a raiz do projeto e executar o arquivo **makefile** regras **clean e all**. Ademais, para se executar o programa basta digitar o seguinte comando:

./bin/run.out <nome_do_arquivo.txt>

Outro fator que vale ser destacado, é que todos arquivos considerados devem ficar ou serão gerados na raiz do projeto.

Análise de Complexidade

(n): equivalente ao número de URL’s adicionadas

- **Adição das URL’s**

A função responsável por essa operação é a **adicionarURLs** que depende diretamente da função do **adicionarURLs** da classe **Escalonador.** Esta possui como ordem de complexidade O(n^2) e em consequência disso, a função principal em questão tem como ordem de complexidade O(n^3).

- **Escalonar Todas as URL’s**

A função responsável por essa operação é **escalanoarTodasAsURLs** da classe **Escalonador** a qual possui como ordem de complexidade O(n).

- **Escalonar URL’s por Quantidade**

A função responsável por essa operação é a **escalonarURLs** da classe Escalonador a qual possui estrutura equivalente a **escalanoarTodasAsURLs** e, sendo assim, a ordem de complexidade da função em questão é O(n).

- **Escalonar URL's por Host**

A função responsável por essa operação é a **escalonarURLsDoHost** da classe Escalonador a qual possui como ordem de complexidade O(n).

- **Visualizar Hosts**

A função responsável por essa operação é a **visualizarHosts** da classe Escalonador a qual possui como ordem de complexidade O(n).

- **Visualizar as URL's do Host**

A função responsável por essa operação é a **visualizarURLsDoHost** da classe Escalonador que possui como ordem de complexidade O(n).

- **Limpar Host**

A função responsável por essa operação é a **limparHost** da classe Escalonador a qual possui como ordem de complexidade O(n).

- **Limpar Tudo**

A função responsável por essa operação é a **limparTudo** da classe Escalonador a qual possui como ordem de complexidade O(n).

Conclusão

O trabalho teve como principal objetivo, na minha visão, o trabalho com as listas e filas e como elas se relacionam em uma aplicação real. Dito isso, por meio dos Hosts e URL's foi possível trabalhar com iterações e ordenações, sendo assim, podendo realizar as mais diversas operações.

Bibliografia

Algoritmos - Teoria e Prática - 3º Edição Americana traduzida, por Thomas Cormen

C++ Language Reference | Microsoft Docs