Estruturas de Dados e Algoritmos II
Trabalho Prático

Departamento de Informática
Universidade de Évora

2013/2014

v1.1

Caça aos erros

A entrada em vigor do novo acordo ortográfico criou a necessidade de reescrever muitos textos existentes. Esta necessidade é sentida, por exemplo, pelas editoras, cujas novas impressões de livros escritos anteriormente à adopção do acordo devem seguir a nova ortografia.

Para ajudar na conversão dos textos, foi-lhe encomendada uma aplicação que, recorrendo a um dicionário, identifique as ocorrências de palavras que deverão ser consideradas incorrectas. A encomenda inclui a exigência de que a aplicação deve não só ser eficiente no processamento de textos completos, mas também no tratamento de palavras isoladas, para poder ser usada como um serviço *web* de correcção ortográfica. (Neste contexto, o programa poderá ser invocado múltiplas vezes, num curto espaço de tempo, para processar somente uma ou duas palavras.)

1 Especificação

A aplicação a criar deverá ler um texto da sua entrada normal (*stdin*) e apresentar na sua saída normal (*stdout*) a lista dos k erros mais frequentes encontrados no texto, o número de ocorrências de cada um desses erros e as linhas onde ocorrem (até a um máximo de 50 por cada erro).

Para identificar os erros, o programa de identificação dos erros recorrerá a um dicionário, previamente criado por outro programa a partir da lista das palavras correctas.

1.1 Criação do dicionário

A aplicação incluirá um programa, escrito em C, que processará uma lista de palavras e construirá o dicionário de referência para a identificação dos erros.

Este programa lerá, da sua entrada normal, uma sequência de palavras, as palavras correctas, uma por linha. As palavras consistirão somente de letras maiúsculas e minúsculas, sem acentos nem cedilhas, e terão um comprimento máximo de 27 letras. O número máximo de palavras do dicionário será 400 000. Cada linha não conterá nada além da palavra.

Este programa deverá guardar o dicionário construído num ficheiro (ou ficheiros) que será (ou serão), posteriormente, consultado(s) pelo programa de identificação de erros.

Além do dicionário, o programa não terá nenhum *output*.

1.2 Identificação de erros

A aplicação incluirá um segundo programa, igualmente escrito em C, que fará a identificação e a localização dos erros de um texto. O programa acederá ao dicionário contido no(s) ficheiro(s) criados pelo programa de construção do dicionário.

Este programa lerá o texto da sua entrada normal como uma sequência de linhas, cada uma contendo zero ou mais palavras. Como acima, as palavras consistirão somente de letras maiúsculas e minúsculas, sem acentos nem cedilhas, e terão um comprimento máximo de 27 letras. Cada linha conterá somente palavras e espaços, em qualquer número e em qualquer posição, e terminará com o carácter fim-de-linha. Um texto terá entre zero e 500 000 palavras, com, no máximo, 100 000 palavras distintas.

Depois de terminar de ler o texto, o programa escreverá na sua saída normal a lista dos k erros mais frequentes, um por linha, por ordem decrescente de frequência, no seguinte formato:

$$\textit{palavra} \sqcup \texttt{(}N\texttt{)} \sqcup l_1 \sqcup l_2 \sqcup \ldots \sqcup l_M$$

onde *palavra* é o erro encontrado, N é o número de ocorrências desse erro no texto, l_1, l_2, …, l_M são as linhas onde se encontraram as primeiras M ocorrências do erro, por ordem não decrescente, M é o mínimo entre N e 50, `(` e `)` são os parêntesis curvos e $\sqcup$ representa *exactamente* um espaço. Se, nas primeiras M ocorrências de um erro, houver mais do que uma ocorrência do erro nalguma linha, o número dessa linha aparecerá tantas vezes quantas as ocorrências do erro nessa linha. A primeira linha lida será a linha 1.

O valor de k será indicado através do primeiro argumento do programa, e estará entre 1 e 100 000. Se não for passado nenhum argumento ao programa, o valor de k será 5. Se forem encontrados menos do que k erros distintos, o programa escreverá na sua saída só as linhas correspondentes aos erros encontrados. Se forem encontrados mais do que k erros distintos, o programa escreverá na sua saída exactamente k linhas.

