

# TRABALHO 1: INDEXAÇÃO DE STRINGS

RAUL H.C. LOPES

## 1. INTRODUÇÃO

O objetivo do trabalho consiste em implementar e comparar duas estruturas diferentes de indexação strings: árvore B e árvore de sufixos de Ukkonen.

## 2. FUNDAMENTALMENTE PRÁTICO

Implemente estruturas de dados para indexação de todos os sufixos de um texto dado. Implemente também algoritmos para pesquisa no texto dado, usando os índices referidos.

### 2.1. As estruturas de indexação.

- **Árvore B:** Implemente algoritmo para indexar todos os sufixos de um texto, usando uma estrutura de árvore B [3] para armazenar o índice. O índice e o texto estarão armazenados em disco.
- **Árvore de sufixos:** Implemente algoritmo para indexar todos os sufixos de um texto, usando uma estrutura de árvore de sufixos com algoritmo de Ukkonen (veja notas de aula) para armazenar o índice.

2.2. **Algoritmo de busca.** Implemente algoritmo para pesquisar a existência de um sufixo em um texto.

- **Árvore B:** implemente algoritmo para pesquisar a existência de sufixo no texto, usando índice baseado em árvore B.
- **Árvore de sufixos:** implemente algoritmo para pesquisar a existência de sufixo no texto, usando índice baseado em árvore de sufixos.

2.3. **Os sufixos a considerar.** O texto a indexar poderá conter qualquer seqüência de caracteres ASCII de 7 bits. Porém, para efeito de índice serão considerados apenas caracteres alfabéticos. Por exemplo, considere o trecho abaixo<sup>1</sup>:

---

<sup>1</sup>Retirado do divetidíssimo **Inside Enderby** de Anthony Burgess.

PFFFFRRRUMMP.

And a very happy New Year to you too, Mr Enderby!

The wish is, however, wasted on both sides

Deve ser considerado para efeito de indexação como o string que começa com:

PFFFFRRRUMMPAndaveryhappyNewYeartoyoutooMrEnderby

Qualquer sufixo desse string deve figurar no índice.

**2.4. Os programas.** Implemente ao menos os seguintes programas:

- programa para indexação, usando árvore B.
- programa para pesquisa de sufixo, usando árvore B.
- programa para indexação, usando árvore de sufixos.
- programa para pesquisa de sufixo, usando árvore de sufixos.

Os programas de indexação receberão na linha de comando dois argumentos:

- nome de arquivo com texto a indexar;
- nome de arquivo que conterá índice.

Os programas de pesquisa receberão na linha de comando dois argumentos:

- nome de arquivo com texto indexado;
- nome de arquivo contendo índice.

O programa lerá uma seqüência de strings de *stdin* e, para cada string, produzirá no *stdout* os índices de todas as ocorrências (inteiros separados por brancos) do respectivo string no texto dado.

**2.5. O que entregar.** O seu trabalho consistirá em um arquivo  $\text{\LaTeX}$  de três a quatro páginas descrevendo o uso de cada programa e dados relevantes sobre a implementação. Além disso, você entregará os fontes dos programas e *makefile* que permitirá gerar compilados e documentação automaticamente. Os programas a entregar são:

- **btreeindex**: programa que gera índice, usando árvore B;
- **btreesearch**: programa de busca, usando árvore B;
- **btreetest**: programa de geração de teste para a indexação com árvore B;
- **ukindex**: programa que gera índice, usando árvore de sufixos;
- **uksearch**: programa de busca, usando árvore de sufixos;
- **uktest**: programa de geração de teste para a indexação com árvore de sufixos.

Você deverá entregar um *tarball* contendo *makefile* e fontes de programas e  $\text{\LaTeX}$ .

**2.6. A avaliação.** Os trabalhos serão avaliados de acordo com os seguintes critérios:

critério	valor
correção de btreeindex	1,0
desempenho de btreeindex	1,5
correção de btreeseach	0,5
desempenho de btreeseach	0,5
qualidade de btreetest	1,0
correção de ukindex	1,0
desempenho de ukindex	1,5
correção de uksearch	0,5
desempenho de uksearch	0,5
qualidade de uktest	1,0
qualidade do documento	1,0
desempenho como referee	1,0
correção de cache	1,0
desempenho de cache	1,0

Nos itens que avaliam correção valem as seguintes observações:

- Índice incorreto vale 0 pontos.
- Cada consulta com resposta incorreta retira um décimo de ponto.
- Cada execução que aborte retira dois décimos de ponto.
- Cada *memory leak* retira um décimo de ponto.

Nos itens que avaliam desempenho, serão adotados os critérios:

- Os trabalhos serão particionados em classes de equivalência de desempenho.
- Se houver  $n$  classes, os trabalhos da classe com melhor desempenho receberão  $n/n$  vezes o total de pontos do item; a segunda classe recebe  $(n-1)/n$  vezes o total de pontos do item; e assim por diante até à última classe (pior desempenho) que recebe  $1/n$  vezes o total de pontos do item.

