Teoria da Informação - 060046 UNISINOS 2019/2

Enunciados dos Trabalho T1 e T2

Observações importantes:

- entrega em 30/09/2019;
- em grupos ou individual;
- entrega (postagem) via moodle

o objetivo geral:

Elaborar uma solução computacional que codifique (compacte) e decodifique (descompacte) arquivos. Para isto deve ser implementado um protótipo (usando a linguagem de sua escolha) que deve ser testado com a compactação e descompactação dos arquivos **alice29.txt** e **sum** do corpus de Canterbury (*corpus.canterbury.ac.nz/descriptions/#cantrbry*).

A meta é desenvolver uma solução de compactação (sem perda – *lossless*) empregando uma combinação (livre) de abordagens de codificação a nível de **símbolo** como Golomb/Elias-Gamma/Fibonacci/unária/delta, bem como Huffman, mas que também trate a ocorrência da repetição de **padrões de símbolos** ao longo do texto, para isto podendo-se lançar mão de abordagens como os métodos RLE, BWT ou MTF, ou ainda algoritmos da família LZ.

Finalmente, após compactado, deve ser empregado o algoritmo de **detecção de erros** CRC sobre uma região específica do arquivo compactado. A escolha desta região é livre, podendo o CRC ser aplicado sobre um cabeçalho usado para armazenar uma árvore Huffman, por exemplo. O decodificador deve testar o CRC contido no arquivo a fim de verificar se esta região não sofreu modificações.

Qualquer uma destas abordagens (algoritmos) pode ser customizada/modificada, desde que estas alterações estejam documentadas no trabalho T1. Também podem ser desenvolvidos e empregados nesta solução algoritmos próprios, que podem ser baseados/inspirados em alguma abordagem já existente. Nesse caso, a estrutura e funcionamento deste algoritmo também devem constar da documentação do T1.

O principal critério no projeto do protótipo é o ***fator de compactação***. Consumo de recursos (como memória e CPU) e o tempo de compactação/descompactação não precisam ser levados em conta nesta implementação.

T1 (o projeto):

Descrever a estrutura (arquitetura) da solução, justificando as escolhas realizadas. Desta arquitetura deve constar o *pipeline dos algoritmos* de

codificação empregados (por exemplo: o texto é primeiramente compactado com o algoritmo Golomb e em seguida o resultado deste passo é codificado com Huffman. Após esta fase, é aplicado CRC-8 sobre os 8 primeiros bytes...).

Também deve constar deste documento as referências a todas bibliotecas/módulos/pacotes de terceiros empregadas no protótipo. Modificações realizadas nos algoritmos também devem estar descritas neste documento.

T2 (a implementação):

Implementação do protótipo, com o teste do mesmo realizando a compactação/descompactação dos arquivos ***alice29.txt*** e ***sum***.

Devem ser entregues/postados: o código executável e o código fonte (ou a URL onde estes se encontram disponíveis), além da documentação básica da implementação (*javadocs*, por exemplo) juntamente com um README com orientações para instalação e execução do protótipo, além de comentários sobre limitações deste e outras observações julgadas pertinentes.

O código pode também ser disponibilizado via *github*, *bitbucket*, etc., assim como a aplicação pode estar hospedada online.

Observações:

- A leitura do arquivo pode ser realizada byte a byte.
- O processo de geração do arquivo compactado pode implicar no uso de um buffer de saída onde os *codewords* (resultantes da codificação do arquivo) irão sendo inseridos um a um, empregando-se para isto operações bit a bit (*bitwise ops*). Finalmente, o conteúdo do buffer de saída e gravado em um arquivo.
- Caso sejam empregado métodos que impliquem no uso de estruturas adicionais de dados, estas deverão ser gravadas no arquivo codificado (compactado) para que o *decoder* consiga realizar a descompactação. Por exemplo, no caso do uso de Huffman deve ser armazenado também um conjunto de dados que permita ao *decoder* reconstituir a arvore Huffman original. Uma situação similar acontece com a codificação Golomb, no que diz respeito ao divisor.