Universidade Estadual do Oeste do Paraná – UNIOESTE
Ciência da Computação

Alunos: Bruno de Castro Brezolin
Vinicius Yuri Capponi

Relatório de Análise do algoritmo de compressão implementado

Foz do Iguaçu 2022

Para realizar a análise do algoritmo implementado, foram escolhidos cinco arquivos diferentes, sendo eles:

- 2 gerados pelo script *gerar_arquivo_teste.py* que acompanha o código fonte do programa
- Lista de todas as palavras do português brasileiro, baixado no site https://www.ime.usp.br/~pf/dicios/index.html
- Bíblia, baixado no site https://umsocorpo.com.br/biblia-sagrada-em-txt-versao-revista-e-atualizada/
- 500.000 dígitos de Pi, gerados com o programa y-cruncher

Com estes arquivos, foi executada uma série de compressões e descompressões a fim de medir o desempenho e a taxa de compressão do algoritmo. Os resultados podem ser vistos na tabela a seguir:

Tipo Arquivo	Descrição	Tamanho normal	Tipo compressão	Tamanho comprimido (% tamanho original)	Tempo Compressão	Tempo Descompressão
Texto gerado por programa	5 palavras repetidas	5.858KB	Palavra	570KB (9.73%)	1.242s	0.187s
Texto gerado por programa	5 palavras repetidas	5.858KB	Caractere	2.604KB (44.45%)	0.583s	0.639s
Texto gerado por programa	1 palavra repetida	52.735KB	Palavra	2.198KB (4.17%)	11.289s	1.032s
Texto gerado por programa	1 palavra repetida	52.735KB	Caractere	15.381KB (29.17%)	4.799s	4.015s
Lista de palavras	Todas as palavras do português brasileiro	2.859KB	Palavra	8.609KB (301.12%)	1.401s	0.676s

Lista de palavras	Todas as palavras do português brasileiro	2.859KB	Caractere	1.524KB (53.30%)	0.346s	0.327s
Texto natural em português	Bíblia em “.txt”	3.930KB	Palavra	2.124KB (54.04%)	1.244s	0.507s
Texto natural em português	Bíblia em “.txt”	3.930KB	Caractere	2.246KB (57.15%)	0.522s	0.556s
Dígitos de Pi	500 mil dígitos	488.282KB	Caractere	213.620KB (43.75%)	56.963s	59.040s
Dígitos de Pi	500 mil dígitos	488.282KB	ZIP (7-zip)	222.347KB (45.53%)	51s	5s

Nas linhas 1, 2, 3 e 4 da tabela, podemos observar que o algoritmo pode chegar a altos níveis de compressão, chegando gerar um arquivo mais de vinte vezes menor que o original na linha 3, onde é o caso ideal para compressão por palavra, tendo apenas uma palavra, o código de Huffman terá apenas 1 bit, que será repetido quantas vezes necessário para gerar o arquivo. Nestas mesmas linhas podemos ver que em alguns casos a compressão por caractere pode ser mais rápida que a compressão por palavra, mas isso acompanha um arquivo significativamente maior e perda de desempenho na descompressão para este caso.

As linhas 5 e 6 mostram um caso ideal para a compressão por caractere, onde a compressão por palavra falhou, devido a não haver palavras repetidas na lista, gerando um arquivo mais que três vezes maior que o original, enquanto a compressão por letras consegue gerar um arquivo apropriadamente comprimido e em tempo muito menor.

Analisando as linhas 7 e 8 da tabela, podemos ver que um caso utilizando texto natural onde letras e palavras têm níveis normais de repetição, ambos os métodos de compressão acabam com resultados muito parecidos, podendo escolher entre uma taxa de compressão ou tempo de execução um pouco melhores

As últimas duas linhas da tabela foram feitas com intuito de comparar o algoritmo desenvolvido com um algoritmo de compressão utilizado no mundo real. Nos quesitos de tempo e taxa de compressão ambos podem ser comparados, novamente trocando um pouco de velocidade por um pouco mais de compressão, porém analisando a taxa de descompressão podemos ver que existe uma grande diferença de performance entre os dois.