Instituto Superior de Engenharia de Lisboa

Comunicação Digital

Relatório da Primeira Série de Problemas

Docente

Prof. Arthur Ferreira

Aluna

53255 Magali dos Santos Leodato

Índice

Intrdução

O presente relatório tem como objetivo apresentar os resultados obtidos no desenvolvimento de um conjunto de exercícios práticos relacionados à Teoria da Informação e Codificação. As atividades abordam temas fundamentais, como o uso de funções matemáticas básicas, análise de fontes de símbolos, geração e compressão de dados, além da aplicação de técnicas de cifra. As tarefas foram organizadas em seis seções, cada uma composta por subitens que envolvem tanto o raciocínio lógico quanto a implementação prática por meio de programação. Os códigos desenvolvidos foram entregues separadamente na linguagem Python e este documento visa descrever de forma clara os objetivos, métodos e resultados obtidos em cada etapa do trabalho.

Exercício 1 – Funções Matemáticas Básicas

(a) Múltiplos de quatro em um intervalo

Foi desenvolvida uma função que, ao receber dois valores inteiros (`left` e `right`), retorna todos os números múltiplos de 4 contidos nesse intervalo, inclusive os limites. A função percorre o intervalo e verifica quais números são divisíveis por 4, exibindo-os em uma lista.

$$n \bmod 4 = 0$$

Resultados:

```
PS C:\Users\magali\Desktop\paython> & C:/Users/magali/AppData/Local/Microsoft/Windows
o.py
Digite o início intervalo a ser considerado.10
Digite o fim do intervalo a ser considerado.60
Os números de quatro sao: [12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60]
PS C:\Users\magali\Desktop\paython>
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                   Python

PS C:\Users\magali\Desktop\paython> & C:/Users/magali/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/magali/Desktop/paython/calculo_multiplo_quatr
o.py
Digite o início intervalo a ser considerado.80
Digite o fim do intervalo a ser considerado.300
Os números de quatro sao: [80, 84, 88, 92, 96, 100, 104, 108, 112, 116, 120, 124, 128, 132, 136, 140, 144, 148, 152, 156, 160, 164, 168, 172, 176, 180, 184, 18
8, 192, 196, 200, 204, 208, 212, 216, 220, 224, 228, 232, 236, 240, 244, 248, 252, 256, 260, 264, 268, 272, 276, 280, 284, 288, 292, 296, 300]
PS C:\Users\magali\Desktop\paython>
```

(b) Máximo Divisor Comum (MDC)

Essa função tem como objetivo encontrar o maior número que divide dois inteiros simultaneamente. Foi utilizado o **algoritmo de Euclides**, conhecido por sua eficiência. O MDC é uma operação comum em problemas de simplificação de frações e divisibilidade.

$$\mathbf{MDC(a,b) = MDC(b, a \bmod b)}$$

O algoritmo continua até que o segundo número se torne zero.

Resultados:

```
Forneça o primeiro número.50
Forneça o segundo número.120
O maior divisor comum entre 50 e 120 é 10.
PS C:\Users\magali\Desktop\paython>
```

```
Forneça o primeiro número.98
Forneça o segundo número.322
O maior divisor comum entre 98 e 322 é 14.
PS C:\Users\magali\Desktop\paython>
```

(c) Progressão Geométrica

Criou-se uma função que gera os `N` primeiros termos de uma progressão geométrica (PG), utilizando como entrada o termo inicial `u` e a razão `r`. A PG é construída multiplicando o termo anterior pela razão, sucessivamente.

$$a_n = u \cdot r^{(n-1)}$$

Resultados:

```
Digite o número de termos: 5
Digite o primeiro termo: 1
Digite a razão: 2
1.0 2.0 4.0 8.0 16.0
```

```
Digite o número de termos: 10
Digite o primeiro termo: 1
Digite a razão: 4
1.0 4.0 16.0 64.0 256.0 1024.0 4096.0 16384.0 65536.0 262144.0
```

(d) Número de Combinações C(n, k)

Essa função calcula o número de maneiras diferentes de escolher `k` elementos de um conjunto com `n` elementos, usando a fórmula matemática de combinações. Foi feita a manipulação de fatoriais, considerando os cuidados necessários para evitar erros com entradas inválidas.

$$C(n,k) = \frac{n!}{k!(n-k)!}$$

Resultados:

```
Digite o valor de n: 12
Digite o valor de k: 5
C(12, 5) = 792
PS C:\Users\magali\Desktop\paython>
```

```
Digite o valor de n: 44
Digite o valor de k: 15
C(44, 15) = 229911617056
```

(e) Valor mínimo e máximo de um vetor

A função recebe um vetor (lista) com números e retorna os menores e maiores valores contidos nesse vetor. O procedimento é simples, mas muito útil em estatística e análise de dados.

Resultados:

```
PS C:\Users\magali\Desktop\paython> & C:/Users/magali/AppData/Loc
mo_vetor.py
Digite os números do vetor separados por espaço: 1 2 13 4 9
Valor mínimo: 1, Valor máximo: 13
PS C:\Users\magali\Desktop\paython>
```

```
Digite os números do vetor separados por espaço: 3 8 32 5 6 4
Valor mínimo: 3, Valor máximo: 32
```

Exercício 2 – Análise de Fontes de Símbolos

(a) Símbolo mais frequente

Foi feita a análise de arquivos do conjunto `TestFilesCD.zip`, determinando o símbolo (caractere) mais frequente em cada arquivo. A probabilidade de ocorrência foi calculada dividindo a frequência pelo número total de símbolos. A informação própria foi obtida utilizando a fórmula de Shannon:

$$I(x) = -\log_2(p(x))$$

(b) Entropia

A entropia representa a quantidade média de informação por símbolo. Para cada arquivo, a entropia foi calculada considerando a distribuição de probabilidades dos símbolos presentes. Isso indica o grau de desordem ou aleatoriedade do conteúdo.

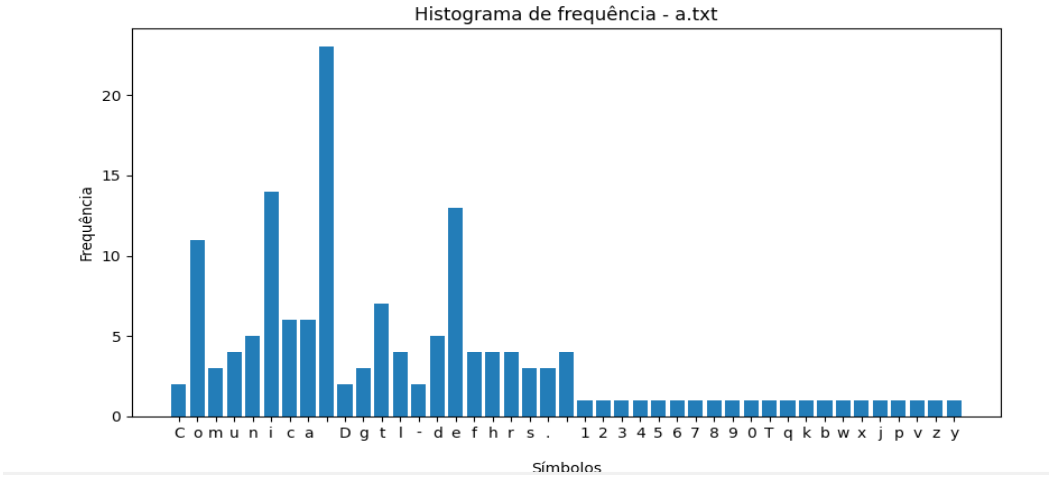
Resultados:

