UNIVERSIDADE FEDERAL DO ABC

PARADIGMAS DE PROGRAMAÇÃO

Q2 – 2021

Wesley Axel de Barros

RA 11201722424

PROJETO FINAL PARA A DISCIPLINA

Codificação de Huffman, LZW e LZO em Haskell.

Santo André, SP

2021

Introdução:

Para o projeto final da disciplina de paradigmas da programação, foi desenvolvido na linguagem Haskell uma interface gráfica web, aonde através de 3 métodos diferentes de compressão, é possível comprimir um texto escrito pelo usuário, permitindo ver o resultado do texto comprimido, permitindo também ver a diferença de tamanho em bytes e bits entre o texto transformado no tipo 'String', o texto comprimido por cada método diferente, demonstrando também a taxa de compressão de cada método e o tempo decorrido que cada algoritmo demorou para realizar a compressão.

Os métodos escolhidos para a compressão do texto foram a **codificação de Huffman**, o algoritmo de compressão **LZW (Lempel-Ziv-Welch)** e o algoritmo de compressão **LZO (Lempel–Ziv–Oberhumer).** Cada algoritmo foi implementado de uma forma diferente, sendo ambos o de Huffman o LZW implementados diretamente, e o algoritmo LZO implementado através de um "codec" disponibilizado no "Hackage.haskell.org".

Para a interface web, foi utilizado uma biblioteca de códigos em Haskell: "Threepenny-GUI", que permite através da linguagem Haskell, implementar uma interface gráfica que utiliza como display um navegador instalado no sistema. (Google Chrome, Opera, Mozilla Firefox, etc.)

Inicializando e rodando o projeto:

Para inicializar e executar o projeto, o mesmo deve ser baixado do repositório GitHub no qual se encontra, e posto em uma pasta dentro de sua máquina, aonde essa pasta será acessada via terminal, que deve possuir a ferramenta "Stack" da linguagem Haskell instalada e configurada.

OBS: O projeto foi desenvolvido em uma máquina virtual Linux Ubuntu através da ferramenta WSL disponibilizada pelo Windows.

Acessando a pasta via terminal, o seguinte comando deve ser executado:

- stack build

Com esse comando executado, a ferramenta Stack vai construir o projeto para que o mesmo possa ser executado. Esse processo pode demorar pois todos os arquivos necessários para o mesmo serão baixados através da ferramenta.

O processo será concluído caso a ferramenta Stack não apresente nenhum erro.

Com o processo construído, o mesmo já pode ser executado através do comando:

- stack run

Executando o comando acima, o projeto entrará em execução, utilizando como endereço na web, o I.P.: (`127.0.0.1`) e a porta: (`8023`)**,** ambos endereço e portas padrão utilizados pela biblioteca da interface gráfica.

O projeto estará em execução caso a seguinte mensagem seja exibida no terminal:

Com isso, apenas resta acessar em um navegador web escolhido pelo usuário, o endereço seguido da porta: **127.0.0.1:8023**

Assim se torna possível testar os algoritmos através de um texto introduzido no campo "Insira o texto para comprimir", e apertando o botão "Comprimir", para assim verificar o texto comprimido e seus dados de tamanho, a taxa de compressão e o tempo decorrido.

Um texto introduzido, é transformado em um tipo "String", que em Haskell, é uma lista de tipos "Char". Cada tipo "Char", ocupa um byte, que são 8 bits de tamanho.

O algoritmo de codificação de Huffman, faz a compressão do texto, verificando as frequências de cara caractere introduzido no texto, e com essa frequência é construído uma arvore binaria recursiva, que com a leitura da junção da frequência de dois caracteres de menor frequência, permite construir uma codificação binaria para cada caractere, fazendo assim a compressão do texto em uma combinação de codificação binaria.

Já o algoritmo LZW, faz a compressão do texto, construindo um dicionário, com todos os símbolos do alfabeto, e para cada caractere do texto, é feito a leitura do dicionário e construído assim uma lista da combinação dos caracteres com base na resposta do dicionário.

O algoritmo LZO, funciona diferente, com base em um codec obtido, vários algoritmos diferentemente implementados, que pegam o texto introduzido transformado no tipo "ByteString" e faz a compressão em blocos de datas, de tamanhos iguais.

Testes e Conclusão:

Realizado diversos testes com os algoritmos, foi possível chegar nos seguintes resultados:

1. Para textos com diversos caracteres diferentes, tamanho pequeno e poucas repetições, os algoritmos de compressão não possuem taxa de compressão alta.
2. Para textos com diversos caracteres diferentes, com um tamanho consideravelmente grande altas repetições, o algoritmo LZO se destaca fazendo a compressão em blocos de data semelhantes.
3. Para textos com poucos caracteres diferentes, com altas repetições ou não, o algoritmo de codificação Huffman se destaca por permitir assim representar o texto codificado através de uma codificação binaria.
4. Para o algoritmo LZW, o mesmo passa a ter boa taxa de compressão com textos de tamanhos grandes, pelo seu método de conversão em uma lista.

Para o resultado de tempo decorrido de cada algoritmo, se utilizou de um Biblioteca externa que não retornou o tempo esperado para cada um.

Com todos os testes realizados na interface, é possível então verificar como cada algoritmo funciona, o desempenho de cada um na linguagem Haskell, e as diferenças de cada um através das taxas de compressão.

Link do vídeo de teste do algoritmo:

https://www.youtube.com/watch?v=ziKMXhAIUR4