UNIVERSIDADE DE SÃO PAULO - ICMC
DEPARTAMENTO DE CIÊNCIAS DE COMPUTAÇÃO
SCC0261 Multimídia - 1º Sem/2016
Prof. Marcelo Manzato (mmanzato@icmc.usp.br).
Estagiário: Rafael D'Addio (rdaddio@icmc.usp.br).

PROJETO

Compressão de Texto

Divulgação: 21/03/2016.
Datas de entrega: Fase 1: 13/05/2016; Fase 2: 30/06/2016.
Grupos de no máximo 2 alunos.

Objetivos:

O objetivo principal do projeto é colocar em prática conceitos teóricos abordados em sala de aula relacionados a técnicas de codificação de mídias. Além disso, serão explorados conceitos adicionais sobre o efeito da aplicação de transformadas em textos e o impacto que isso acarreta em compressão. Para tal, os grupos deverão implementar a **Transformada de Burrows-Wheeler** (descrição dela pode ser encontrada na bibliografia ao final desse documento, que estará disponível no TIDIA), bem como técnicas de compressão e descompressão aplicadas a texto em geral.

Fases:

O projeto será dividido em duas fases incrementais, cada uma possuindo datas de entrega específicas.

Projeto - Fase 1 – entrega: 13/05/2016: cada grupo deverá desenvolver um compressor e um descompressor que seja capaz de realizar a **Transformada de Burrows-Wheeler (BWT),** e as seguintes rotinas de compressão: **Huffman** e **Run-length**, de modo independente e/ou integrado. Desse modo, o sistema deverá receber como parâmetro quais rotinas ele deverá realizar, podendo efetuar diferentes combinações entre elas, ou executá-las isoladamente.

O codificador deverá portanto receber como entrada um **arquivo de texto**, e como saída, deverá gerar um arquivo **binário** contendo os dados comprimidos. Espera-se que, após a descompressão, os arquivos original e descomprimido sejam idênticos.

O codificador deve ser responsável por fornecer para o decodificador todas as informações necessárias para realizar a decompressão em um cabeçalho no arquivo comprimido. Tais informações abrangem a configuração de rotinas utilizadas, árvore de Huffman (se necessário), tabelas de conversão de códigos, etc.

A implementação da BWT pode ser realizada de qualquer modo, porém, a bibliografia sugere (Capítulo 2) alternativas para a sua implementação eficiente. Implementações que seguirem tais sugestões serão bonificadas.

O tamanho dos blocos de texto a serem processados de cada vez pela transformada deverá ser flexível e definido através de um parâmetro ajustado pelo usuário. O programa deverá ser capaz de ser executado

por linha de comando, recebendo também como parâmetro a configuração requerida e os arquivos de entrada/saída. A linha de comando de chamada do codificador deverá ter o seguinte formato:

```
encode -i arquivo_original.txt -o arquivo_binario.bin  --bwt=X --txtblck=X --huffman=X --runl=X
```

Onde `--bwt`, `--huffman` e `--runl` são parâmetros binários (`X==[true/false]`), e `--txtblck` é o parâmetro que corresponde ao tamanho do bloco de texto a ser processado pelo BWT. Para descompressão:

```
decode -i arquivo_binario.bin -o arquivo_descomprimido.txt
```

Em conjunto com o código fonte, deverão ser entregues instruções de compilação e execução em um arquivo .txt separado.

Projeto - Fase 2 – entrega: 30/06/2016: a segunda fase do projeto engloba a realização de um estudo relacionado à capacidade de compressão das rotinas implementadas na fase 1 do projeto. Nessa etapa, os grupos deverão aplicar diversas combinações das 3 rotinas implementadas, bem como deverão analisar variações no tamanho do bloco de texto processado pela BWT. Toda análise deverá ser documentada em um relatório (mínimo de 4 páginas), contendo as comparações e discussões referentes aos resultados.

Deverão ser executadas as seguintes configurações, as quais serão comparadas entre si:

- BTW, Huffman e Run-length isolados;
- Huffman + Run-length (*);
- BTW + Huffman
- BTW + Run-length;
- BTW + Huffman + Run-length (*);

(*) A ordem de execução dos codificadores combinados deverá ser estipulada pelo grupo, visando obter melhor compressão.

A BTW deverá ser usado com os seguintes valores de bloco de texto: 100, 500, 1000 e 10000 caracteres.

As combinações serão aplicadas em bases de dados de diferentes domínios, disponíveis no site: http://pizzachili.dcc.uchile.cl/texts.html. No mínimo 3 bases diferentes deverão ser utilizadas. Cada base contém cerca de três a cinco arquivos de tamanhos diferentes (50 MB, 100 MB, 200 MB, etc).

O relatório deverá ter no mínimo 4 páginas, contendo as seguintes seções:

0. Título do relatório, nome, e-mail e no. USP dos autores
1. Introdução
2. Metodologia adotada (escrever aqui como configurou seu programa, quais bases/arquivos foram usados, decisões de projeto, se mudou alguma coisa no programa entregue na 1a. fase, etc.)
3. Resultados obtidos (razão de compressão, tamanho em MB dos arquivos original/comprimido, tempo de processamento, etc.)
4. Discussão sobre os resultados (escrever uma análise/justificativa dos seus dados de acordo com o conhecimento teórico obtido na literatura e nas aulas. A aplicação em diferentes domínios influenciam nesses resultados? Por quê?)
5. Conclusões (considerações finais, limitações do estudo, o quê aprendeu com o estudo, etc.)

Na exposição dos resultados, é encorajado o uso de tabelas e/ou gráficos. Por exemplo, pode-se usar um gráfico de barras para comparar diferentes técnicas na mesma figura. Relatórios que exibirem os resultados de maneira mais clara serão bonificados.

Considerações sobre o projeto (fase 1 e fase 2):

1. Cada grupo deverá desenvolver um projeto original. Cópias, mesmo parciais, não serão toleradas.
2. A implementação poderá ser feita em C/C++ (compilador gcc) ou Java (JDK 8). Não será permitido o uso de bibliotecas terceirizadas (apenas as que vêm por padrão no compilador). Toda correção será feita em ambiente Linux/Unix.
3. O código deverá ser documentado, contendo comentários indicando brevemente o que cada módulo, função/método e classe faz.
4. O programa deverá operar por linha de comando (sem interface gráfica), permitindo que parâmetros de entrada, controle e saída sejam definidos por lá (veja especificação acima de como definir esses parâmetros).
5. Deverão ser entregues na 1a. fase: os arquivos contendo o código-fonte e um arquivo TXT (relatório) contendo as informações do grupo e instruções de compilação e execução dos códigos.
6. Deverá ser entregue na 2a fase: um arquivo PDF contendo o relatório. Se houver alteração no código da 1a. para a 2a. fase, será necessário re-enviar o código novamente.
7. Todos os arquivos deverão ser agrupados em um único arquivo ZIP.
8. Modo de entrega: um dos componentes do grupo deverá fazer *upload* dos arquivos no seu respectivo escaninho no Tidia até a data da entrega.
9. Será descontado 1.0 ponto da nota final por dia de atraso.

Referências

[1] Donald Adjeroh, Timothy Bell, and Amar Mukherjee. 2008. The Burrows-Wheeler Transform: Data Compression, Suffix Arrays, and Pattern Matching (1 ed.). Springer Publishing Company, Incorporated.

Bom trabalho!