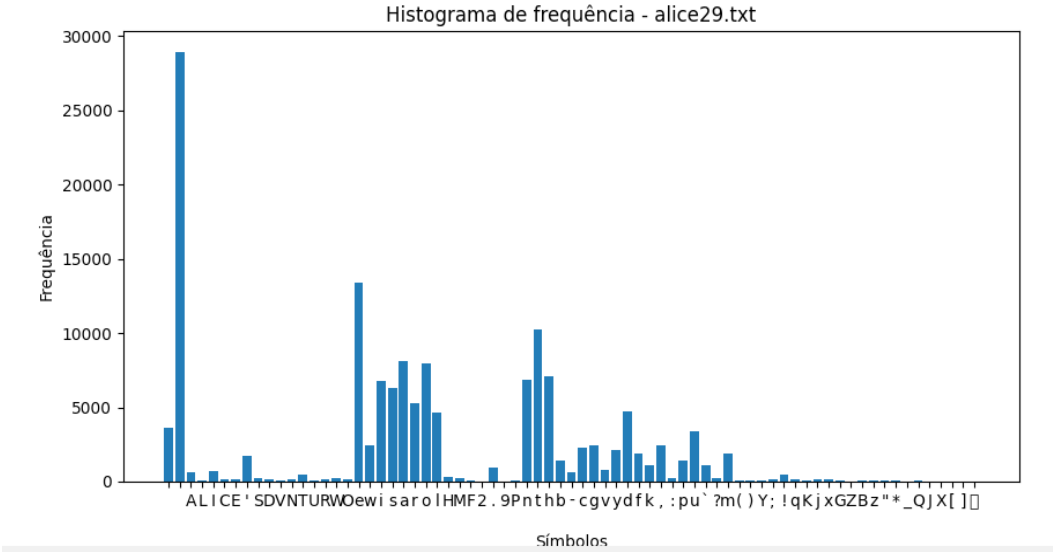
```
Arquivo: a.txt
  Símbolo mais frequente: 32 (byte)
  Probabilidade: 0.1420
  Informação própria: 2.8163 bits
  Entropia: 4.8558 bits/símbolo

Arquivo: alice29.txt
  Símbolo mais frequente: 32 (byte)
  Probabilidade: 0.1900
  Informação própria: 2.3957 bits
  Entropia: 4.5677 bits/símbolo

Arquivo: arrays.kt
  Símbolo mais frequente: 32 (byte)
  Probabilidade: 0.3418
  Informação própria: 1.5490 bits
  Entropia: 4.3375 bits/símbolo

Arquivo: barries.jpg
  Símbolo mais frequente: 0 (byte)
  Probabilidade: 0.0086
  Informação própria: 6.8580 bits
  Entropia: 7.9687 bits/símbolo

Arquivo: barries.tif
  Símbolo mais frequente: 244 (byte)
  Probabilidade: 0.0120
  Informação própria: 6.3786 bits
  Entropia: 7.6494 bits/símbolo

Arquivo: bird.gif
  Símbolo mais frequente: 2 (byte)
  Probabilidade: 0.0093
  Informação própria: 6.7515 bits
  Entropia: 7.8944 bits/símbolo
```

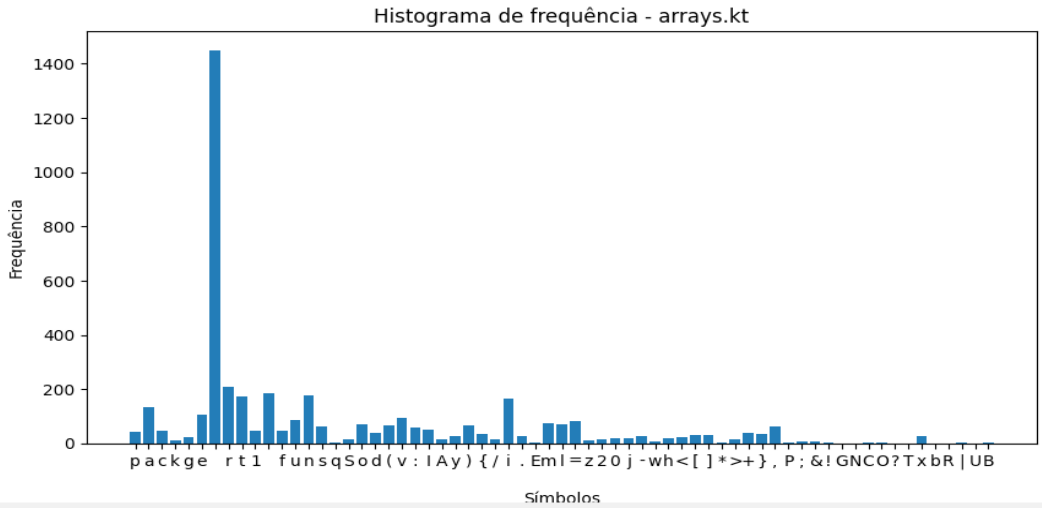
```
Arquivo: bird.gif
  Símbolo mais frequente: 2 (byte)
  Probabilidade: 0.0093
  Informação própria: 6.7515 bits
  Entropia: 7.8944 bits/símbolo
```

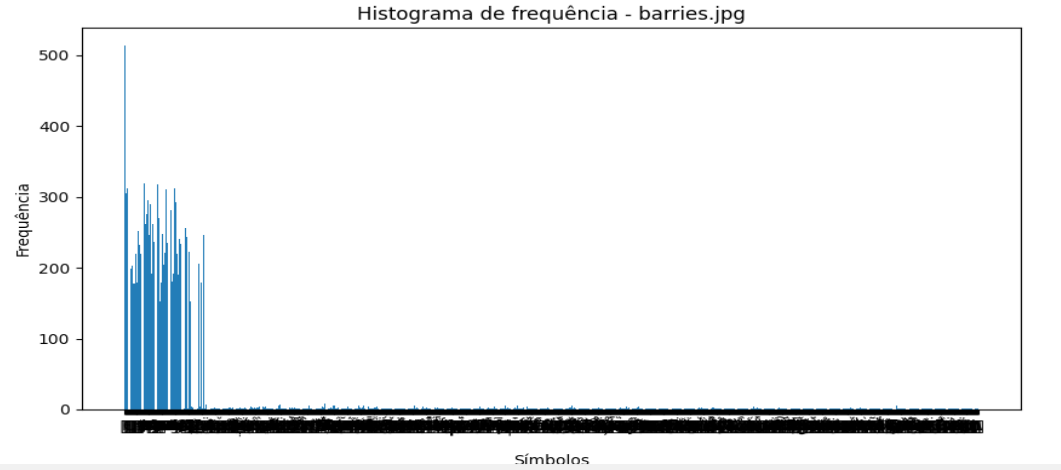
(c) Histograma

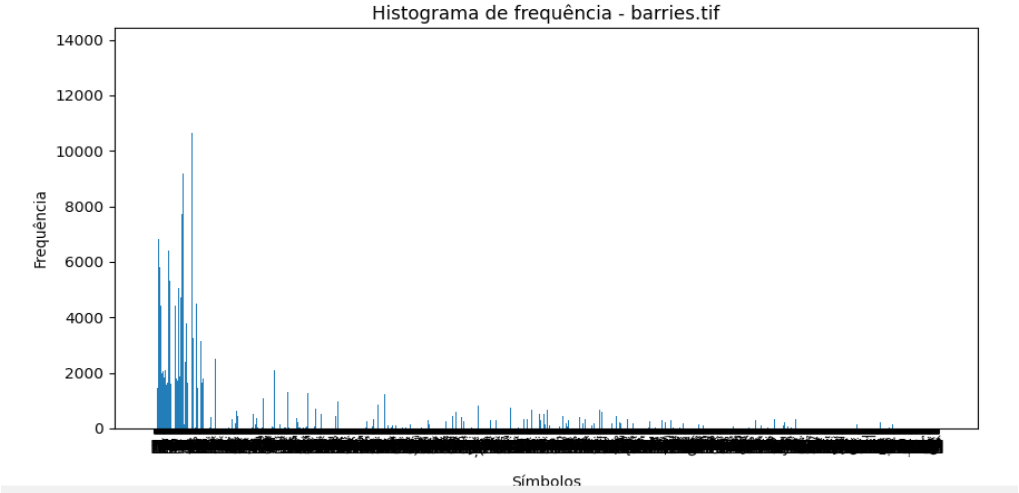
Foi gerado um histograma para cada arquivo, onde o eixo X representa os símbolos (ou seus códigos ASCII) e o eixo Y representa sua frequência. Isso permite visualizar rapidamente a distribuição dos símbolos.

