MAC323/BCC SEGUNDO EXERCÍCIO-PROGRAMA

VERIFICADOR ORTOGRÁFICO

Y. KOHAYAKAWA

Data de entrega. Você deve entregar este EP até as 18:00 do dia 30/4/2001, na secretaria do Departamento de Ciência da Computação, sala 256, bloco A.

1. Introdução

Neste exercício-programa, você estudará o uso de *árvores binárias de busca* (ABBs) na implementação de tabelas de símbolos, dando especial atenção à eficiência de tais implementações.

2. O problema a ser resolvido

O programa que você escreverá neste exercício deve resolver o seguinte problema: (*i*) a entrada de seu programa é um texto e uma coleção de dicionários (listas de palavras) e (*ii*) a saída de seu programa deve ser uma lista contendo as palavras do texto que não ocorrem em nenhum dos dicionários fornecidos.

2.1. **Um programa exemplo.** Inicialmente, você deve estudar cuidadosamente o programa `wordtest` escrito por Knuth, disponível em

`http://www.ime.usp.br/~yoshi/2001i/mac323/EPs/EP2/wordtest`

O programa de Knuth está escrito em sua linguagem de programação favorita, o CWEB. Veja a página

`http://www.ime.usp.br/~pf/mac328/aulas/cweb.html`

do professor Paulo Feofiloff sobre o CWEB.

O programa de Knuth usa uma estrutura chamada *treap*. Você não precisa estudar o funcionamento desta estrutura. Você deve, entretanto, entender que um treap é usado no programa de Knuth para implementar uma tabela de símbolos. Você também deve estudar os pontos finos da funcionalidade do programa de Knuth. Por exemplo, o que são as opções de linha de comando de `wordtest`?

3. O seu programa

A sua tarefa principal neste exercício-programa é escrever *um clone do programa de Knuth*. Do ponto de vista do usuário, seu programa deve se comportar exatamente da mesma forma que `wordtest`.

As diferenças entre seu programa, digamos `ep2`, e o programa de Knuth devem ser as seguintes:

Date: Versão de 2 de abril de 2001.

1. O `wordtest` supõe que as palavras do texto de entrada vêm uma por linha. Você deve implementar uma opção de linha de comando adicional, digamos `-t`, para permitir que o texto de entrada seja um arquivo texto "normal", com várias palavras por linha. Assim, quando fazemos

   ```
   ep2 -t [outras opções e argumentos]
   ```

   o seu programa deve primeiro isolar as palavras do texto antes de processá-las, como fazia seu primeiro EP.

   Observação. Você pode achar interessante estudar o programa `tr` do Unix; em particular, estude o que faz a chamada

   ```
   tr -cs a-zA-Z '\n'
   ```

   Os programas `tr` e `wordtest` em conjunto podem simular o seu programa com a opção `-t`.
2. A tabela de símbolos em seu programa deve ser implementada em um módulo separado, digamos `ST.c`, e o acesso a ela deve ser estritamente através de uma interface, digamos `ST.h`.
3. Você deve escrever duas implementações de `ST.c`. Uma delas deve usar uma ABB sem balanceamento. A segunda deve usar *alguma* ABB balanceada (fica a seu critério qual—escolha uma boa!). O seu sistema precisa ser tal que ele possa ser compilado com qualquer uma das duas implementações de `ST.c` sem qualquer modificação. Organize seus arquivos em dois conjuntos (diretórios). A única diferença entre os dois conjuntos deve ser o arquivo `ST.c`.
4. O seu programa deve ser "portável". Observe que no mínimo o seu programa deve compilar sem problemas com o `gcc`.

Observação. Costumo dar este EP regularmente, pois eu acredito que ele é bastante instrutivo. Você pode ver outras edições deste EP nas páginas abaixo:

1. MAC122, 2o. semestre de 1998:

   http://www.ime.usp.br/~yoshi/mac122/EPs/ep3.html

2. MAC324, 1o. semestre de 1999:

   http://www.ime.usp.br/~yoshi/mac324/projeto

Note que o nosso EP é uma *variante* dos EPs acima. Entretanto, pode valer a pena dar uma olhada nas páginas acima, pois elas contêm várias dicas.

4. Análise de desempenho

Faça uma análise empírica de desempenho, comparando seus dois programas e o programa `wordtest`. Faça um pequeno relatório com os resultados e conclusões.

4.1. **Dados.** Procure implementar sua tabela de símbolos de forma que os seus dois programas possam manipular arquivos de tamanho 'razoavelmente grandes' como, por exemplo, o *King James' Bible*, que contém algo como 800.000 palavras.

Como no primeiro exercício programa, uma boa fonte de arquivos para testar o seu programa é o Projeto Gutenberg.

4.2. **Dicionários.** Você pode usar os dicionários do Unix para processar textos em inglês. Para o português, você pode usar os dicionários

1. `br` (2.7M): uma lista de palavras do português brasileiro,
2. `pt` (4.2M): uma lista de palavras do português português,
3. `uniao` (4.8M): uma união de `br` e `pt` (sem repetições, naturalmente),

todos disponíveis em

`http://www.ime.usp.br/~yoshi/Dicios`

Gerei o arquivo acima para o português brasileiro a partir do trabalho de Ricardo Ueda. Veja, em particular, a sua página sobre `br.ispell`. Note que uma condição que devemos respeitar é que qualquer coisa que produzirmos com a ajuda deste arquivo deve ficar livremente redistribuível (veja a licença GNU GPL).

Para gerar os arquivos para o português português, usei o trabalho de José João Almeida. Este trabalho também é distribuído de acordo com a licença GNU GPL. Para entender a motivação e filosofia dos termos da licença GNU GPL, veja a página do Projeto GNU.

Uma página muito interessante sobre recursos computacionacionais para o português é

`http://www.portugues.mct.pt/recursos.html`.

Consulte!

5. Observações

1. *Este EP é individual.*
2. Preste atenção na modularização descrita neste enunciado.
3. Seja cuidadoso com sua programação (correção, documentação, apresentação, clareza do código, etc), dando especial atenção a suas estruturas de dados. A correção será feita levando isso em conta.
4. *Não imponha restrições arbitrárias sobre a entrada.* Quando usado forma inesperada, o seu programa deve parar de forma 'graciosa'.
5. Organizem-se na turma para fazer bons testes. Comparem entre vocês o desempenho de seus programas.
6. Seja criativo. Se seu programa fizer algo a mais do que foi pedido você poderá ganhar algum bônus na nota. Por outro lado, você *deve* respeitar as especificações dadas neste enunciado.
7. Entregue o seu EP seguindo os moldes usuais: listagem impressa, cabeçalho claro, disquete com os arquivos (coloque um `Makefile`) e eventuais arquivos de teste.

Observação final. Enviem dúvidas para a lista de discussão da disciplina. Eventualmente, ajustes no enunciado ocorrerão ao discutirmos este EP na lista.