Poderá receber até um ponto extra por trabalho de referee qualquer aluno que em grupo com outro colega apresente artigo para a turma (que poderá ser contestado) com comparação de desempenho e avaliação de correção de ao menos três trabalhos diferentes.

Os pontos de correção e desempenho de cache referem-se a implementação e avaliação de sistema de cache para reduzir acesso a disco conforme detalhado na seção 4.5.

### 3. CORREÇÃO FORMAL

Formalize em **ACL2** e prove a correção total dos algoritmos da seção 2.

### 4. REGRAS PARA EXECUÇÃO DO TRABALHO

O trabalho substitui parte do peso das provas, logo valem as seguintes observações:

- Este trabalho poderá ser executado em grupos de até 2 alunos. No entanto, valem as seguintes observações:
  - Se durante a execução deste trabalho ou do próximo ficar claro que um dos elementos do grupo não trabalhou (famoso parasita no vácuo do bobo), eu anulo o trabalho do grupo. Moral da história: se seu colega de grupo se candidatar a parasita no seu vácuo (ou em outra posição mais delicada e incômoda), afaste-o do grupo e eu terei prazer em exterminá-lo.
  - Trabalhos individuais valerão mais: terão peso 1.25 na nota final, reduzindo o peso da médias das provas em 1.25.
- Cuidado com troca de informação entre grupos! Plágio neste curso é considerado falta gravíssima:  $d > 1$  plágios anulam  $2^d$  trabalhos/provas.
- O trabalho é seu e não do professor. Eu darei referências e dicas, mas cabe a você desenvolver algoritmos, implementar programas, definir casos de teste. Não deixe para me procurar apenas nos últimos dias antes da entrega. Estarei disponível para tirar dúvidas a partir de 17.15h de segunda-feira e entre 12h e 13h de quarta-feira.

**4.1. O produto.** O produto a ser entregue consiste em documento  $\text{\LaTeX}$  e fontes de programas.

Qualidade de documento (que contribui para a nota final), inclui o uso competente do  $\text{\LaTeX}$  para gerar, por exemplo: referências cruzadas, referências bibliográficas (use *bibtex*), equações e tabelas decentemente formatadas. Por exemplo, índice remissivo de símbolos principais dos programas é extremamente bem-vindo. O documento ideal deveria ter a cara do  $\text{\TeX}$ book [2].

Você poderá apresentar sua documentação distribuída por vários documentos  $\text{\LaTeX}$ . Mas, garanta que as referências cruzadas entre documentos estão corretas e aprenda a usar o pacote *hyperref* do  $\text{\LaTeX}$  para criar links entre documentos. Aliás, se seu Linux/FreeBSD tiver

o pacote *teTeX-doc* instalado, você poderá ver documentação sobre os pacotes L<sup>A</sup>T<sub>E</sub>X disponíveis na sua instalação em:

**/usr/share/texmf/doc/index.html**

**4.2. Testes.** É bem-vindo e recomendado o uso de ferramentas de auxílio a teste automático, como: *nanna*, *aunit*, *cppunit*, *check*, *expect*, *dejagnu*, etc. **Importante:** testes bem especificados são baseados em condições de correção formal de algoritmos: testes de loops, por exemplo, garantem progresso, término, segurança (propriedade invariante.) Em resumo, validação por teste não elimina a necessidade de conhecer os fundamentos de prova formal de correção de programas. A avaliação da qualidade dos seus testes levará isso em conta.

**4.3. Arquivo submetido.** Você submeterá um arquivo de nome **tp1.tar.gz**

Eu usarei os seguintes comandos para obter seus programas compilados e seu documento em formato *PDF*:

```
> gzip -dc tp1.tar.gz | tar -xf -  
> make pdf  
> make compile
```

**4.4. Data de entrega.** O prazo final para entrega do trabalho está estabelecido em 21/03/2003, 18h, sem chance de alteração.

**4.5. Pontos extras.** Cada aluno poderá receber até três pontos extras se as seguintes tarefas forem realizadas:

- trabalho como referee na avaliação de ao menos três trabalhos diferentes — um ponto;
- implementação de sistema de cache para acesso a disco — 2 pontos.

O sistema de cache deverá fornecer acesso mapeado em memória para os arquivos de índice e texto. Use hash tornar mais eficiente aos *frames* do seu índice.

## REFERÊNCIAS

- [1] John Bentley and Robert Sedgewick, *Ternary tree*, Dr. Dobb's (1998).
- [2] Donald E. Knuth, *TeX: The program*, Addison-Wesley, 1986.
- [3] ———, *The art of computer programming, volume 3: Sorting and searching*, Addison-Wesley, 1998.