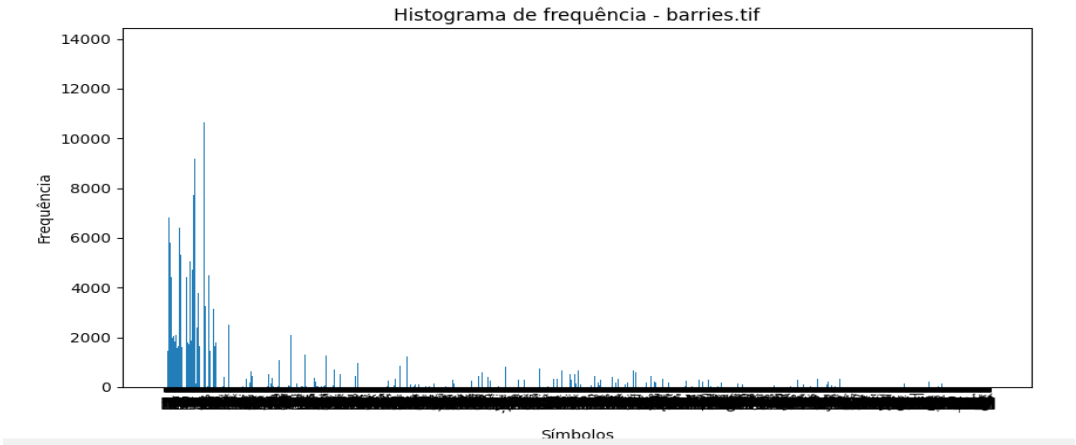


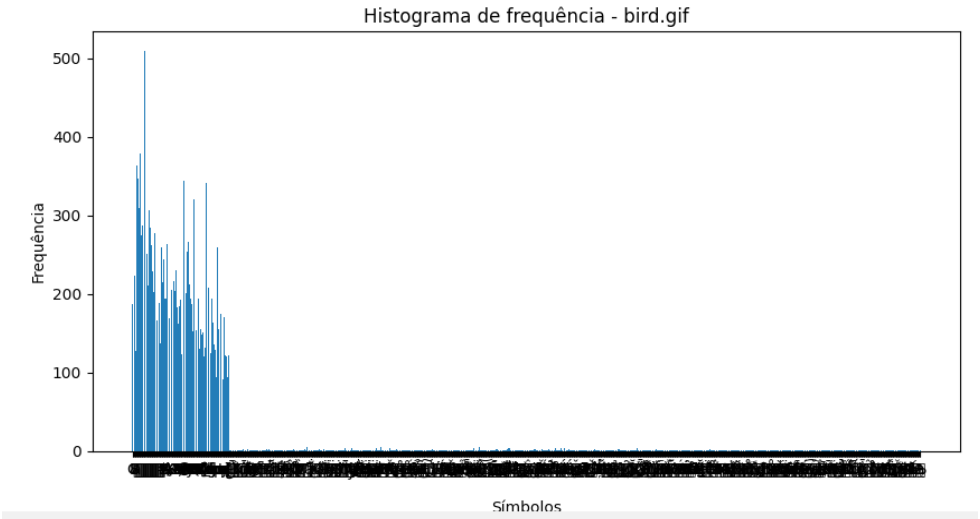


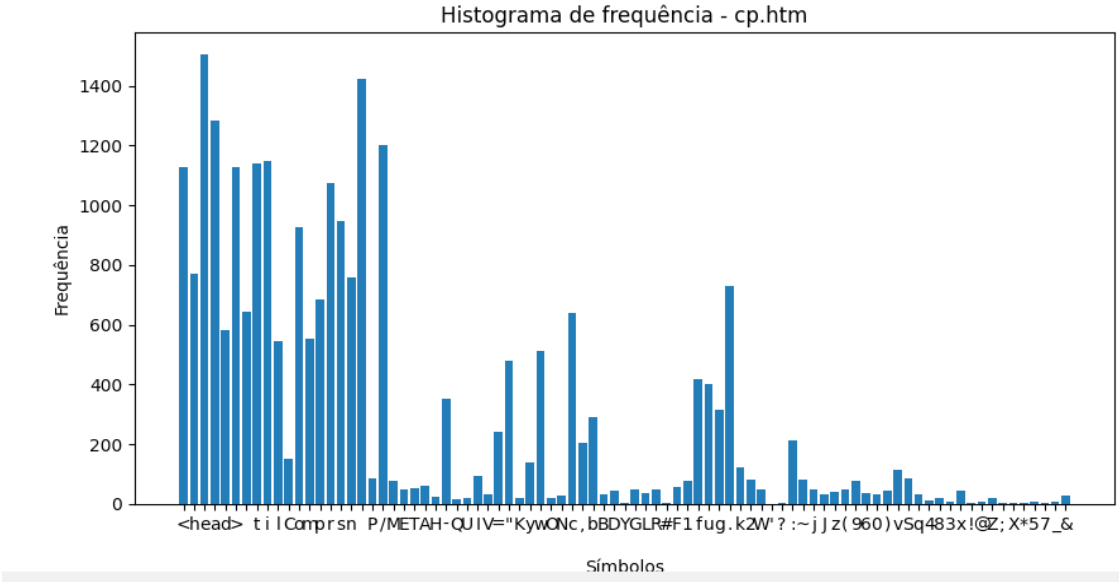

Histograma de frequência - arrays.kt
Frequência
Símbolos
0
200
400
600
800
1000
1200
1400
p a c k g e r t 1 f u n s q S o d (v : I A y) { / i . E m l = z 2 0 j - w h < [ ] * > + } , P ; & ! G N C O ? T x b R | U B

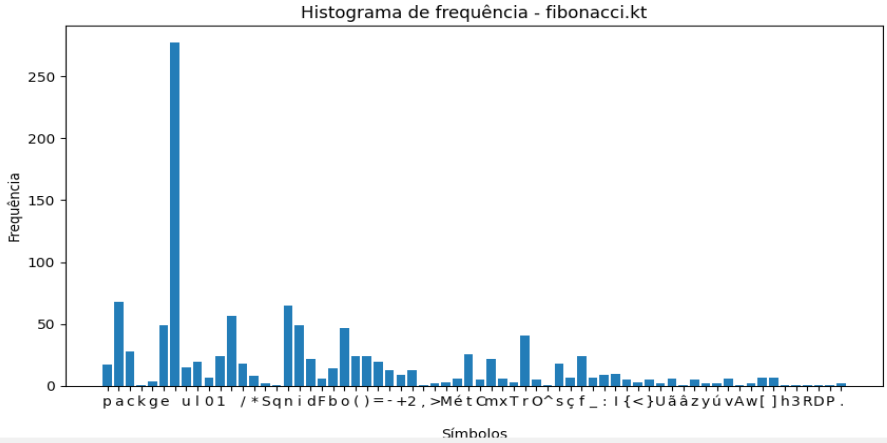

Histograma de frequência - barries.jpg
Frequência
Símbolos
0
100
200
300
400
500

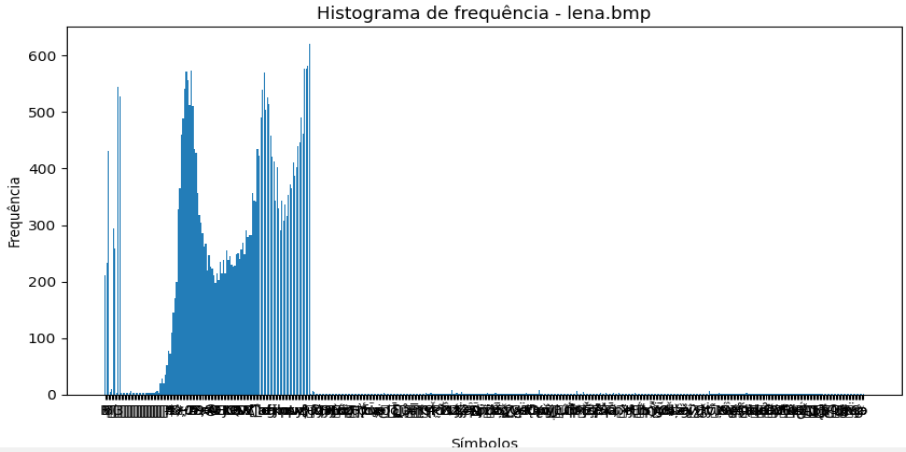

Histograma de frequência - barries.tif
Frequência
Símbolos
0
2000
4000
6000
8000
10000
12000
14000

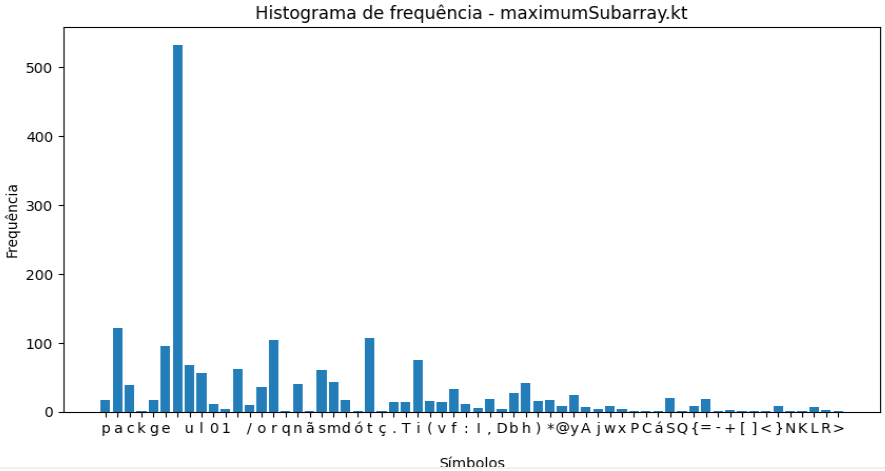

Histograma de frequência - barries.tif
Frequência
Símbolos
0
2000
4000
6000
8000
10000
12000
14000

