UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
INTRODUÇÃO À TEORIA DA INFORMAÇÃO

Trabalho 1 - LZW

ANTÔNIO JONAS GONÇALVES – 2016021023

JORDY ALLYSON DE SOUSA - 11426758

THIAGO ALVES DE ARAUJO – 2016019787

1 – INTRODUÇÃO

O projeto consiste em desenvolver um compressor e descompressor aplicando o algoritmo LZW (Lempel-Ziv-Welch) e utilizando um tamanho de dicionário variável.

2 – DESENVOLVIMENTO

Para o desenvolvimento do projeto, foi utilizado a linguagem de programação python. Abaixo podemos ver os trechos da implementação do compressor e descompressor. O desenvolvimento do código foi baseado em pseudocódigos que estão presentes na literatura e em alguns artigos.

```
174
175     #Percorre o arquivo
176     for c in Dados:
177
178         novo_c = string + chr(c)
179
180         if novo_c in Dicionario:
181             string = novo_c
182
183         else:
184             Dados_cod.append(Dicionario[string])
185
186             #Se o dicionario não estiver cheio, insere o novo chr
187             if(len(Dicionario) < Tam_tab):
188                 Dicionario[novo_c] = Tam_dict
189                 Tam_dict += 1
190                 string = chr(c)
191
192             #Se o dicionario estiver cheio, reseta
193             else:
194                 Dados_cod.append(Dicionario[string])
195                 Tam_dict = 256
196                 Dicionario = {chr(i): i for i in range(Tam_dict)}
197                 string = ""
198
199     if string:
200         Dados_cod.append(Dicionario[string])
201
```

Figura 1 Compressor

```
134
135     #Percorre o arquivo
136     for code in Dados:
137
138         if not (code in Dicionario):
139             Dicionario[code] = string + (string[0])
140
141         Dados_decod += Dicionario[code]
142
143         if not(len(string) == 0):
144             Dicionario[Prox_cod] = string + (Dicionario[code][0])
145             Prox_cod += 1
146
147         string = Dicionario[code]
148
149         if len(Dicionario) == Tam_tab:
150             Tam_dict = 256
151             Dicionario.clear()
152             Dicionario = dict((i, chr(i)) for i in range(Tam_dict))
153
```

Figura 2 Descompressor

É valido ressaltar que os dados comprimidos foram salvos utilizando uma estrutura do python que salva dados binários em 16bits.

2 – RESULTADOS

Abaixo podemos ver os gráficos com os resultados obtidos após realizarmos os testes variando o tamanho do dicionário de K = 9 até K = 16. O primeiro gráfico mostra o tamanho dos arquivos obtidos após comprimirmos o arquivo de texto e o arquivo de vídeo com todos os valores de K. Lembrando que o arquivo de texto “corpus16Mb” originalmente possui um tamanho de 15272Kb e o arquivo “mapa.mp4” possui um tamanho de 22515Kb.

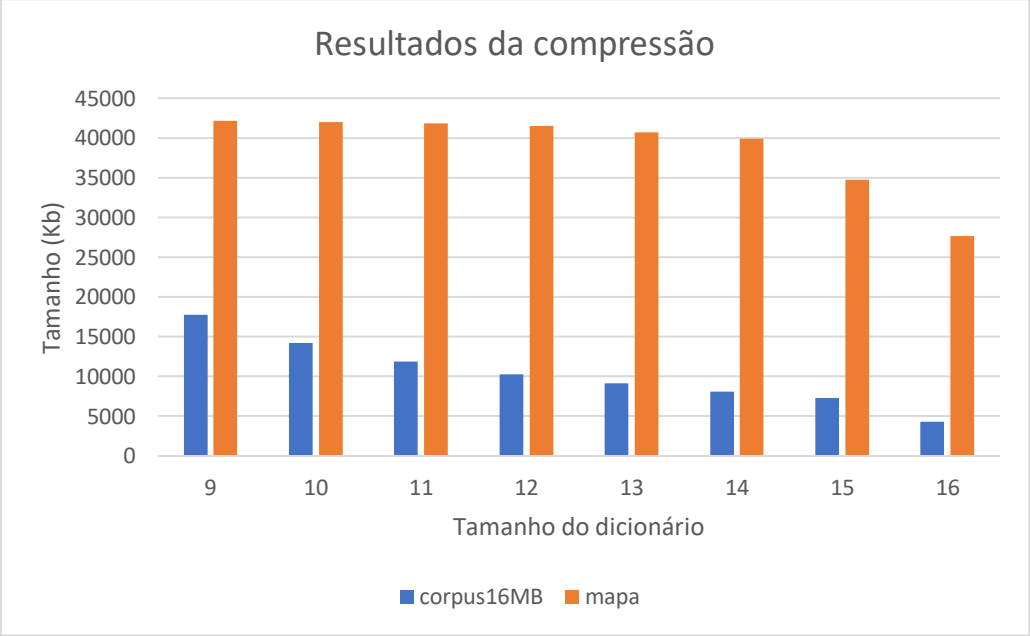

Figura 3 Tamanho x K

Agora, dividindo os resultados obtidos pelo tamanho original do arquivo, obtemos a razão de compressão para cada valor de K. Abaixo podemos observar os resultados.

