UFMG/ICEx/DCC — Algoritmos e Estrutura de Dados 3

Trabalho Prático 5

Bacharelado — 1º Semestre de 2012

Este trabalho prático tem por objetivo a manipulação e o processamento de cadeias de caracteres através do uso de algoritmos de busca em strings.

1 Definição do Problema

Este trabalho consiste na criação de um Corretor Ortográfico (*Spell Checker*). Basicamente, esse tipo de ferramenta permite corrigir erros ortográficos de um texto específico de forma automatizada. Para que essa tarefa possa ser executada de maneira adequada, é necessário ter como referência um dicionário de palavras para, assim, estabelecer quais foram os equívocos cometidos pelo escritor do texto.

Idealmente, seria interessante que essa ferramenta funcionasse de maneira interativa com o usuário. Assim, seria possível aplicar funções diversificadas sobre o texto como acrescentar uma palavra ao dicionário ou até mesmo ignorar uma sugestão dada pelo corretor. Contudo, neste trabalho iremos reduzir o número de funções a serem consideradas e permitir a elaboração de uma resposta determinística (única) que não seja dependente da "opinião" ou do "discernimento" do aluno ou do usuário da ferramenta.

Então, a criação de um Corretor Ortográfico pode ser formulada da seguinte maneira: dado como entrada um dicionário de palavras e um texto qualquer, o aluno deverá implementar um algoritmo para processamento e casamento de cadeias de caracteres para corrigir os erros ortográficos identificados no texto de acordo com o dicionário fornecido.

Cabe ressaltar que podem haver casos de "empates" no momento da correção. Por exemplo, suponha que as palavras "mode", "node" e "code" estejam no dicionário disponibilizado e apareça no texto de entrada o termo "aode" - inexistente no dicionário. Como as palavras "mode", "node" e "code" possuem 3 caracteres em comum com o termo "aode" em análise, todas elas são candidatas igualmente possíveis para substituir o termo incorreto. Como o programa não é interativo e, portanto, não permite o usuário determinar a correção, iremos assumir que em casos de empate, a prioridade será definida pela ordem alfabética dos termos candidatos. Nesse caso, "aode" será subtituído por "code".

Como resultado final desse trabalho, espera-se que o aluno apresente o texto corrigido e algumas métricas para avaliação do texto e do processo de correção. Essas métricas são divididas em 2 grupos, abaixo:

1. Métricas do texto:
   - Número total de caracteres.
   - Número total de palavras.
   - Lista em ordem decrescente de até 20 palavras mais frequentes com filtragem de *stopwords**.
2. Métricas do processo de correção ortográfica:
   - Número total de correções realizadas.
   - Número de empates encontrados para os casamentos de caracteres.

* *Stopwords* são termos que representam preposições, artigos, ou pronomes.

O aluno **deve** elaborar um algoritmo que utilize estruturas com alocação de memória dinâmica.

1.1 Entrada e Saída

O arquivo executável deve ser chamado de *tp5* e deve receber como parâmetro os arquivos *input.txt*, *dictionary.txt*, *stopwords.txt* e *output.txt*, conforma apresentado abaixo:

```
./tp5 -i input.txt -d dictionary.txt -s stopwords.txt -o output.txt
```

O arquivo *input.txt* deve conter o texto a ser verificado pelo corretor ortográfico. O arquivo *dictionary.txt*, por sua vez, deve conter a lista de todas as palavras que compõem o dicionário, enquanto o *stopwords.txt* deve apresentar a lista das palavras ignoradas na contagem dos termos mais frequentes. Por fim, o arquivo *output.txt* deve possuir todas as métricas pedidas nesta especificação, assim como o texto corrigido.

Exemplo de entrada (arquivo *input.txt*):

```
A sailr wennt to sea to ee wht he culd ee. // texto original
```

Observe que o texto do arquivo *input.txt* pode estar em letras maiúsculas e/ou minúsculas, no entanto seu corretor ortográfico não faz distinção entre elas, portanto para todos os fins $\{A, a\}$ são considerados como o mesmo caractere. O texto de entrada pode apresentar sinais de pontuação como (vírgula, ponto final, ponto de exclamação, ponto de interrogação, dois pontos, etc.), porém eles devem ser ignorados. Não se preocupem com acentuação de palavras, pois o idioma utilizado em todos os arquivos mencionados será o inglês.

Exemplo de dicionário (arquivo *dictionary.txt*):

```
sailor
went
sea
see
what
he
could
```

Exemplo de filtro (arquivo *stopwords.txt*):

```
the
to
a
an
```

Observe que as palavras do dicionário *dictionary.txt* e do arquivo de filtragem *stopwords.txt* não necessariamente serão apresentadas em ordem alfabética. Além disso, podem haver palavras que estão presentes em ambos os arquivos. Sendo assim, para verificar se uma palavra precisa ser corrigida, verifique se ela está tanto no dicionário quanto na lista de *stopwords*.