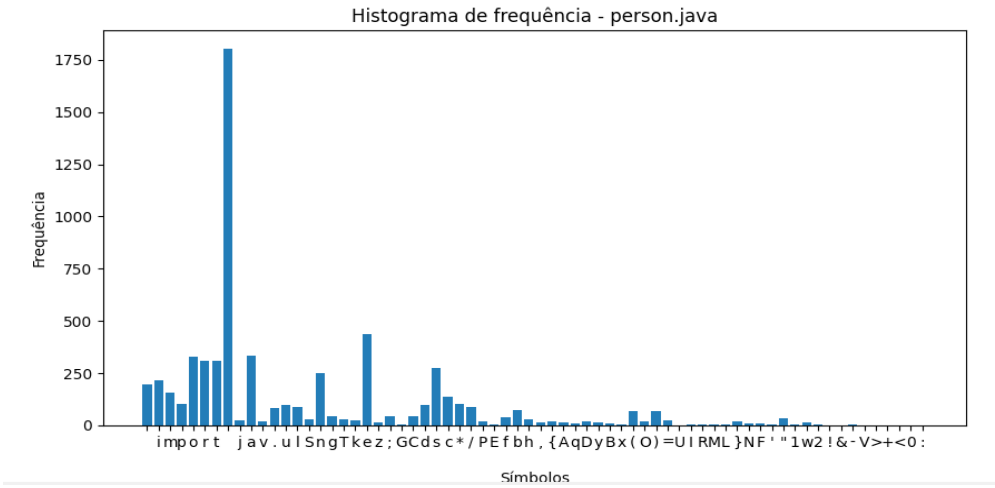

Histograma de frequência - bird.gif
Frequência
Símbolos
0
100
200
300
400
500

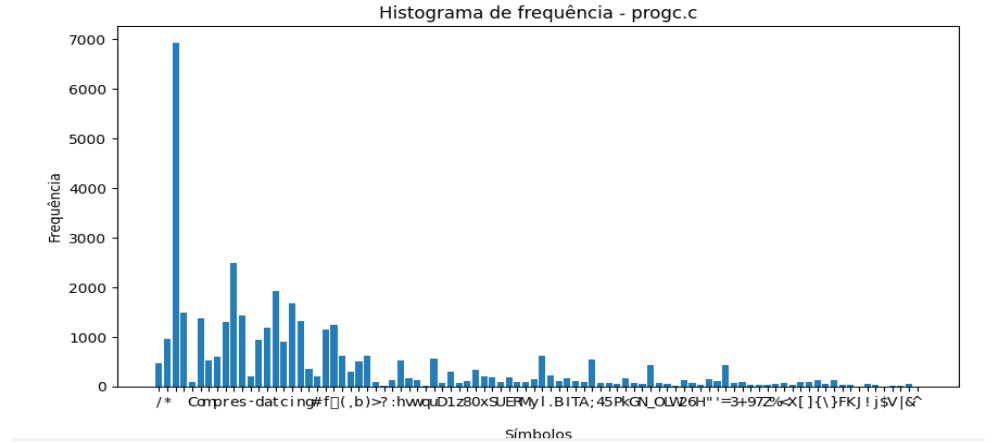

Histograma de frequência - cp.htm
Frequência
Símbolos
0
200
400
600
800
1000
1200
1400
<head> t i l Comprsn P/METAH-QUIV="KywQNc,bBDYGLR#F1fug.k2W'?:~jJz(960)vSq483x!@Z;X*57_&

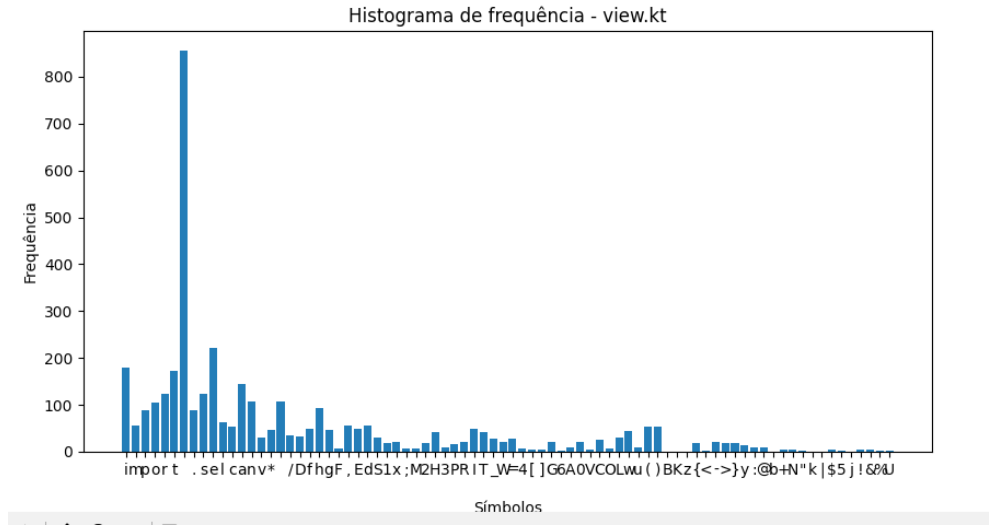

Histograma de frequência - fibonacci.kt
Frequência
0
50
100
150
200
250
p a c k g e u l 0 1 / * S q n i d F b o ( ) = - + 2 , > M é t C m x T r O ^ s ç f _ : l { < } U ã â z y ú v A w [ ] h 3 R D P .
Símbolos


Histograma de frequência - lena.bmp
Frequência
0
100
200
300
400
500
600
Símbolos


Histograma de frequência - maximumSubarray.kt
Frequência
0
100
200
300
400
500
p a c k g e u l 0 1 / o r q n ã s m d ó t ç . T i (v f : l , D b h) * @ y A j w x P C á S Q { = - + [ ] < } N K L R >
Símbolos


Histograma de frequência - person.java
Frequência
Símbolos
import javaulSngTkez;GCdsc*/PEfbh,{AqDyBx(O)=UIRML}NF'"1w2!&-V>+<0:


Histograma de frequência - progc.c
Frequência
Símbolos
/* Compres-datcing#f(,b)>?:hwwquD1z80xSUERMyl.BITA;45PkGN_OLW26H"'=3+97Z%-X[]{\}FKJ!j$V|&^


Histograma de frequência - view.kt
Frequência
Símbolos
import .selcanv* /DfhgF,EdS1x;M2H3PRIT_W=4[]G6A0VCOLwu()BKz{<->}y:@b+N"k|$5j!&%U

Exercício 3 – Implementação de Fontes de Símbolos

(a) Fonte genérica com distribuição definida

Implementou-se uma função que, dada uma lista de probabilidades para `M` símbolos e o número total `N` de símbolos desejados, gera um arquivo com essa distribuição. Isso foi feito com sorteios aleatórios ponderados. Para cada ficheiro, foi determinado o símbolo com maior frequência. A **probabilidade de ocorrência** é dada por:

$$p(x) = \frac{\text{frequência de } x}{\text{número total de símbolos}}$$

A **informação própria** de um símbolo é dada por:

$$I(x) = -\log_2(p(x))$$

Entropia:

A entropia representa a quantidade média de informação por símbolo. Para cada arquivo, a entropia foi calculada considerando a distribuição de probabilidades dos símbolos presentes. Isso indica o grau de desordem ou aleatoriedade do conteúdo.

$$H(X) = -\sum_{i=1}^{n} p(x_i) \cdot \log_2(p(x_i))$$

Onde:

- $p(x_i)$ é a probabilidade do símbolo x_i
- n é o número de símbolos distintos

Implementou-se uma função que, dada uma lista de probabilidades para `M` símbolos e o número total `N` de símbolos desejados, gera um arquivo com essa distribuição. Isso foi feito com sorteios aleatórios ponderados.

(b) Geração de formatos específicos

A fonte genérica foi utilizada para gerar:

- **Senhas robustas**, com letras, números e símbolos, contendo de 10 a 14 caracteres.
- **Endereços IPv4**, no formato `X.X.X.X` com números de 0 a 255.
- **Endereços IPv6**, no formato hexadecimal com blocos separados por `:`.
- **Tuplos hexadecimais** com 8 elementos, simulando registros de memória ou identificadores.

Foram gerados 10 exemplos para cada tipo.

