Instituto Superior de Engenharia de Lisboa
Codificação de Sinais Multimédia

2º Semestre de 2020/2021

O ficheiro **Jupyter Notebook** com o relatório e código implementado deve submetido no Moodle até **2 de Maio**. O nome do ficheiro deve ter o seguinte formato: `T41Gxx_TP2.ipynb` ou `T42Gxx_TP2.ipynb` ou `T4NGxx_TP2.ipynb`, consoante a turma que estiverem a frequentar onde `xx` se refere ao número do grupo (consultar "grupos por turma" na página Moodle da disciplina).

Este Trabalho explora os conceitos de compressão de dados sem perdas baseados na teoria da informação. Deve ter em consideração que as funções realizadas devem conter um descrição e recomenda-se o uso de células "Markdown" para o efeito. Os resultados obtidos devem estar claramente apresentados recomendando-se o uso de gráficos ou tabelas para o efeito.

1. Elabore uma função (`gen_huff_table`) que gere uma tabela com o código binário para cada símbolo de um dado conjunto, usando o método de Huffman. Esta função deve ter como parâmetros de entrada um conjunto de símbolos e as suas probabilidades (ou em alternativa pode usar o número de ocorrências de cada símbolo, dado pelo seu histograma). Também pode em alternativa gerar não uma tabela mas outra estrutura de dados com os códigos pretendidos (ex: dicionário).

2. Elabore uma função (`encode_huff`) que dada uma mensagem (sequência de símbolos) e a tabela da ponto anterior, retorne uma sequência de bits com a mensagem codificada.

3. Elabore uma função (`decode_huff`) que dada uma sequência de bits (mensagem codificada) e a tabela do ponto 1, retorne uma sequência de símbolos (mensagem descodificada). Garanta que a mensagem retornada por esta função é igual à mensagem que é dada como parâmetro de entrada da função `encode_huff`.

4. Elabore uma função (`write2file`) que dada uma sequência de bits (mensagem codificada) e o nome do ficheiro, escreva a sequência de bits para o ficheiro.

5. Elabore uma função (`read2array`) que dado o nome do ficheiro, leia uma sequência de bits (mensagem codificada) contida no ficheiro.

6. Teste as funções elaboradas usando para o efeito os seguintes ficheiros com diferentes tipos de média. Adicionalmente, pode usar mais outros ficheiros à sua escolha que achar pertinentes.

Imagem: Use as imagens `LenaColor.tif` e `LenaGray.tif`.

Texto: Use os ficheiros `DecUniversalDH.pdf` e `DecUniversalDH.txt`.

Áudio: Use os ficheiros `HenryMancini-PinkPanther30s.mp3` e `HenryMancini-PinkPanther.mid`.

a) Gere o código usando a função realizada no ponto 1. Meça o tempo que demora a função.

b) Meça a entropia e o número médio de bits por símbolo. Calcule a eficiência.

c) Faça a codificação da mensagem contida no ficheiro (usando a função realizada no ponto 2). Meça o tempo que a função demora a fazer a codificação.

d) Grave um ficheiro com a mensagem codificada, usando a função realizada no ponto 4. Veja o tamanho do ficheiro.

e) Leia do ficheiro o conjunto de bits, usando a função realizada no ponto 5.

f) Faça a descodificação da mensagem (usando a função realizada no ponto 3.) Meça o tempo que a função demora a fazer a descodificação.

g) Compare a mensagem descodificada com a original e verifique que são iguais (erro nulo).

Exemplo para o ficheiro com imagem

```
from time import time
from os import path
import cv2
import matplotlib.pyplot as plt
```

```
# Lê a imagem em níveis de cinzento
x = cv2.imread("LenaGray.tif",cv2.IMREAD_GRAYSCALE)

# Converte a imagem (matriz) numa sequência de números (array)
xi = x.ravel()

# Calcula o histogram
h, bins, patches = plt.hist(xi,256,[0,256])

# Gera o código de Huffman
t0 = time()
tabela_codigo = gen_huff_table(np.arange(0,256),h)
t1 = time()
print "time:", t1-t0

# Codifica e grava ficheiro
seq_bit0 = encode_huff(xi,tabela_codigo)
write2file(seq_bit0, filename)
t2 = time()
print "time:", t2-t1

# Lê ficheiro e descodifica
seq_bit1 = read2array(filename)
yi = descode_huff(seq_bit1,tabela_codigo)
t3 = time()
print "time:", t3-t2

size_ini = path.getsize("filename original image")
size_end = path.getsize("filename compressed")
print "taxa: ", 1.* size_ini / size_end

plt.show()
cv2.waitKey(0)
plt.close("all")
cv2.destroyAllWindows()
```