A saída do programa estará ordenada decrescentemente pela frequência dos erros encontrados. Se dois erros tiverem o mesmo número de ocorrências, aparecerá primeiro aquele correspondente à palavra lexicograficamente menor. (Na ordem lexicográfica, $\texttt{A} < \texttt{AA} < \texttt{AAA} < \cdots < \texttt{AAB} < \cdots < \texttt{AZ} < \cdots < \texttt{Aa} < \cdots < \texttt{B} < \texttt{BA} < \cdots < \texttt{Z} < \cdots < \texttt{a} < \cdots < \texttt{b} < \cdots < \texttt{z} < \cdots$.)

O programa não produzirá nada na saída além do descrito acima e não terá nenhum argumento além do valor de k.

1.3 Erros

Uma palavra que está contida no dicionário de referência *tal e qual* como escrita no texto, *não* é um erro. Uma palavra que não está contida no dicionário de referência tal como escrita no texto, mas que está contida no dicionário só composta por letras minúsculas, *não* é um erro. Qualquer palavra encontrada no texto que não satisfaça uma das condições anteriores *é* um erro.

Por exemplo, se o dicionário só contiver as palavras `sim` e `Maria`, nenhuma das palavras `sim`, `SIM`, `Sim`, `sIm`, `siM` e `sIM` constitui um erro. A palavra `Maria` também não constitui um erro, mas qualquer outra forma dessa palavra, como `maria`, `MARIA` e `MaRiA`, constitui um erro. Estas três formas da palavra constituem erros *distintos*.

O dicionário pode conter a mesma palavra escrita de várias formas.

2 Exemplos

Retirando os acentos e todos os símbolos que não são letras ao texto da primeira secção deste enunciado, obtém-se:

```
Caca aos erros

A entrada em vigor do novo acordo ortografico criou a necessidade de reescrever
muitos textos existentes Esta necessidade e sentida por exemplo pelas editoras
cujas novas impressoes de livros escritos anteriormente a adopcao do acordo
devem seguir a nova ortografia

Para ajudar na conversao dos textos foi lhe encomendada uma aplicacao que
recorrendo a um dicionario identifique as ocorrencias de palavras que deverao
ser consideradas incorrectas A encomenda inclui a exigencia de que a aplicacao
deve nao so ser eficiente no processamento de textos completos mas tambem no
tratamento de palavras isoladas para poder ser usada como um servico web de
correccao ortografica Neste contexto o programa podera ser invocado multiplas
vezes num curto espaco de tempo para processar somente uma ou duas palavras
```

Aplicado a este texto, e recorrendo a um dicionário pós-acordo ortográfico, o programa de identificação de erros produziria:

```
adopcao␣(1)␣5
correccao␣(1)␣13
incorrectas␣(1)␣10
```

Para o parágrafo da secção anterior, na forma:

```
Por exemplo se o dicionario so contiver as palavras sim e Maria nenhuma das
palavras sim SIM Sim sIm siM e sIM constitui um erro A palavra Maria tambem nao
constitui um erro mas qualquer outra forma dessa palavra como maria MARIA e
MaRiA constitui um erro Estas tres formas da palavra constituem erros distintos
```

o resultado seria:

```
MARIA␣(1)␣3
MaRiA␣(1)␣4
maria␣(1)␣3
```

Usando como dicionário de referência um dicionário Inglês, a saída do programa para o parágrafo acima, com $k = 5$, seria:

```
constitui␣(3)␣2␣3␣4
erro␣(3)␣2␣3␣4
palavra␣(3)␣2␣3␣4
palavras␣(2)␣1␣2
Estas␣(1)␣4
```

3 Realização e entrega

3.1 Realização

O trabalho será realizado individualmente ou por grupos de dois elementos. Só serão considerados os trabalhos de grupos cujos elementos tenham todos obtido a pré-qualificação para o trabalho.

O código C entregue deverá estar de acordo com o *standard* C99, poder ser compilado com o GCC e executado em Linux. O código será compilado, com as opções `-std=gnu99 -Wall`, com o comando

```
gcc -std=gnu99 -Wall *.c
```

Todas as escritas e leituras de informação em disco deverão ser controladas explicitamente pelo programa.

3.2 Ambiente de execução e restrições

Na entrega, os programas serão testados numa máquina com sistema operativo Linux, de 32 bits, com um disco magnético normal (HD, e não SSD), e com páginas (de disco) de 4096 *bytes*.

A memória utilizada por cada programa estará limitada a 96 MB e o espaço disponível no disco para o dicionário é de 500 MB.

O tempo de execução do programa de criação do dicionário estará limitado a 60 segundos (limite que poderá ser revisto, se for necessário).