(c) Geração de três ficheiros com entropia controlada

Foram gerados três ficheiros com características específicas de entropia:

- **Ficheiro 1:** 100 símbolos com 4 possíveis valores, entropia ≈ 1.75
- **Ficheiro 2:** 1000 símbolos com os mesmos 4 valores e entropia ≈ 1.75
- **Ficheiro 3:** 10000 símbolos com 256 símbolos diferentes e entropia ≈ 4.0

As probabilidades foram ajustadas cuidadosamente para atingir os valores desejados.

Resultados:

ficheiro1.txt

```
1  DCCACCCADDABCDAADACDCDDCCDAABACBCACCDCDBAAACDCCDACCDADDACDADDADAADDDDCCDDADAADDADACDDCCCCCCACACACAAA
```

ficheiro2.txt

```
1  BDACCDDACDCADCACDACACCCCCADDACADACCACCCDCDDCDDDDDDADCCDDCACCACDDADACDCACCDACCDCCACCADDAADCCDCCCADDDADCDADAACACCAACCAAACDCAB
```

ipv4.txt

```
1   146.132.103.46
2   222.25.1.71
3   59.76.20.241
4   192.85.135.56
5   131.40.146.239
6   226.22.238.44
7   75.25.175.50
8   143.185.237.100
9   180.82.164.35
10  112.5.152.178
11
```

ipv6.txt

```
1   cc4a:f1cc:c97d:a55d:e278:1118:78dd:bab6
2   e72e:ad81:1ef5:340f:8c61:008c:5fce:e444
3   0a5f:7576:031a:8ea9:5278:2e75:c61c:6bd9
4   259f:5cad:cf47:f684:b414:b430:09f9:752a
5   eba6:9a1c:0324:75be:1cf7:fd6d:61a9:71c5
6   c1a5:1e29:888c:d53d:d109:b9af:75a9:f8c2
7   78ba:efc4:67e0:2540:86ac:a2aa:4428:9ace
8   68db:c9e4:c6f6:0011:b59d:b711:978f:e04f
9   d9e4:eddb:be11:d91f:9e07:49e3:0a09:41b6
10  47cd:4c25:ef84:ad2f:969b:3a8c:c2b7:f6bd
11
```

senhas.txt

```
1   6YQ]3)~0f.r:;
2   Uy>92&nWZd5
3   SD]>_4;1!K\OSy
4   bU~3_jDXaHYd
5   vxsN#xk2Cl:)"
6   4J)Ap^lN`zEbu
7   $5wm-]:ju}La
8   <i>FJEc*<o~y
9   4CI$@zp]DE,
10  V*Cuau+&0\d
11
```

tuplos_hex.txt

```
1   F8, 34, 3B, F6, 5E, 76, 74, ED
2   80, FF, 5F, 17, 02, 13, 49, 9B
3   5F, 44, F0, AC, 43, 80, C3, CD
4   06, FC, DC, 99, 43, 96, D0, CA
5   D4, 20, 63, 55, EB, 3C, E8, 2F
6   BA, 28, 9D, 5C, 6F, BB, D2, 7C
7   A9, 41, 46, DF, 3E, 9C, F3, CC
8   80, B6, 60, 25, CC, 3E, 1F, 42
9   16, 50, BC, 06, 69, B0, 92, 61
10  E7, FC, EC, 17, 9B, E8, B4, 42
11
```

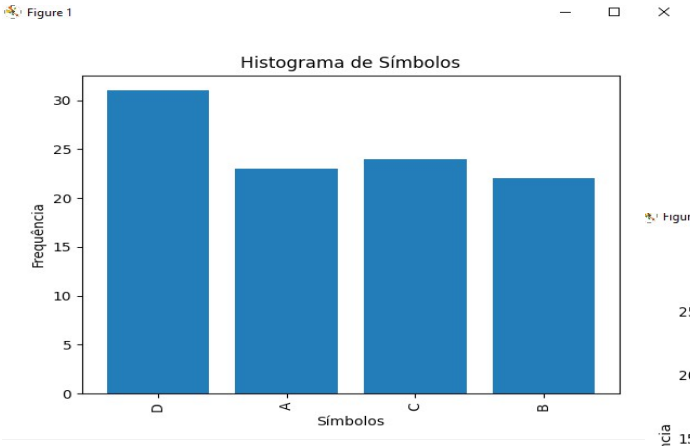
ficheiro3.txt

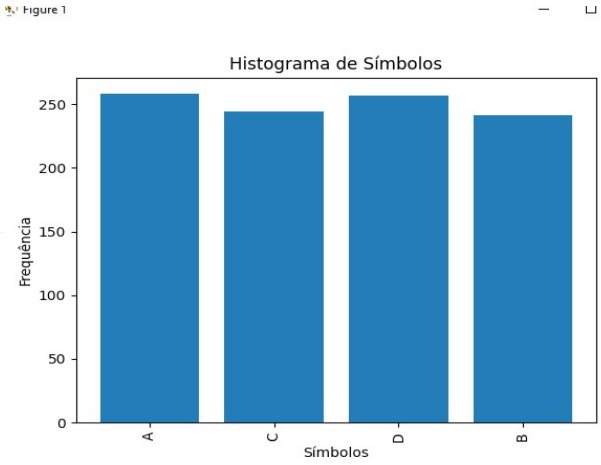
C:\Users\magali\Desktop\paython\ficheiro3.txt

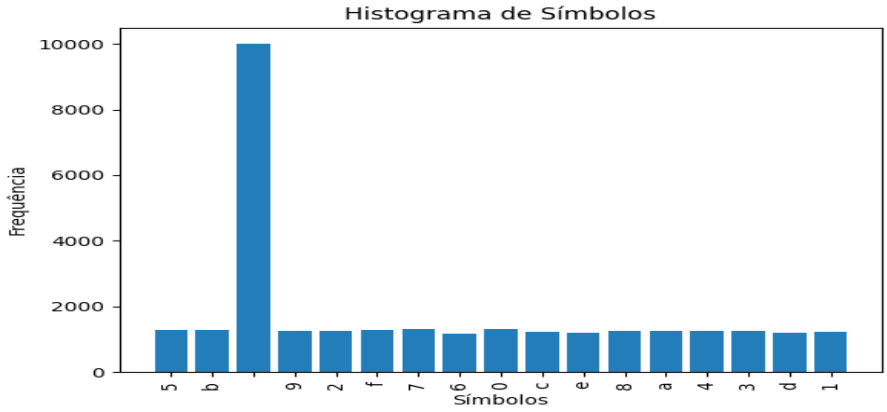
```
29 1c d3 c9 7b 7e a7 c0 d4 ae 41 ec 9f 7a d5 ca a6 27 cb 2a 1b be 75 7b f8 71 c6 bc e2 05
78 7b b6 36 ae e1 f2 53 84 04 30 ff 64 9d d5 a1 62 c1 ef 1c 5c bf 82 02 1d 28 c9 cd 94 f1 c5 24 1d 6a 64 86 27 ac a1 5b
96 d7 c3 0e 8f 27 fe 72 95 55 c2 e2 52 62 dd 46 64 03 4f 48 df 06 c6 74 b1 4c 64 10 3e 81 a1 e3 7f 6d e3 8f 67 cd 08 0d
a9 70 9b 78 13 91 01 1d 92 a4 54 5b af 94 b9 8f 13 bc c8 b2 2e 35 70 a6 29 ee 61 90 c1 36 33 cf 55 78 fe 6b 1c 95 ce 0a
92 d9 0e a5 f0 15 5b 25 a3 26 93 98 a5 be 3b 57 65 7b bd e6 93 5b 3e e1 9f 98 77 56 9e 10 41 77 86 fc f6 20 7d 30 57 d7
08 52 d8 52 62 4c 46 54 51 eb e6 36 06 ec 73 0c fb b8 6f fb d9 7b 8a 4d 9a 7a 8d 55 bf 21 4e 64 2d b9 32 cd 51 06 2f 5a
37 f0 c8 f2 94 33 d6 83 ef 09 ad 93 55 90 42 72 b5 a0 94 da 9c 7b d4 91 37 b7 b0 61 ea e0 1d 64 1f f4 96 df c5 07 c1 5c
9b 1b 1e 25 c9 5d 41 2a 2a 34 13 b5 0d b2 2c 8a c4 e5 ae 60 e3 4a 87 8f 19 c8 f6 e2 7c d3 23 d2 56 76 1c b0 27 ac 07 7b
e7 07 56 3d db 48 1f 2d 8b 9e 68 b2 86 4a c8 ff 04 a7 5e a0 ef 47 a6 48 01 35 b2 c8 ff 4a ba 88 a5 e3 30 6b ff 9b 47 b6
68 a5 c8 f5 e6 61 a9 91 e9 52 09 fb 99 c2 c2 05 ab 4f 29 fa 0d 21 d5 9c 8e 8e 6e 66 f6 1b c4 7a 81 af ae 21 2c b1 42 29
ac 02 e5 7a bf 35 d7 ea 6e 2d ce 7f 49 bd 41 ee 69 e0 f0 44 f4 64 08 80 e2 14 b0 14 4b 38 06 5a f1 d4 65 8e de b8 8c 26
dd bd e6 75 3f 28 9f ad c2 9c a9 b6 3c 66 47 9c 7a 25 80 16 5a cc 56 c0 93 0f fa 90 75 a0 28 41 34 27 a6 ea 65 66 82 75
aa 83 79 45 c2 44 26 47 01 de b8 cb 22 1d 72 dc 5b 35 18 e0 13 1d 08 92 41 32 b5 7e 4b 8b ff 06 cf ff c8 09 ac bb d4 57
c5 98 85 9c 55 0e 9c bb c3 8c c3 5c 02 11 04 5f 72 e6 2c 35 59 39 88 66 fc e1 ad 99 ef fe 6f 53 f6 8a 26 ed 4c 19 3d 3f
```

