Universidade Federal de Minas Gerais
Departamento de Ciência da Computação

DCC004 – Algoritmos e Estruturas de Dados II

Trabalho prático

Contagem de palavras

Valor: XX pontos
Entrega: 26/11/2018

1 Objetivo

Implementar um programa que lê um texto qualquer e imprime, em ordem alfabética, as palavras e sua frequência em um arquivo de saída.

2 Descrição

Vocês receberão um arquivo texto qualquer (um arquivo no formato `.txt`). Suponha o arquivo abaixo:

```
Dois mais dois são quatro, já dez vezes dez são cem.
```

Após ler o arquivo acima, seu programa deve imprimir a seguinte saída, contendo, em ordem alfabética, cada palavra encontrada no texto e o número de vezes que ela aparece:

```
cem 1
dez 2
dois 2
e 1
mais 1
quatro 1
são 2
vezes 1
```

A pontuação e as letras maiúsculas podem atrapalhar. Em relação às letras maiúsculas, é possível fazer uma função que converte em minúsculas utilizando código ASCII. Para remover a pontuação, deem uma olhada no link abaixo que pode ajudar (não testei as soluções propostas):

http://stackoverflow.com/questions/1841758/how-to-remove-punctuation-from-a-string-in-c

Neste programa, a ordenação deve ser feita apenas no final. Primeiro, deve-se guardar as palavras, na medida em que são lidas, em um vetor desordenado. Para saber se uma palavra já foi lida ou não, você deve implementar a busca de duas formas:

(a) Pesquisa sequencial

(b) Tabela hash com endereçamento aberto

Uma dica: Na tabela hash, guarde o índice da palavra no vetor desordenado. Assim, ao consultar a tabela hash, o resultado da consulta será "em qual posição do vetor desordenado está a palavra procurada".

Após ler o arquivo todo e calcular a frequência de todas as palavras, o vetor desordenado deve ser ordenado. Devem ser implementados dois métodos:

(a) Um com caso médio $O(n^2)$ (Bubble, Insertion ou Selection, você escolhe).

(b) Um com caso médio $O(n \log n)$ (Quick, Heap, Merge, você escolhe).

Assim você terá quatro combinações possíveis para executar, e a ideia é comparar o tempo de execução de cada uma destas 4 combinações:

1. Pesquisa sequencial com ordenação em $O(n^2)$
2. Pesquisa sequencial com ordenação em $O(n \log n)$
3. Pesquisa em hash com ordenação em $O(n^2)$
4. Pesquisa em hash com ordenação em $O(n \log n)$

Atenção: Seu programa deve receber quatro parâmetros na linha de comando. Um exemplo abaixo (no Windows substitua `./tp3` por `tp3.exe`)

```
./tp3 arquivo.txt 0 0 saida.txt
```

O primeiro parâmetro da linha de comando é o arquivo de entrada. O segundo parâmetro é 0 (para pesquisa sequencial) ou 1 (para tabela hash). O terceiro é 0 (ordenação $O(n^2)$) ou 1 (ordenação $O(n \log n)$). O último parâmetro é o nome do arquivo de saída.

Tempo de CPU: Para calcular o tempo do seu programa, você pode utilizar o exemplo que disponibilizei no Moodle (que funciona tanto no Linux quanto no Windows).

C++: Vocês podem utilizar algumas funcionalidades de C++ neste trabalho. No caso, podem usar `vector` e passagem por referência. Se achar mais fácil, pode também utilizar a biblioteca `string`. Não é permitido utilizar bibliotecas de C++ que já implementam alguns algoritmos.

Arquivos de teste: Os arquivos de exemplo disponibilizados foram gerados utilizando os links abaixo:

`http://www.cafw.ufsm.br/~bruno/disciplinas/desenvolvimento_web/material/lerolero.html`

`http://sebpearce.com/bullshit/`

3 O que deverá ser entregue

Para submeter o trabalho, envie pelo Moodle um arquivo `.zip` contendo apenas o código (arquivos `.cc` e `.h`). **Não inclua o executável**. O arquivo pode ser também nos formatos `.bz2`, `.rar`, `.tar.gz` ou `.tgz`. Ao receber seu código, nós compilaremos seu programa em um ambiente Linux com o seguinte comando:

```
g++ -Wall -lm *.cc -o tp3
```

O `-lm` é opcional e será usado apenas se você utilizar a biblioteca `math.h`.

Não há problema algum em desenvolver o trabalho no Windows, apenas tome cuidado para não utilizar bibliotecas que não compilam no Linux (por exemplo: `<windows.h>`). A documentação deve conter apenas:

- Explicação de qual a função hash utilizada.
- Análise de complexidade dos algoritmos.
- Tabela contendo o tempo de execução dos 4 casos para cada arquivo texto de exemplo.