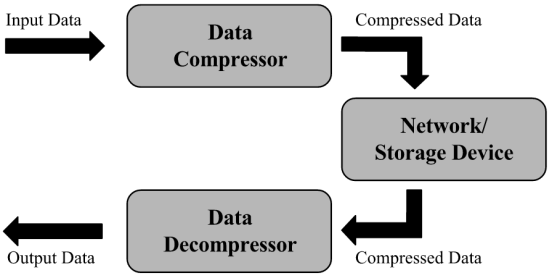
Diferentes codecs para compressão de dados

Nos dias de hoje, com o crescimento em tamanho das mais variadas fontes de informação, torna-se indispensável recorrer a técnicas de compressão, de modo a diminuir o tamanho dos dados, facilitando o seu armazenamento e transmissão em canais com largura de banda limitada.

Figura 1- Componentes do Sistema de Compressão de Dados



Fonte- Gupta, A. Bansal, V. Khanduja, Modern Lossless Compression Techniques: Review, Comparison and Analysis, Second International Conference on Electrical, Computer and Communication Technologies (ICECCT 2017)

Há dois tipos de técnicas de compressão: *lossless* e *lossy*. Tal como o nome indica, as técnicas *lossless* não levam a qualquer perda de informação, sendo a compressão feita de modo a representar a fonte por um número menor de bits, podendo-se posteriormente reconverter o ficheiro resultante da compressão na fonte original.

Nas conversões do tipo *lossy* ocorre perda de informação, sendo removidos bits desnecessários, resultando num ficheiro mais compacto. Este ficheiro nunca poderá ser convertido no ficheiro original, devido há perda de parte da informação na compressão. São exemplos de técnicas deste tipo o JPEG e o MP3.

Neste trabalho iremos apenas explorar técnicas de compressão do tipo *lossless*.

Alguns algoritmos de compressão *lossless*

Run Length Encoding:

O algoritmo procura símbolos repetidos. De seguida, substitui os símbolos seguidos repetidos por um único carater e o número de vezes que esse símbolo ocorre.

aaaabbbbbbbaaccccddaabb

(A, 4)

Código de Huffman:

Os símbolos são codificados de acordo com a sua ocorrência (quanto mais frequente os símbolos, menor o número de bits usados para os codificar). Para definir os códigos para cada símbolo, recorre-se a uma árvore binária.

Figura 2- Árvore de Huffman

Diagram

Description automatically generated

Fonte- <https://www.google.com/url?sa=i&url=http%3A%2F%2Fhomes.sice.indiana.edu%2Fyye%2Flab%2Fteaching%2Fspring2014-C343%2Fhuffman.php&psig=AOvVaw0u4ZXFRyd2_4rll3H_FbDJ&ust=1605305905348000&source=images&cd=vfe&ved=0CAIQjRxqFwoTCJiM7NWE_uwCFQAAAAAdAAAAABAi>

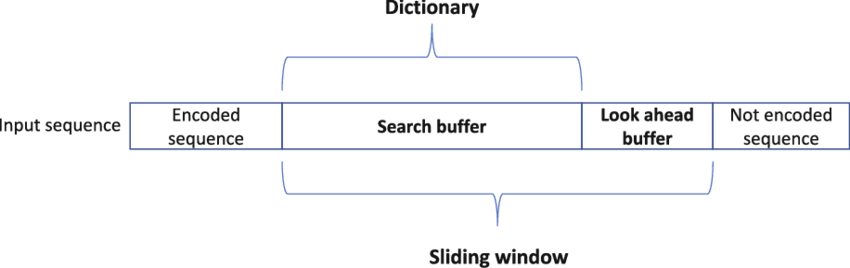
LZ 77:  
 A fonte é percorrida desde o início, lendo um carater de cada vez.

Existem dois buffers :

*Search buffer* – buffer com os últimos *Ns* símbolos codificados

*Look-ahead buffer* – buffer com os próximos *NL* símbolos a serem codificados

Figura 3- Sliding Window no LZ77



Fonte- https://www.researchgate.net/figure/Overview-of-LZ77-lossless-compression-algorithm\_fig2\_320497774

Procura-se no *search buffer* o maior padrão que ocorre no *look-ahead buffer* (início no primeiro símbolo). De seguida, codifica-se a posição relativa (permite codificar sequências de comprimento variável):