(d) Histograma e entropia dos ficheiros

Cada ficheiro gerado teve sua entropia e histograma calculados. Os resultados mostraram que os valores calculados se aproximam dos teóricos esperados, comprovando que a geração de símbolos foi feita corretamente.







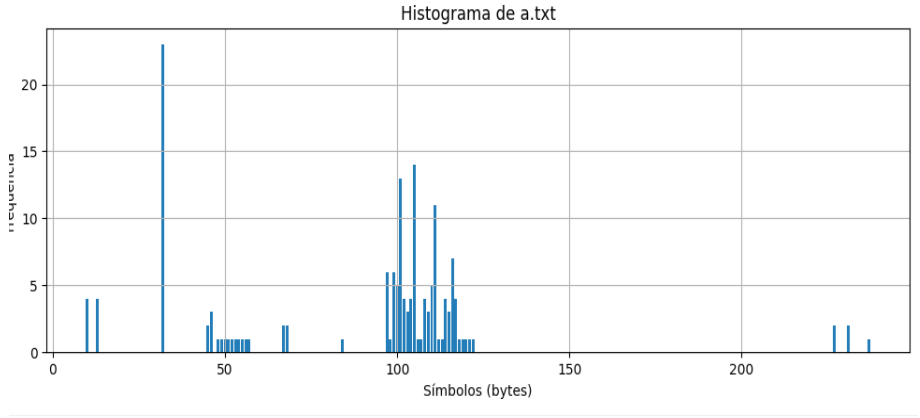
Exercício 4 – Compressão de Dados

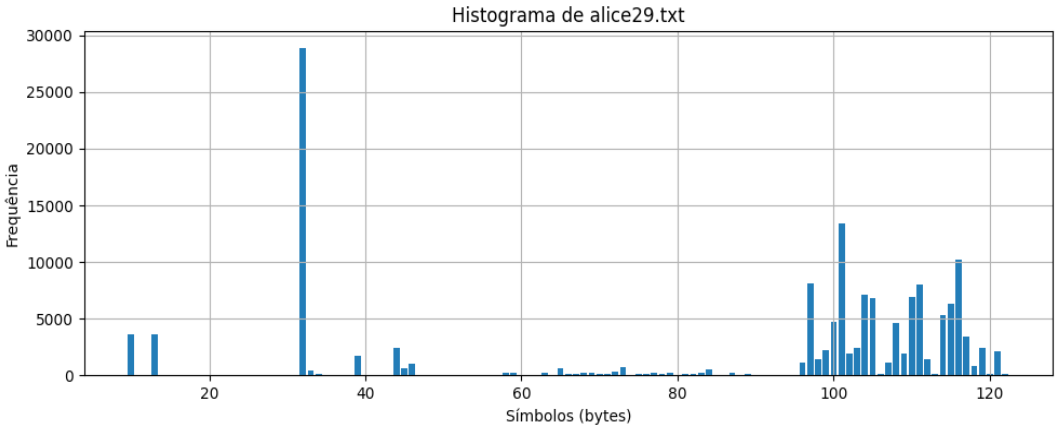
(a) Compressão e entropia

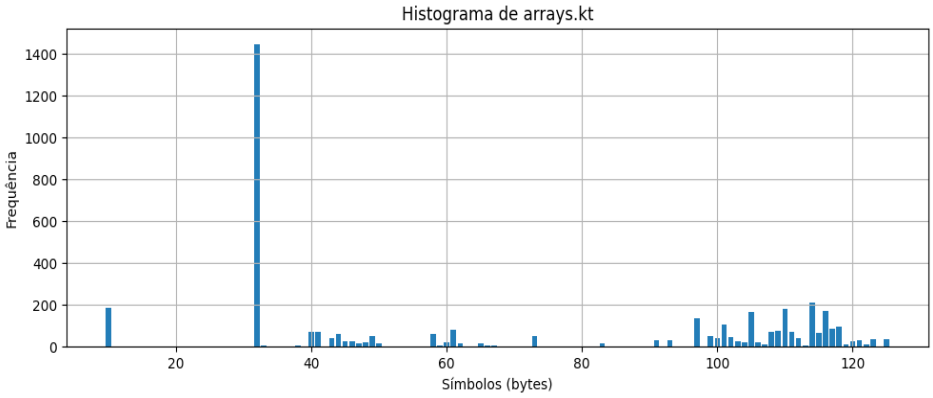
Cada ficheiro do **Canterbury Corpus** teve sua entropia calculada. Em seguida, foi comprimido utilizando uma ferramenta como **7-Zip**. A razão de compressão foi determinada comparando os tamanhos antes e depois da compressão.

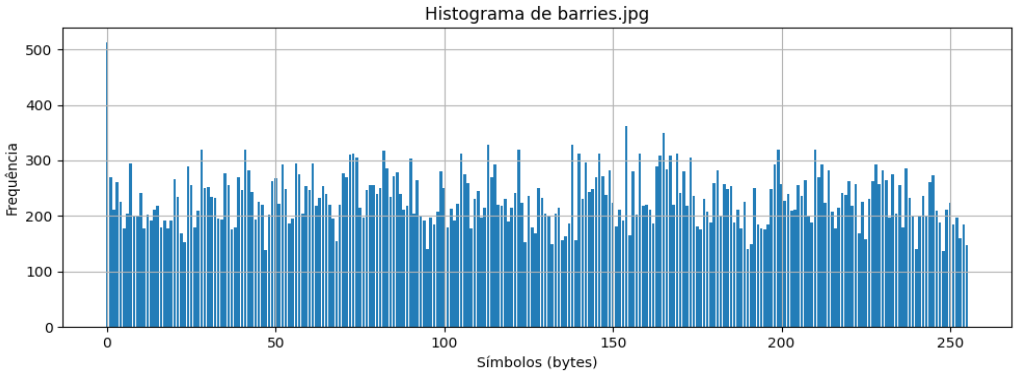
(b) Gráfico: Entropia vs Razão de compressão

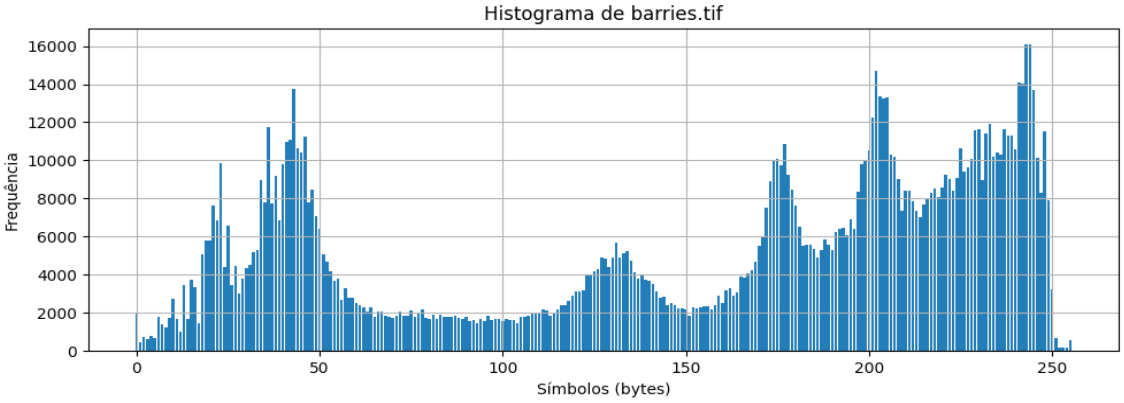
Foi gerado um gráfico com a entropia no eixo X e a razão de compressão no eixo Y. Como esperado, os arquivos com **menor entropia** foram mais facilmente comprimidos, demonstrando a relação entre **redundância da informação** e **compressibilidade**.

