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

Profs. Gisele L Pappa e Wagner Meira Jr.
Estrutura de Dados 2021.02

Trabalho Prático 1 - Escalonador de URLs

Valor: 10 pontos (+ 2 para desafios)
Data da entrega: 09/12/2021

1. Introdução

Um dos principais componentes em uma máquina-de-busca é o **coletor**. Com o auxílio de robôs (também conhecidos como *crawlers, spiders*), eles varrem a Internet e realizam o download do conteúdo apontado por uma URL (o endereço de uma página). Sem perda de generalidade, essas URLs obedecem o seguinte formato:

```
<protocolo>://<host><path>?<query>#<fragmento>
```

Sabe-se que hoje existem trilhões de URLs e muitas delas apontam para conteúdo inexistente. Visto que a quantidade de recurso é limitada, o coletor necessita de um **escalonador** para definir a ordem que as páginas apontadas pelas URLs serão coletadas. A ordem depende da estratégia de coleta adotada e as duas mais conhecidas são: *depth-first* (busca em profundidade), que coleta todas as URLs de um host antes de passar para o próximo; e *breadth-first* (busca em largura), que prioriza a variedade e coleta URLs de diferentes hosts simultaneamente, coletando uma URL de cada sítio.

2. Especificação

Neste trabalho, você deverá desenvolver um **escalonador de URLs**, e adotar a estratégia *depth-first* visando priorizar sítios e URLs encontradas primeiro. Desta forma, o escalonador possui internamente uma **fila** de sítios conhecidos e cada sítio, por sua vez, possui uma **lista** de URLs conhecidas, conforme ilustrado abaixo:

```
host1.com: [http://host1.com] -> [http://host1.com/p1] -> [http://host1.com/p2.php]
host2.net: [http://host2.net] -> [http://host2.net/path]
h3.com.br: [http://h3.com.br/home.htm] -> [http://h3.com.br/a/b/c/news.html]
…
```

O *host* corresponde ao nome do sítio sem o "http://" e "www", e a lista de URLs do sítio deve estar ordenada, para priorizar URLs mais próximas à raiz do site, ou seja, menos profundas na hierarquia de diretórios. Isto quer dizer que http://ufmg.br deve ser coletada antes de http://ufmg.br/dcc/pos, e este será nosso critério de profundidade, calculado pela quantidade de barras no *path* da URL. Além desta, as seguintes regras também devem ser respeitadas:

1. Apenas o protocolo HTTP é permitido.
2. A barra "/" ao final da URL não deve ser considerada, logo "http://www.ufmg.br/" corresponderá a "http://www.ufmg.br"
3. O "www" no início da URL não deve ser considerado, logo "http://www.ufmg.br" corresponderá a "http://ufmg.br"
4. Apenas páginas HTML são permitidas. Para isto, arquivos com as seguintes extensões serão descartados: .jpg, .gif, .mp3, .avi, .doc e .pdf
5. Desconsiderar `#<fragmento>` nas URLs.
6. Obedecer a seguinte prioridade no escalonamento: hosts conhecidos primeiro e, em seguida, URLs com menor profundidade. Para URLs com mesma profundidade dentro do mesmo host, priorize aquelas conhecidas primeiro.

Observação 1: Antes de inserir uma nova URL na lista, compare a profundidade e já insira na posição correta.

O Coletor irá se comunicar com seu escalonador por meio de um arquivo de entrada (que será dado como argumento, na linha de comando, para o seu programa), que possui uma sequência de comandos a serem executados; e a resposta será gravada em um arquivo de saída. O arquivo de saída deve ter o mesmo nome do arquivo de entrada, seguido do sufixo "-out". Por exemplo, se o nome do arquivo de entrada for "coleta1.txt", o arquivo de saída deve ter o nome "coleta1-out.txt".

Assim, seu escalonador deverá ser capaz de executar os comandos a seguir:

- `ADD_URLS <quantidade>:` adiciona ao escalonador as URLs informadas nas linhas seguintes. O parâmetro `<quantidade>` indica quantas linhas serão lidas antes do próximo comando.
- `ESCALONA_TUDO:` escalona todas as URLs seguindo as regras estabelecidas previamente. Quando escalonadas, as URLs são exibidas e removidas da lista.
- `ESCALONA <quantidade>:` limita a quantidade de URLs escalonadas.
- `ESCALONA_HOST <host> <quantidade>:` são escalonadas apenas URLs deste host.
- `VER_HOST <host>:` exibe todas as URLs do host, na ordem de prioridade.
- `LISTA_HOSTS:` exibe todos os hosts, seguindo a ordem em que foram conhecidos.
- `LIMPA_HOST <host>:` limpa a lista de URLs do host.

- `LIMPA_TUDO:` limpa todas as URLs, inclusive os hosts.

Por exemplo, simulamos o funcionamento do escalonador com a respectiva entrada e saída ilustrada abaixo:

ENTRADA:

```
ADD_URLS 5
http://www.globoesporte.com/page1.js?q=copa#group1
ftp://ftp.globo.com
http://www.globo.com/
http://www.uol.com.br/
http://www.superdownloads.com.br/
ESCALONA 2
LISTA_HOSTS
ADD_URLS 4
http://www.globo.com/videos/jornalnacional 3
http://globo.com/ 1
http://globoesporte.com/spfc/index.html 4
http://globo.com/videos/ 2
ESCALONA_HOST globo.com 2
VER_HOST globo.com
```

SAÍDA:

```
http://globoesporte.com/page1.js?q=copa
http://globo.com
```

ESCALONA 2

```
globoesporte.com
globo.com
uol.com.br
superdownloads.com.br
```

LISTA_HOSTS

```
http://globo.com
http://globo.com/videos
```

ESCALONA_HOST globo.com 2

```
http://www.globo.com/videos/jornalnacional
```

VER_HOST globo.com

Observação 2: Seu escalonador não precisa garantir que uma URL não seja coletada novamente. Entretanto, deve evitar este problema enquanto ela ainda estiver na lista de URLs, ou seja, URLs duplicadas em memória.

Desafios (1 ponto extra para cada desafio)

Desafio 1: Altere seu escalonador para adotar a estratégia *breadth-first*. Assim, apenas uma URL por vez de cada host deve ser escalonada. O escalonador voltará a um determinado host apenas quando passar por

todos os outros.
Desafio 2: Altere seu escalonador para adotar uma estratégia *best-first*. Para isto, você escolherá uma URL do melhor host, que será aquele com maior número de URLs.

3. 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, 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**.

3. 1 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).

3.2 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 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 no início da Seção 3.

4. 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. Trabalhos entregues com atrasos sofrerão penalização de 2^{d-1} pontos, com d = dias de atraso.

5. Considerações Finais

1. Comece a fazer esse trabalho prático o quanto antes, enquanto o prazo de entrega está tão distante quanto jamais estará.
2. Leia **atentamente** o documento de especificação, pois o descumprimento de quaisquer requisitos obrigatórios aqui descritos causará penalizações na nota final.

[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.

3. Certifique-se de garantir que seu arquivo foi submetido corretamente no sistema.
4. 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**.

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
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.