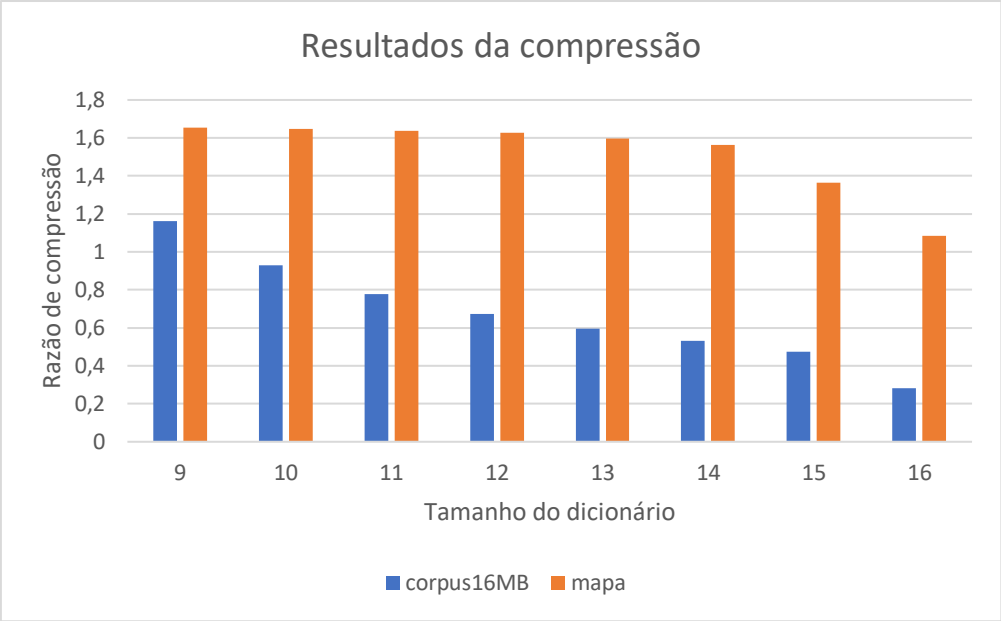

Figura 4 RC x K

Como dito anteriormente, todos os dados foram salvos utilizando 16bits, resultando em arquivos comprimidos com um tamanho um pouco maior do que o LZW pode comprimir. Abaixo podemos ver os gráficos apresentando os resultados obtidos caso a quantidade de bits utilizadas para salvar os arquivos fossem iguais ao valor K utilizado.