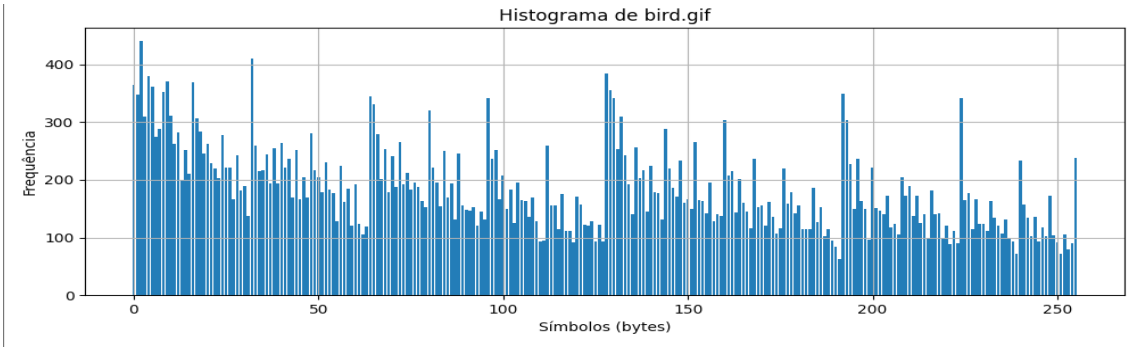


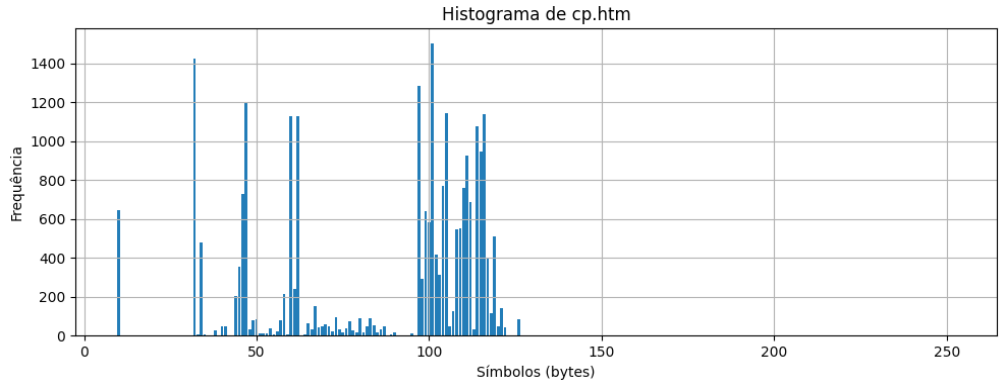


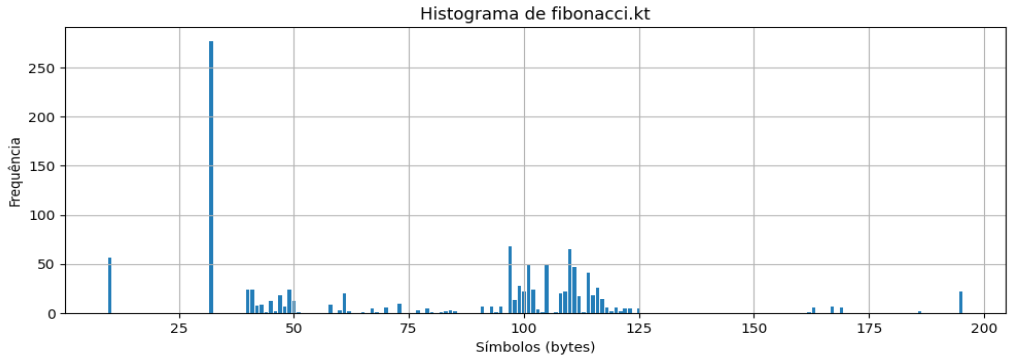

Histograma de arrays.kt
Frequência
Símbolos (bytes)
0
200
400
600
800
1000
1200
1400
20
40
60
80
100
120

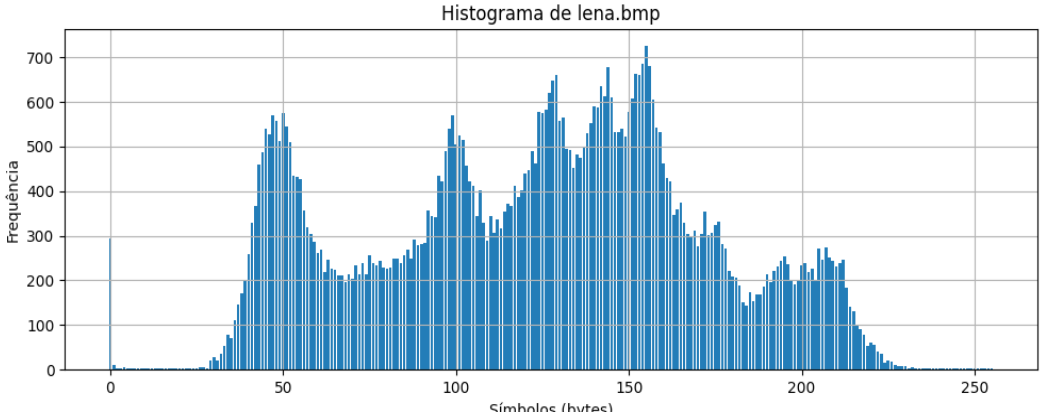

Histograma de barries.jpg
Frequência
Símbolos (bytes)
0
100
200
300
400
500
0
50
100
150
200
250

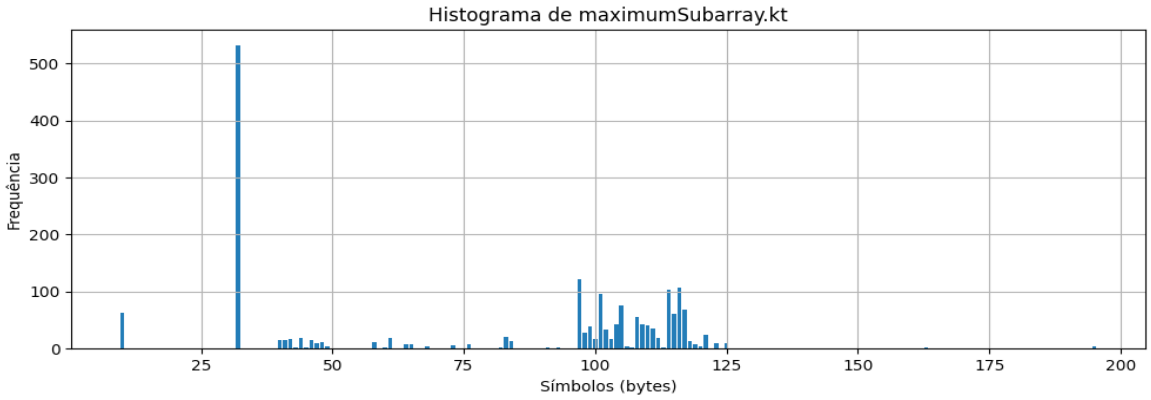

Histograma de barries.tif
Frequência
Símbolos (bytes)
0
2000
4000
6000
8000
10000
12000
14000
16000
0
50
100
150
200
250

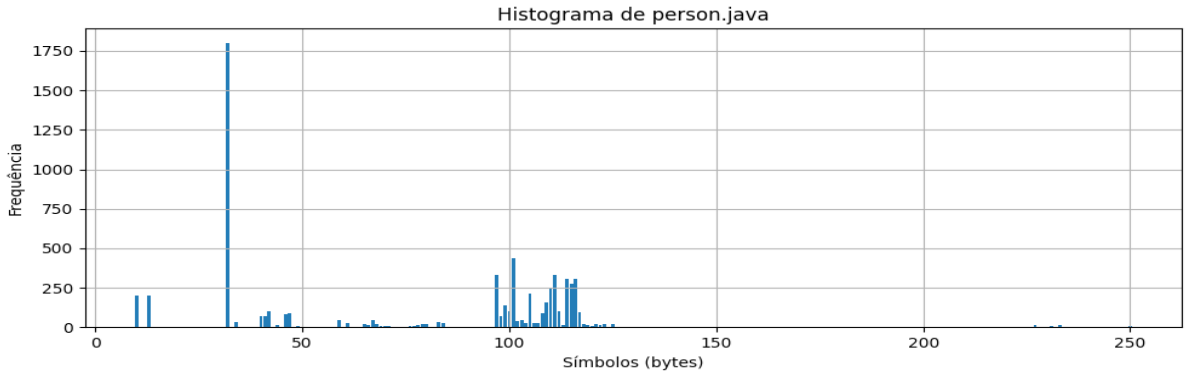

Histograma de bird.gif
Frequência
Símbolos (bytes)
0
100
200
300
400
0
50
100
150
200
250

