Enunciado dos Trabalhos T1 e T2 - GA

Observações importantes:

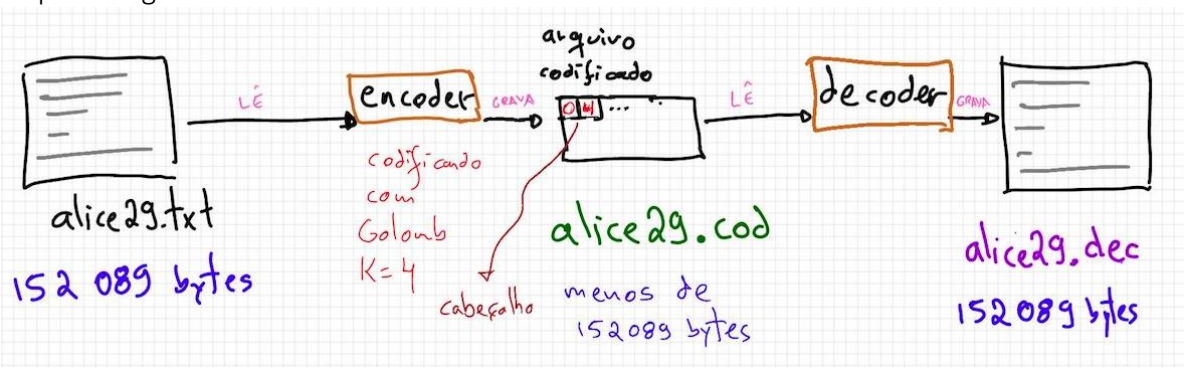
- entrega em 06/outubro/2021 via postagem na tarefa moodle
- pode ser desenvolvido em grupos (de até 4 pessoas) ou individualmente

Objetivo do T1:

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*), disponíveis na pasta junto ao enunciado no moodle.

A meta é desenvolver uma solução de compactação (sem perda – *lossless*) empregando as abordagens de codificação a nível de **símbolo**. As formas de codificação que devem ser suportadas são: Golomb, Elias-Gamma, Fibonacci, Unária e Delta. O usuário deve poder escolher o tipo de compactação que será empregado, bem como o arquivo a ser codificado/decodificado.

A figura a seguir apresenta a estrutura do processo: a partir da leitura do arquivo original, o *encoder* gera o arquivo codificado (compactado); o *decoder* pode então ler o arquivo codificado e gerar um arquivo decodificado, que deve ser exatamente igual ao arquivo original.



No arquivo compactado, os dois primeiros bytes formam um cabeçalho, armazenando meta informação sobre como o arquivo foi codificado: o primeiro byte indica o tipo de codificação (0: Golomb, 1:Elias-Gamma, 2:Fibonacci, 3:Unária e 4:Delta) e o segundo byte irá possuir o valor do divisor caso tenha sido usada a codificação Golomb (caso contrário o valor do segundo byte é zero); consequentemente os *codewords* resultantes do processo de codificação são armazenados no arquivo a partir do terceiro byte.

Observações:

- A leitura e gravação dos arquivos 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

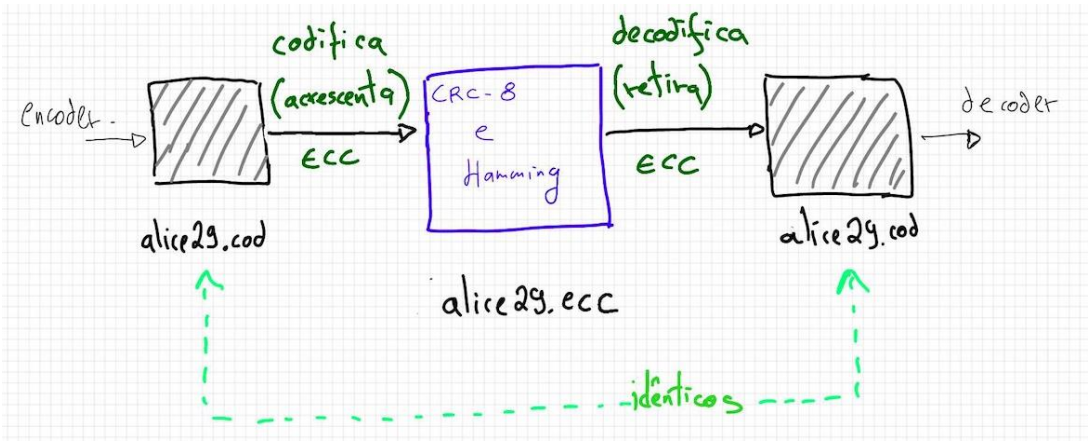
sendo inseridos um a um, empregando-se para isto operações bit a bit (*bitwise ops*). Finalmente, o conteúdo do buffer de saída é então gravado em um arquivo.

