# Trabalho 4 – Compressão com Huffman

Maria Isabel Nicolau 2310699

João Vitor Mallet 2310604

## Introdução

O trabalho consiste em implementar um compactador e um descompactador de arquivos texto baseados no algoritmo de Huffman. Este algoritmo utiliza várias das estruturas de dados estudadas no curso, tais como array, lista encadeada/lista de prioridades e árvore binária, vistos em aula.

## Descrição das Funções

### Leitura do arquivo e contagem de frequências

A função contar_frequencias lê o conteúdo do arquivo de entrada (irene.txt) e armazena a quantidade de vezes que cada caractere aparece. Utiliza-se um vetor de 256 posições (ASCII) para isso.

### Construção da árvore de Huffman

Com os dados de frequência, a função construir_arvore monta a árvore binária de Huffman. Para isso, criei uma lista de prioridade simples que insere os nós em ordem crescente de frequência, e vou combinando os dois menores até sobrar apenas um nó (a raiz da árvore).

### Geração dos códigos

A função gerar_codigos percorre a árvore recursivamente e atribui um código binário (como string) para cada caractere, salvando isso em uma tabela. Usei '0' para a esquerda e '1' para a direita.

### Compactação

Na função compactar_arquivo, percorro o arquivo original novamente e substituo cada caractere pelo seu código binário, bit a bit, agrupando em bytes para gravar no novo arquivo binário (saida_compactada.bin). Salvei também no começo do arquivo o número total de bits úteis.

### Descompactação

A função descompactar_arquivo lê o arquivo binário e, bit a bit, percorre a árvore de Huffman até chegar em uma folha, que é um caractere do texto original. Isso é gravado no arquivo de saída (saida_descompactada.txt), que deve ser igual ao original.

## Observações e Conclusões

- O arquivo txt esta sem acentos maiúsculas e sem simbolos especiais como orientado eplo professor na aula do dia 16/06

- A saída descompactada ficou exatamente igual ao texto original, mostrando que a compressão é sem perdas.

- Uma  parte difícil foi lidar com a gravação bit a bit no arquivo compactado, garantindo que não sobrassem bits "lixo" no final — para isso, precisei salvar o número total de bits no início do arquivo.

## Exemplo de Entrada e Saída

**Entrada (irene.txt):**

irene preta

irene boa

irene sempre de bom humor

imagino irene entrando no ceu

licenca meu branco

e sao pedro bonachao

entra irene voce nao precisa pedir licenca

**Saída (saida_descompactada.txt):**
*(igual ao original, validando a compressão sem perdas)*

**Saida da main:**

Tabela de frequências:

(10): 7

(32): 21

a (97): 13

b (98): 4

c (99): 9

d (100): 4

e (101): 25

g (103): 1

h (104): 2

i (105): 11

l (108): 2

m (109): 5

n (110): 15

o (111): 13

p (112): 5

r (114): 14

s (115): 3

t (116): 3

u (117): 3

v (118): 1

Tabela de códigos Huffman:

: 11111

 : 100

a: 1011

b: 01000

c: 0101

d: 01001

e: 110

g: 0111100

h: 011000

i: 1010

l: 011001

m: 01101

n: 001

o: 1110

p: 01110

r: 000

s: 011111

t: 111100

u: 111101

v: 0111101

Compactação concluída

Descompactação concluída