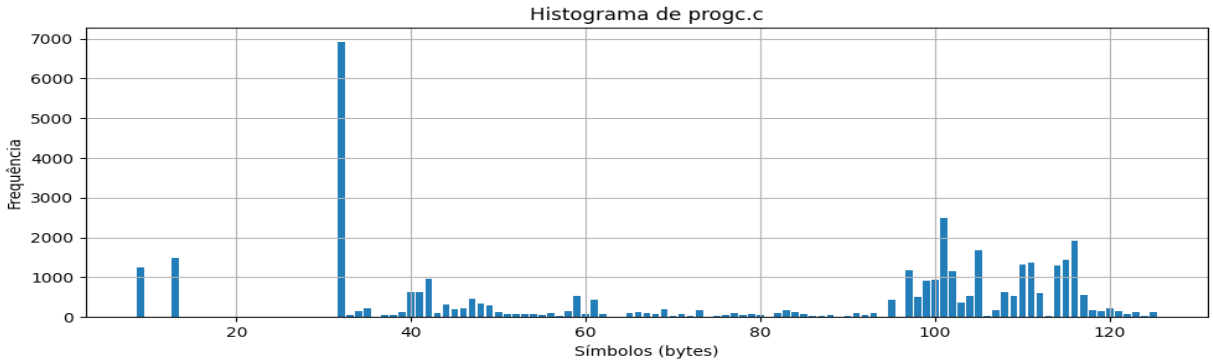

Histograma de cp.htm
Frequência
Símbolos (bytes)
0
200
400
600
800
1000
1200
1400
0
50
100
150
200
250

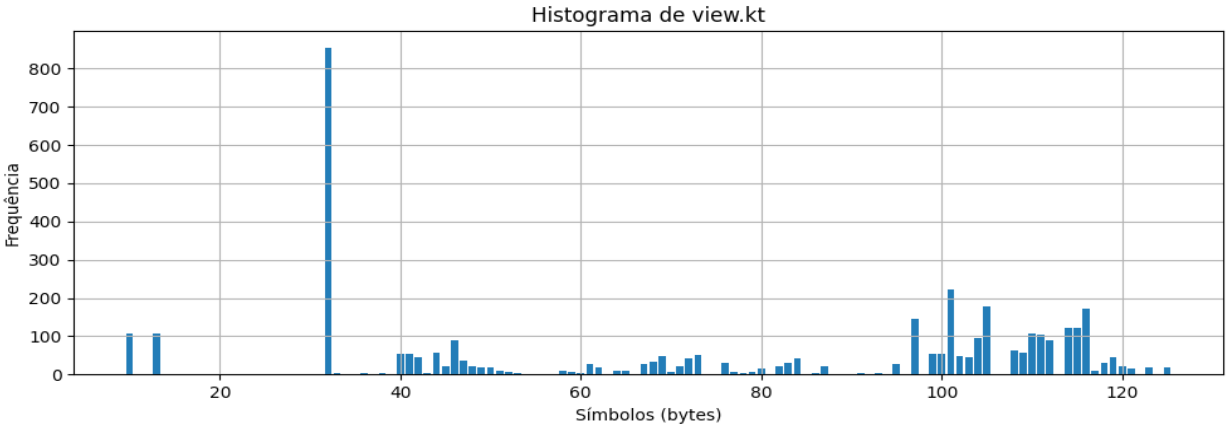

Histograma de fibonacci.kt
Frequência
Símbolos (bytes)
0
50
100
150
200
250
25
50
75
100
125
150
175
200


Histograma de lena.bmp
Frequência
0
100
200
300
400
500
600
700
0
50
100
150
200
250


Histograma de maximumSubarray.kt
Frequência
Símbolos (bytes)
0
100
200
300
400
500
25
50
75
100
125
150
175
200


Histograma de person.java
Frequência
Símbolos (bytes)
0
250
500
750
1000
1250
1500
1750
0
50
100
150
200
250


Histograma de progc.c
Frequência
Símbolos (bytes)
0
1000
2000
3000
4000
5000
6000
7000
20
40
60
80
100
120


Histograma de view.kt
Frequência
Símbolos (bytes)
0
100
200
300
400
500
600
700
800
20
40
60
80
100
120

Exercício 5 – Cifra de Vigenère

(a) Funções de cifra e decifra

Foi implementado o algoritmo da cifra de Vigenère, que utiliza uma palavra-chave para cifrar e decifrar o conteúdo de um arquivo. A cifra é baseada em operações de soma e subtração no alfabeto.

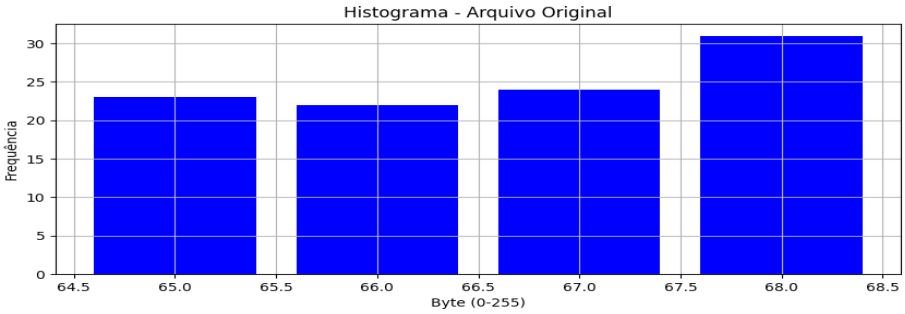
(b) Teste com arquivos reais

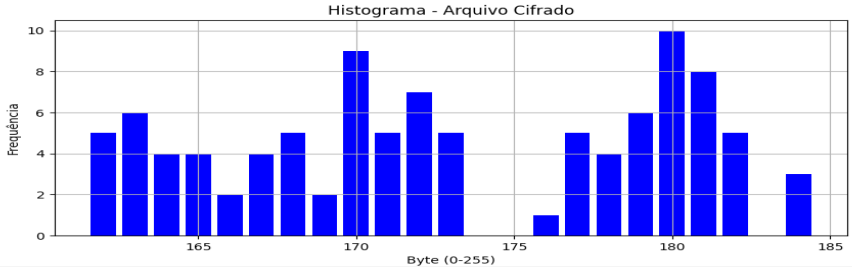
Dois arquivos foram cifrados e, em seguida, decifrados com a mesma chave. Os histogramas e entropias foram comparados nas três versões: **texto claro**, **texto cifrado** e **texto decifrado**. Os resultados demonstraram que a cifra altera completamente o padrão dos símbolos, aumentando a entropia, enquanto a decifra recupera fielmente o conteúdo original.

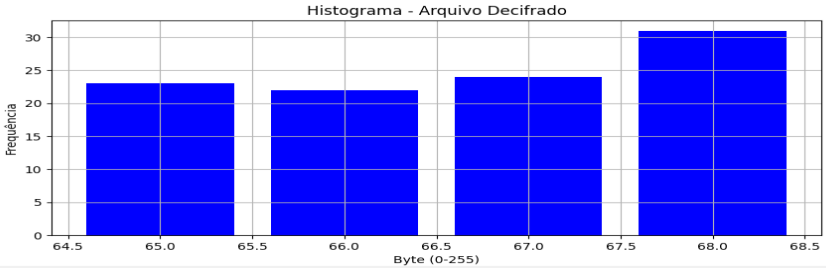
Resultados:

```
Entropia do arquivo original:   1.9862
Entropia do arquivo cifrado:    4.1755
Entropia do arquivo decifrado:  1.9862
```

Histograma







Conclusão

Este trabalho prático proporcionou uma aplicação ampla dos conceitos fundamentais da Teoria da Informação. Através de funções matemáticas básicas, geração de dados com controle de entropia, análise estatística e compressão de arquivos, foi possível compreender como a informação é estruturada, manipulada e protegida. As implementações mostraram-se eficazes, e os resultados obtidos nos testes experimentais validam os objetivos propostos. A etapa final, com a cifra de Vigenère, ilustrou bem a importância da segurança da informação, reforçando o valor da codificação na proteção de dados sensíveis. O trabalho também contribuiu para o desenvolvimento das competências de programação, análise crítica e representação gráfica.