- O *encoder* e o *decoder* podem ser na realidade o mesmo programa, sendo somente parametrizado de forma distinta para cada caso (um parâmetro pode ser o modo de operação: codificar ou decodificar)

Objetivo do T2:

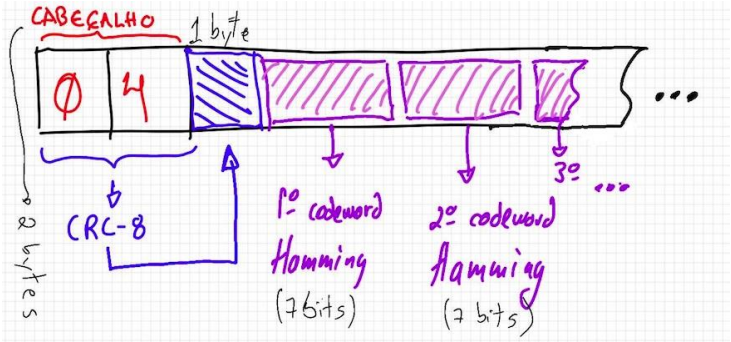
Neste trabalho deve ser acrescentado ao cenário do T1 técnicas de tratamento de ruído (códigos de correção de erro - ECC). Essa funcionalidade pode ser acrescentada de maneira separada/independente ou acrescida/integrada às implementações do *encoder* e do *decoder*.

A figura a seguir apresenta esta nova etapa no processo, que fica após a codificação do arquivo original e antes da decodificação do mesmo.



Depois do arquivo ter sido codificado, ele é recodificado ganhando informação adicional para tratamento de ruído. Devem ser implementadas duas técnicas:

- Logo após o cabeçalho do arquivo, deve ser acrescentado/gravado um byte resultante do cálculo CRC-8 (ATM) dos dois bytes do cabeçalho;
- Depois disso serão armazenados os *codewords* Hamming formados a partir da leitura da informação dos *codewords* presentes no arquivo codificado. Por ex.: a cada 4 bits dos *codewords* do arquivo codificado alice29.cod é gerado um *codeword* Hamming de 7 bits que será armazenado no arquivo alice29.ecc



Em função do uso de codificação Hamming o tamanho do arquivo com ECC é maior que o do arquivo codificado.
Caso haja necessidade pode ser acrescentada informação adicional ao arquivo codificado (por ex., o valor do tamanho em bytes do arquivo original).

Tratamento dos erros:

No caso de ocorrência de erro no processo de decodificação do arquivo com ECC, deve ser apresentado para o usuário (ou gravado em um arquivo de log) as informações pertinentes ao erro. São dois os tipos de erro tratados:

- o erro pode ter sido detectado na decodificação de uma palavra Hamming, e nesse caso o erro deve ser corrigido mas o evento deve ser notificado ao usuário (ou anotado no log);
- o erro pode ser detectado na checagem/verificação do CRC do cabeçalho (em função de alguma alteração sofrida nos 2 primeiros bytes do arquivo codificado); nesse caso o usuário é avisado mas não há correção a ser realizada - nesta situação, o ideal é impedir o processo de decodificação do arquivo codificado, pois a meta informação (o cabeçalho) foi corrompido.

Observações finais:

Implementar os algortimos. Não empregar/reusar código pronto/já existente.

Devem ser entregues/postados:

- o código executável e o código fonte (ou a URL onde estes se encontram disponíveis) e
- a documentação básica da implementação juntamente com um README com orientações para instalação e execução do protótipo, além de comentários sobre limitações deste e outras observações julgadas pertinentes. 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

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