Primeiro trabalho prático de implementação
Contador de caracteres em arquivos texto

Computação Concorrente (MAB-117) — 2016/2
Prof. Silvana Rossetto

[1]DCC/IM/UFRJ
4 de outubro de 2016

Descrição do problema

O algoritmo de Huffman é um método de compressão de dados sem perda para arquivos texto. (`https://en.wikipedia.org/wiki/Huffman_coding`). Na primeira parte do algoritmo, o número de ocorrências de cada caractere é contabilizado. Com base nesse resultado, códigos distintos são gerados para cada caractere, os quais serão usados para comprimir o arquivo. Caracteres com maior número de ocorrências recebem códigos de tamanhos menores, o que garante bons resultados para a compressão.

Neste trabalho, vamos **implementar a primeira etapa do algoritmo de Huffman**. Dado um arquivo texto, o programa deverá retornar a lista de caracteres alfanuméricos encontrados no texto e o número de ocorrências de cada caractere considerado.

Duas versões do problema devem ser implementadas e comparadas: **uma versão sequencial e outra concorrente**. O objetivo é conseguir que o tempo total de processamento da versão concorrente seja menor que o tempo total de processamento da versão sequencial para arquivos de tamanhos grandes.

Apenas as letras maiúsculas (A-Z) e minúsculas (a-z), os algarismos decimais (0-9), os caracteres de pontuação (? ! . : ; _ - ( )) e os seguintes caracteres especiais `@ % & $ #` e precisam ser contabilizados. Caracteres acentuados e os demais caracteres especiais e de acentuação e pontuação não precisam ser considerados.

O trabalho deverá ser implementado na linguagem C, usando a biblioteca `Pthreads` para a implementação concorrente. Não é necessário que todas as threads executem a mesma tarefa.

Entrada

Para a execução do programa sequencial, os seguintes dados deverão ser fornecidos como entrada, nesta ordem:

```
<nome do arquivo de entrada> <nome do arquivo de saída>
```

Para a execução do programa concorrente, os seguintes dados deverão ser fornecidos como entrada, nesta ordem:

```
<nome do arquivo de entrada> <nome do arquivo de saída>
<quantidade de threads>
```

Exemplo de arquivo de entrada O texto abaixo mostra um exemplo de conteúdo do arquivo de entrada:

```
Para ser grande, sê inteiro: nada
          Teu exagera ou exclui.
Sê todo em cada coisa. Põe quanto és
          No mínimo que fazes.
Assim em cada lago a lua toda
          Brilha, porque alta vive.
Ricardo Reis, in "Odes"
Heterónimo de Fernando Pessoa
```

Exemplos de arquivos de entrada foram extraídos do site `http://www.gutenberg.org/` e estão disponíveis no arquivo "livros.zip". Esses arquivos podem ser concatenados para gerar arquivos maiores. Outros exemplos de arquivos texto podem ser buscados na Internet.

Exemplo de arquivo de saída

O arquivo de saída deverá seguir o formato **csv**, com duas colunas, a primeira nomeada "Caractere" (conterá o caractere) e a segunda nomeada "Qtde" (conterá a quantidade de ocorrências encontradas do caractere). **Apenas os caracteres que apareceram no texto devem ser registrados no arquivo de saída.**

Um arquivo **csv** (*Comma-separated values*) é um arquivo de texto onde cada registro é armazenado em uma linha separada e cada campo dentro de um registro é separado por vírgula (uma definição formal é dada em https://tools.ietf.org/html/rfc4180). Arquivos **csv** podem ser abertos usando um ferramenta de planilha eletrônica (ex., openoffice).

Para o exemplo de entrada dado, o seguinte arquivo de saída deverá ser gerado:

```
Caractere, Qtde
., 4
:, 1
A, 1
B, 1
F, 1
H, 1
N, 1
O, 1
P, 2
R, 2
S, 1
T, 1
a, 24
c, 5
d, 10
e, 21
f, 1
g, 3
h, 1
i, 12
l, 5
```

```
m, 6
n, 9
o, 15
p, 1
q, 3
r, 10
s, 11
t, 6
u, 7
v, 2
x, 2
z, 1
```

Etapas do trabalho

A execução do trabalho deverá ser organizada nas seguintes etapas:

1. Compreender o problema, pensar e esboçar uma solução concorrente;
2. Projetar as estruturas de dados e as funçoes que deverão ser implementadas;
3. Construir um conjunto de casos de teste para avaliação da aplicação;
4. Implementar a solução projetada, avaliar a corretude do programa, refinar a implementação e refazer os testes;
5. Redigir os relatórios e documentar os códigos.

Artefatos que deverão ser entregues

1. **Relatório**: documentação do projeto da solução (esboço das estruturas de dados e lógica principal dos algoritmos da aplicação), testes realizados e resultados obtidos.
   Dentro de cada programa deverá ser medido separadamente o tempo total de leitura do arquivo, de processamento dos caracteres e de escrita do arquivo de saída. Todos esses valores deverão ser exibidos na saída do programa.
   Os arquivos de saída dos programas sequencial e concorrente deverão ser comparados para confirmar a corretude das soluções. O ganho de desempenho da aplicação concorrente deverá ser medido dividindo-se o tempo total da versão sequencial pelo tempo da versão concorrente (T_s/T_c). Os resultados deverão ser incluídos no relatório na forma de tabelas e de gráficos, variando-se o tamanho dos arquivos de entrada e o número de threads (1, 2, 4, 8).
   Os tamanhos dos arquivos de entrada deverão se aproximar dos seguintes valores: 500MBytes, 1GBytes, 2GBytes, 3GBytes, 4GBytes, 5GBytes.
   O relatório deverá conter informações suficientes para o professor compreender a solução proposta sem precisar recorrer ao código fonte do programa.
2. **Código fonte** e **roteiro para compilação e execução**.

Enviar para o email `computacao.concorrente.ufrj@gmail.com` com o assunto **"Trabalho 1 - 2016/2"** e dentro do corpo da mensagem os **nomes completos dos autores**.

Data de entrega e critérios de avaliação

O trabalho deverá ser feito em **dupla** e deverá ser entregue até o dia **25 de outubro de 2016**.

Os seguintes itens serão avaliados com o respectivo peso:

- Compilação, execução correta da solução e ganho de desempenho obtido: 5 pontos
- Relatório (incluindo projeto da solução e testes realizados): 2 pontos
- Modularidade, organização e documentação do código fonte, estratégias de paralelização: 3 pontos

Não é permitido copiar a solução do trabalho de outros colegas ou de terceiros.

Os alunos integrantes da equipe poderão ser chamados pelo professor para apresentar/explicar o trabalho.