A saída do programa deve ser impressa em um arquivo *output.txt* e deve conter as seguintes informações, na ordem indicada abaixo:

1. número total de caracteres do texto.
2. número total de palavras do texto (as *stopwords* também devem ser levadas em consideração).
3. lista em ordem decrescente de até 20 palavras mais frequentes e o número de suas ocorrências no texto corrigido.
4. número total de correções realizadas.
5. número de empates ocorridos.
6. texto corrigido.

Caso o texto seja composto de um número inferior a 20 palavras distintas, todas as palavras devem ser listadas. Há um marcador (#Texto) na linha anterior ao texto corrigido.

Exemplo de saída genérico (arquivo *output.txt*):

```
45 // número de caracteres
11 // número de palavras
see 2 // lista em ordem decrescente de até 20 palavras mais frequentes e o valor de suas frequências
could 1
he 1
sailor 1
sea 1
went 1
what 1
4 // número total de correções realizadas
0 // número de empates ocorridos
#Texto
A sailor went to sea to see what he could see // texto corrigido
```

Observe que a lista das palavras mais frequentes é ordenada de **duas** formas: primeiro em ordem decrescente pela frequência das palavras e segundo em ordem lexicográfica das palavras.

Os comentários nos exemplos de entrada e saída padrão foram apresentados apenas por propósitos didáticos, logo eles **não** devem ser incluídos nos arquivos para submissão.

Entrada e saída padrão devem seguir rigorosamente o formato descrito. Instâncias distintas para o problema devem ser geradas pelo próprio aluno para testar e avaliar seu algoritmo.

2 O que deve ser entregue:

2.1 Documentação: deve abranger pelo menos os seguintes pontos

- Introdução do problema apresentado.
- Modelagem e solução do problema.
- Apresentação e justificativa das ordens de complexidade de tempo e espaço.
- Principais decisões de implementação.
- Análise experimental quanto aos critérios especificados no trabalho.
- A documentação **não** pode exceder 10 páginas.

2.2 Código:

- O código fonte do trabalho deve ser submetido para compilação e execução em ambiente Linux, tendo como padrão os computadores dos laboratórios de graduação do DCC;
- Deve ser **obrigatoriamente** escrito na linguagem C (trabalhos implementados em outras linguagens como C++/Java/Python e outras **não** serão aceitos);
- As estruturas de dados devem ser alocadas dinamicamente e o código deve ser modularizado (ou seja, dividido em múltiplos arquivos fonte e fazendo uso de arquivos cabeçalho - .h);
- O utilitário Make deve ser utilizado para compilar o programa **(o arquivo de makefile deve ser submetido juntamente com o código fonte)**;
- A saída deve ser impressa seguindo estritamente o formato da especificação, caso contrário o resultado será considerado errado;
- O arquivo executável deve ser chamado de **tp5**. Não serão aceitos outros nomes de executáveis além do mencionado;
- Faça seu código de forma legível.

2.3 Entrega:

- Data de entrega: 07/06/2012
- Submissão: a documentação e o código do trabalho devem ser submetidos ao *minha.ufmg*. Para isso, compacte os dois (formato *tp5_NomeSobrenome.zip*) e faça a submissão. Teste seu arquivo compactado antes de enviá-lo.
- Apenas a documentação deve ser entregue impressa na secretaria do DCC. Não coloque nos escaninhos dos professores, entregue para a secretária para que sua documentação seja colocada no envelope de AEDS3. A documentação impressa pode ser entregue no dia útil seguinte da submissão digital. **Trabalhos que não tiverem a documentação entregue na secretaria, dentro do prazo de entrega, receberão nota 0.**
- Será postada uma planilha no Moodle sobre a entrevista do trabalho, leia-a e siga as orientações para o agendamento da sua entrevista.
- Será adotado média harmônica entre a pontuação obtida na execução e na documentação do TP, o que implica em valor zero caso alguma das partes não seja apresentada.
- A política para desconto por atraso de entrega do trabalho prático considera a fórmula:

  $$\frac{2^{d-1}}{0.32}\%$$

  onde d é o atraso em dias úteis. Note que após 5 dias úteis, o trabalho não pode ser mais entregue.

Observações

- O formato de submissão **DEVE** ser respeitado: *tp5_NomeSobrenome.zip*. Por favor, **NÃO** submetam trabalhos em outros formatos (.rar, .tar, .tar.gz). Certifique-se que seu nome completo foi incluído no nome do arquivo.
- **EXCLUA** o arquivo executável do arquivo compactado a ser submetido no *minha.ufmg*.
- **INCLUA** seu email na capa da documentação.

Referências

Projeto de Algoritmos com implementação em Pascal e C. Nívio Ziviani - *http://www.dcc.ufmg.br/algoritmos/*.

Introduction to Algorithms. T. Cormen, C. Leiserson, R. Rivest, and C. Stein. MIT Press, 3rd edition, 2009.