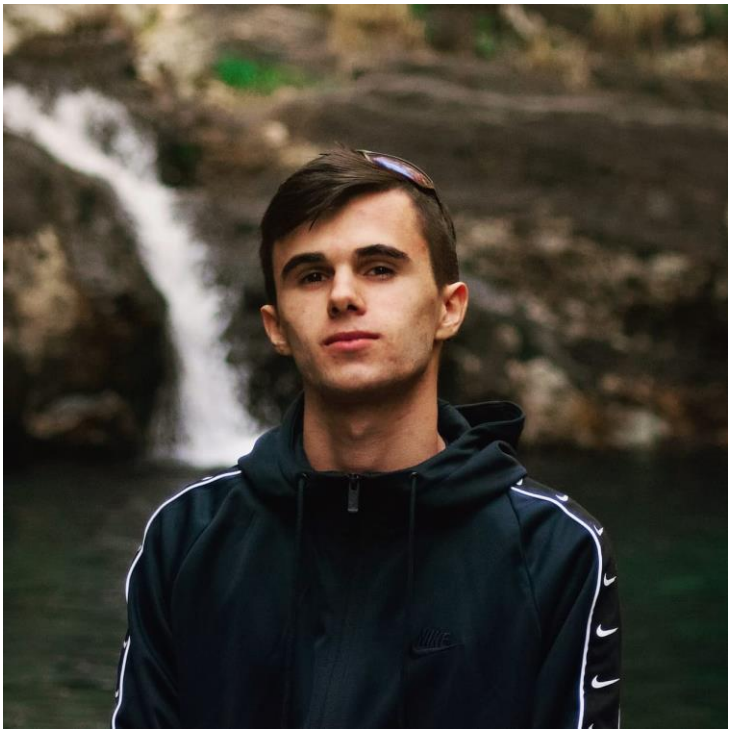
# Comunicação de Dados

## Trabalho Prático Alternativo

Universidade do Minho



**Nelson Miranda Ribeiro**

**A93188**

**MIEI**

# Índice

# 1. Estratégias Escolhidas

Neste trabalho, a estratégia que escolhi para trabalhar como estrutura de dados foi um **dicionário,** onde existe uma **key,** que representa um código, e que está associada a um dado **content**, representando o seu padrão. Optei também por adicionar ao dicionário uma variável de controlo que é a **max_key** que basicamente tem o código do último padrão existente no dicionário, podendo assim otimizar o código em muitas situações.

Outro tipo de estratégia usada foi a criação de um código modelar, ou seja, foram criadas muitas funções auxiliares à volta do dicionário e não só, onde eram depois mais tarde usadas em funções mais complexas, permitindo assim uma boa otimização do código em geral.

Optei, no uso das strings, de tentar fazer realloc's, começando sempre com o pior caso possível que pude imaginar para que o programa pudesse funcionar em todas as situações e depois, através de informação recebida, realocava essa memória.

Em suma, para a compressão **LZWd** e no processamento por blocos, segui o enunciado proposto.

# 2. Análise do Código

Relativamente à análise do código, as funções que gostava mais de realçar são: "**find_patern**" e "**process_string**".

Na "find_patern", as strings first_pattern e second_pattern são inicializadas com o mesmo símbolo que se está a analisar. Se se verificar que o padrão já se encontra no dicionário, ao second_pattern é lhe adicionado o próximo símbolo da sequência para depois se verificar se irá se tornar numa sub-string ou num padrão já existente no dicionário. O first_pattern só é alterado quando o second_pattern é um padrão do dicionário.

A "process_string", é a função principal da compressão. Ela processa a sequência de símbolos, de tal forma que a cada iteração do ciclo while identifica dois padrões consecutivos, adiciona ao dicionário um novo padrão resultante da concatenação dos dois padrões referidos e por fim adiciona a um buffer de outputs o código correspondente ao primeiro padrão. A "process_string" vai iterar até chegar ao fim da sequência de símbolos.

# 3. Resultados Obtidos

Dado o ficheiro original para exemplo com a seguinte sequência de caracteres: “ABABACBABABAABBABBAB” obtive os seguintes resultados:

```
Nelson Ribeiro, a93188, MIEI/CD, 14-fev-2021
Numero de blocos: 1
Tempo de execucao do modulo: 4.00 ms
Ficheiro gerado: aaa.txt.lzwd
```

Com debug ativado o resultado foi o seguinte:

```
aaa.txt.lzwd
1    (65)(66)(257)(65)(67)(258)(259)(257)(258)(258)
```

# 4. Conclusão

Em suma, este trabalho originou várias barreiras, vários bugs em que muitos deles foi possível encontrar uma resposta e seguir em frente, porem, sinto que este trabalho ainda se encontra com alguns bugs que não consegui resolver e gostava de ter tido mais tempo para os resolver. Sinto também, que pela falta de tempo na realização de mais testes quando a ferramenta debug não se encontra ativada, ou seja, quando o output tem de ser uma sequência binária, pode não se encontrar completamente funcional.

Foi sem dúvida um trabalho desafiante, no futuro tenho intenções de voltar a pegar neste trabalho, rever o código e tentar otimizar certos momentos para de seguida tentar fazer a descompressão LZWd.