# **Projeto 3**

# **PZYP: Compressão LZSS**

Conteúdo

[**Projeto 3** 1](#_Toc96637677)

[**PZYP: Compressão LZSS** 1](#_Toc96637678)

[Conteúdo 2](#_Toc96637679)

[1. Introdução e Objetivos 3](#_Toc96637680)

[2. Desenhos e Estrutura 5](#_Toc96637681)

[3. Implementação 7](#_Toc96637682)

# Introdução e Objetivos

No seguimento da formação em Técnico de Programação, como trabalho final da UFCD 10794- Programação Avançada em Python, foi-nos proposto realizar deste projeto cujo objetivo passa por aplicar os conhecimentos adquiridos no desenvolvimento de aplicações úteis para a manutenção e administração de sistemas.

Com este projeto, pretendeu-se desenvolver uma aplicação para **compressão** e **descompressão de ficheiros**,utilizando um código modular, organizado em funções, com tipos de dados apropriados, recorrendo a algumas bibliotecas e tendo como base o algoritmo LZSS, o sucessor do mais conhecido algoritmo de compressão, o LZ77.

Para que seja possível entender qual o modo de funcionamento da aplicação, procurou-se saber em que consistia o algoritmo LZSS e um pouco a sua história.

O algoritmo LZ77 realiza a compressão através da substituição de uma sequência repetida de bytes por uma referência, que indica a posição original da repetição (**distância** ou **offset**) e a sua dimensão (**comprimento** ou **length**). Ou seja, este algoritmo possui uma janela (buffer) que vai analisando e guardando os bytes processados e, sempre que é lida uma cópia de uma sequência de bytes já existente na janela, o LZ77 substitui essa sequência por uma referência <distância, comprimento>.

A janela possui um tamanho fixo, de modo que, quando os dados a analisar são superiores ao tamanho máximo da janela, ela precisa “deslizar” sobre os dados. Assim, quando o tamanho máximo da janela é atingido, sempre que uma sequência de bytes é adicionada à janela, é removida a mesma quantidade de bytes do lado dos bytes processados mais antigos, criando assim o efeito deslizante. Por este motivo a janela é muitas vezes designada de **janela deslizante**. No processo de descompressão, a janela é também recriada para deslizar pelo ficheiro e utilizada para interpretar as referências que for encontrando.

No entanto, o tamanho da janela pode variar. Em teoria, quanto maior a janela maior a taxa de conversão, ou seja, quanto maior a janela mais bytes podem ser armazenados e, por conseguinte, mais referências podem ser geradas. Porém, isto também torna o processo mais lento uma vez que terá que efetuar pesquisas mais demoradas. Outra consequência de utilizar uma janela maior é a necessidade de ter disponível um maior número de bits para representar quer a distância e quer o comprimento, sendo que tipicamente utilizam-se menos bits para o comprimento. Pode acontecer, no entanto, encontrarmos uma referência com um comprimento maior que a distância. Isto significa que, a sequência repetida depende de bytes que ainda não foram colocados na janela e representa uma **repetição consecutiva** de uma mesma sequência.

Com a compressão, o que se pretende é que o ficheiro comprimido seja mais pequeno que o original e para tal, a referência deve ser menor que a sequência que se pretende substituir. Com o LZ77 isto nem sempre acontece porque este algoritmo substitui cópias de qualquer dimensão por referências. Ou seja, o compressor cria uma referência <distância, comprimento> para uma sequência repetida, mas também cria uma referência com distancia e comprimento 0 para os bytes não comprimidos (**literais**), o que por vezes torna o ficheiro comprimido maior que o original.

Para minimizar estes problemas do LZ77 e torna-lo mais eficiente, James Storer e Thomas Szymanski desenvolveram o algoritmo LZSS, que continua a utilizar o conceito do LZ77 de criação de referências <distância, comprimento> para sequências repetidas, permitindo o descompressor identificar a localização da sequência original, mas, não cria referências de repetições com qualquer tamanho, definindo um **break even point**, e prefixa com uma **flag de compressão** os bytes comprimidos (ou referências) com um bit a 1 e os bytes literais com um bit a 0.