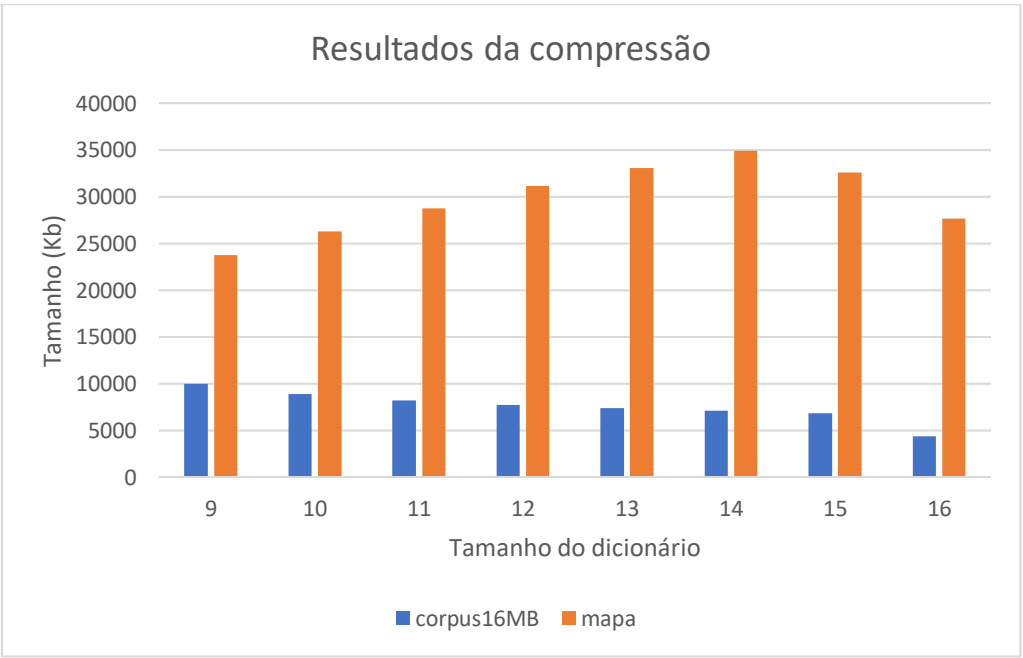

Figura 5 Tamanho x K com bits variantes

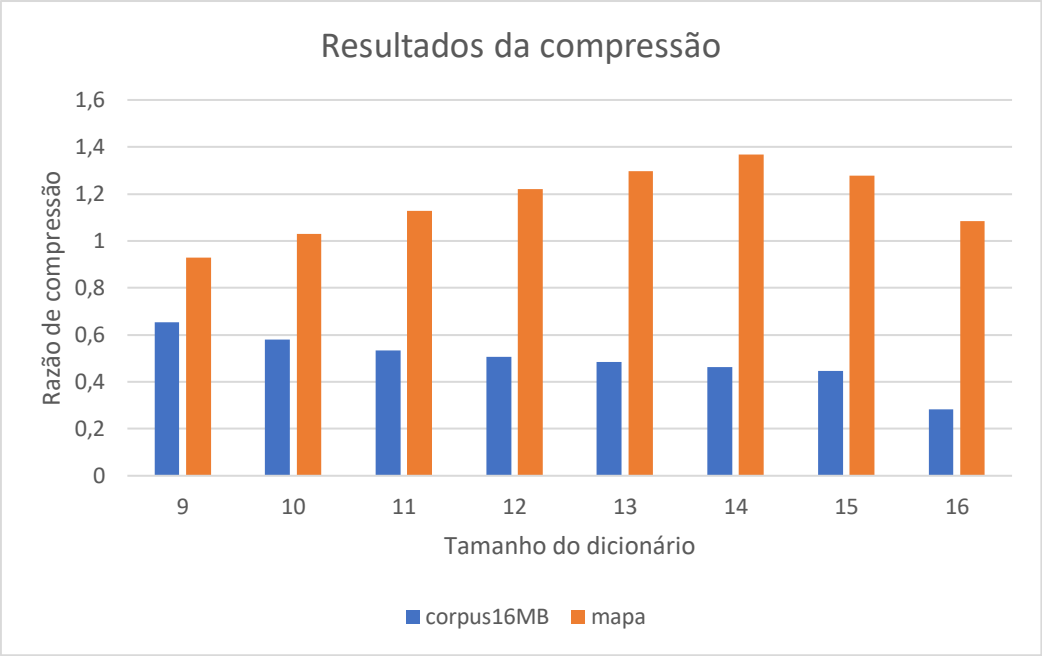

Figura 6 RC x K com bits variantes

Por fim, vamos analisar o tempo gasto para comprimir os arquivos. Diversos fatores podem alterar a quantidade de tempo necessária para a codificação/decodificação e para obtermos valores precisos, uma grande quantidade de testes deveriam terem sidos realizados. Porém, abaixo podemos ver o tempo gasto por cada K para codificar o arquivo de texto.

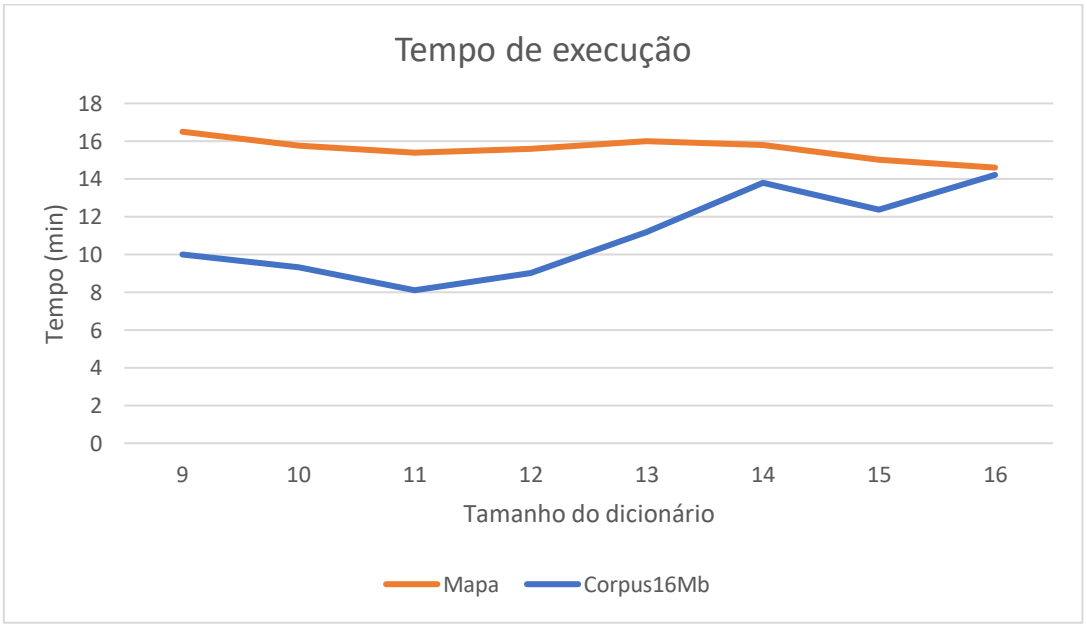

Figura 7 Tempo x K