O programa de identificação e localização de erros deverá conseguir processar um texto com 400 000 palavras em até 1 segundo, e deverá conseguir processar, em sequência, 20 palavras em até 1 segundo (ou seja, em 1 segundo deverá ser possível executar o programa 20 vezes seguidas, processando uma única palavra de cada vez que é executado).

O dicionário de referência para o Português, usado neste enunciado e nos testes, pode ser descarregado daqui:

`http://www.di.uevora.pt/~vp/eda2/dic/pt-ao.txt.gz`.

3.3 Entrega

O trabalho será submetido através do Mooshak, uma aplicação que compilará o código e executará e testará os programas criados, no concurso "EDA2 (Trabalho)". O endereço para acesso ao concurso é:

`http://www.di.uevora.pt/~mooshak/cgi-bin/execute?command=login&contest=eda2_2013`.

O acesso ao Mooshak faz-se através da identificação do grupo e de uma *password*, que serão fornecidas após o envio da constituição do grupo ao docente de EDA2.

As submissões poderão ter uma de duas formas:

- Um ficheiro com todo o código do programa e com extensão `.c`.
- Um arquivo `tar` comprimido, num ficheiro com extensão `.tgz`, contendo somente os ficheiros com o código do programa (com extensão `.c` ou `.h`).

  Neste caso, os ficheiros deverão ser extraídos para a directoria corrente.

  Um comando que pode ser usado para criar um arquivo com estas características é o seguinte:

  ```
  tar cvzf <nome-do-arquivo>.tgz  <lista-dos-.c-e-.h>
  ```

No concurso estão definidos três problemas:

C (Criação)

Neste problema deve ser submetido o programa que processa e cria o dicionário, descrito na Secção 1.1.

Antes de submeter algum programa aos restantes problemas, deverá ter sido submetido, com sucesso, um programa a este problema.

Uma vez submetido, com sucesso, um programa a este problema, só será necessário voltar a submeter outro se o formato do dicionário mudar.

E (Entrega)

Este é o problema em que é feita a entrega do programa de identificação e localização de erros, especificado na Secção 1.2.

Durante a sua execução, o programa só poderá recorrer ao dicionário criado pelo programa de criação do dicionário, que já terá de ter sido submetido no problema C (Criação).

Só serão aceites os programas que passarem todos os testes deste problema.

T (Testes)

Este problema serve para testar o programa de identificação de erros com os testes disponíveis em:

`http://www.di.uevora.pt/~vp/eda2/testes/`.

Tal como no problema E (Entrega), o dicionário já deverá ter sido criado.

Importante Qualquer diferença entre os *outputs* dos programas e o que deviam ser (e tal como foram descritos) levará à sua não aceitação, assim como qualquer mensagem do compilador. Também impedirá a sua aceitação os programas terminarem sem ser por `return 0` (ou equivalente).

3.4 Relatório

Além do código, deverá ser entregue um relatório, em papel, com

- a identificação dos elementos do grupo,
- uma descrição pormenorizada das estruturas de dados usadas (bonecos, também conhecidos como diagramas, são algo que dá muito jeito para descrever estruturas de dados),
- a descrição do formato do(s) ficheiro(s) com o dicionário,
- a descrição do funcionamento do programa de identificação e localização de erros, nomeadamente do que acontece quando é lida uma palavra do texto e quando o texto chega ao fim (devem explicar o que acontece, sem apresentar código nem nomes de funções),
- a justificação de todas as escolhas feitas (suportada na complexidade dos algoritmos usados e na análise dos acessos a disco feitos pelo programa).
- o código dos programas, exactamente como submetido no Mooshak.

3.5 Considerações finais

A um trabalho aceite pelo Mooshak corresponderá uma nota igual ou superior a 10, desde que acompanhado de um relatório que cumpra os objectivos e que *todos* os elementos entregues sejam da autoria de *todos* os membros do grupo.

Qualquer elemento usado no trabalho que não seja da autoria dos elementos do grupo, deverá estar devidamente assinalado e a sua origem indicada. (Este uso só é admissível em situações *muito pontuais*, como, por exemplo, na escolha de uma função de *hash*.)

Na classificação do trabalho serão avaliadas a qualidade das estruturas de dados e dos algoritmos utilizados, a qualidade do código (incluindo a sua clareza, a sua legibilidade, e os comentários) e a qualidade do relatório.

3.6 Datas

A data limite de entrega do **programa** é 6ª-feira, dia 23 de Maio de 2014.

A data limite de entrega do **relatório** é 2ª-feira, dia 26 de Maio de 2014, até às 17h00.

A data das **discussões** é 4ª-feira, dia 28 de Maio de 2014.

Bom Trabalho