Ao definir um break even point, isto é, um limite mínimo de comprimento de uma repetição passível de ser comprimida, o LZSS só gera referências para repetições cujo comprimento seja superior ao valor definido, isto é, se o break even point for 2 o LZSS procura apenas repetições de sequências com pelo menos 3 bytes, nunca sendo então geradas referências para sequências com 0, 1 e 2 bytes. Como o tamanho máximo de uma sequência é de 15 bytes, o compressor pode utilizar estes valores para representar comprimentos até 18 bytes, se subtrair 3 ao comprimento da repetição. Já o descompressor necessita somar 3 ao comprimento de cada referência antes de recriar a sequência original.

Com a flag de compressão, o descompressor processa o ficheiro de entrada dependendo do valor do primeiro bit, ou seja, se o primeiro bit estiver a 0, ele lê o byte seguinte e coloca-o diretamente no ficheiro de saída e no fim da janela. Por outro lado, se o primeiro bit estiver a 1, ele lê os 2 bytes seguintes que devem indicar uma referência para uma sequência já lida, utilizando-a para localizar a sequência original na janela, e, a partir desta, recriar a cópia que é depois colocada na janela e no ficheiro de saída.

Com isto, o LZSS consegue resumir sequências maiores e aumentar, ainda que ligeiramente, a taxa de compressão.

Feita esta análise e para uma melhor orientação na criação desta aplicação, elaborou-se um fluxograma com o respetivo algoritmo de compressão e outro com o algoritmo de descompressão.

A elaboração deste projeto terminou com a criação da interface gráfica da aplicação, a partir da framework PyQT. Aqui, é possível anexar um documento proveniente do explorador de ficheiros, escolher a opção de comprimir ou descomprimir documento, selecionar o nível de compressão e por fim, indicar se pretende obter o ficheiro final encriptado e apontar o endereço para o guardar.

# Desenhos e Estrutura

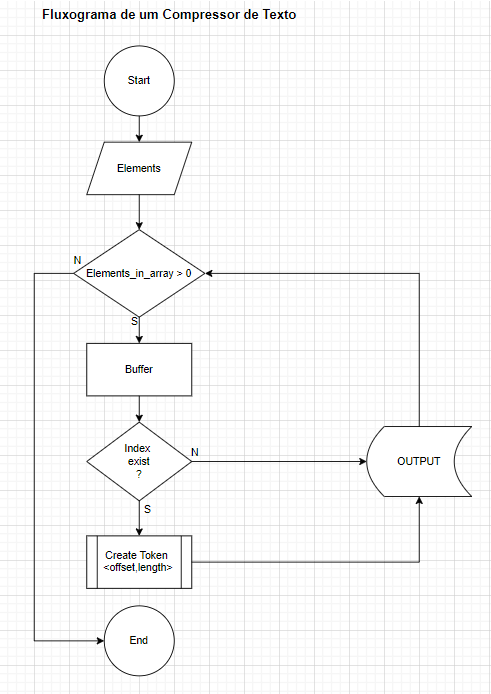


Figura 1- Compressor PZYP.

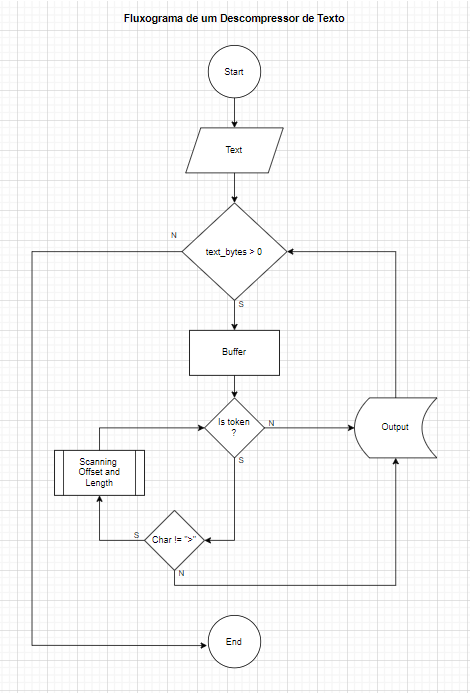


Figura 2- Descompressor PZYP

# Implementação

<https://go-compression.github.io/algorithms/lzss/>

https://en.wikipedia.org/wiki/Deflate