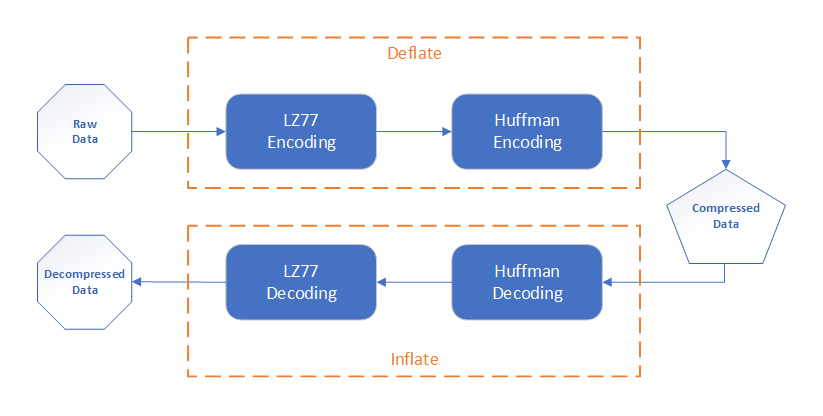
{Offset, Length, código próximo símbolo}

Próximo símbolo é enviado para o caso de não existir padrão ({*0,0,símbolo*}) – Comprimento do padrão {0,...,*NS+NL*}

Deflate:

Este algoritmo divide a fonte em vários blocos. Cada bloco é comprimido através duma combinação do LZ77 e de Códigos de Huffman.

Figura 4- Passos do algoritmo Deflate

Fonte- <https://www.euccas.me/zlib/>

Predição por Partial Matching:

Encontram-se ocorrências múltiplas de strings de tamanho N (onde N é o valor de uma constante definida).

Ao que sobrar, tentar encontrar outras ocorrências de strings de ordem N-1.

Repetir até que todas as combinações possíveis tenham sido encontradas e lhe tenham sido atribuídas a respectiva probabilidade a cada string de acordo com a sua ocorrência.

Enviar as probabilidades obtidas para o codificador aritmético adaptativo.

Comparação de diferentes algoritmos de compressão

Para escolhermos o melhor algoritmo de compressão, devemos ter em conta três fatores:

## - tipo de imagem

## - taxa de compressão

## - velocidade de compressão

Figura 5- Taxa de compressão de diferentes tipos de ficheiros com diferentes algoritmos

Chart, bar chart

Description automatically generated

Fonte- Gupta, A. Bansal, V. Khanduja, Modern Lossless Compression Techniques: Review, Comparison and Analysis, Second International Conference on Electrical, Computer and Communication Technologies (ICECCT 2017)

Na figura de cima, podemos observar a taxa de compressão de diferentes algoritmos para diversos tipos de ficheiro. Neste trabalho, pretendemos comprimir ficheiros de imagem, que corresponde ao ficheiro “mr” na figura. Assim, podemos verificar que os algoritmos do tipo PPM apresentam as melhores taxas de compressão, tendo o Deflate a pior.

Figura 6- Velocidade de compressão de diferentes algoritmos

Chart, bar chart

Description automatically generated

Fonte- Gupta, A. Bansal, V. Khanduja, Modern Lossless Compression Techniques: Review, Comparison and Analysis, Second International Conference on Electrical, Computer and Communication Technologies (ICECCT 2017)

**T2 TI**

The Rapid growth in the field of multimedia and in digital imaging increased the need to transmit digital images .Transmitting the images without compressing them takes more disk space as well as much time for transmission over the network. So the basic idea is to remove the redundancy of data presented within the image so that we can reduce the size of image without affecting the essential data in it( without compromising its quality).

The motive of image compression is to reduce the amount of data required for representing sampled digital images and therefore reduce the cost for storage and transmission.

The best algorithm is measured depends on the following 3 factors: quality of the image, amount of compression, speed of compression

Data compression techniques are used to reduce the size of the data which enables its storage and transmission over a limited bandwidth channel.

The input data is processed by a Data Compressor, which usually iterates over the data twice. In the first iteration, the compression algorithm tries to gain knowledge about the data which in turn is used for efficient compression in the second iteration. The compressed data along with the additional data used for efficient compression is then stored or transmitted through a network to the receiver. The receiver then decompresses the data using a decompression algorithm. Though data compression can reduce the bandwidth usage of a network and the storage space, it requires additional computational resources for compression and decompression of data, and this additional processing may be harmful to some applications such as embedded devices which have very limited processing capability[2]. Hence, a compression algorithm needs to find a balance between the compression compression/decompression time.

Compression techniques can be divided into two types, (a)lossless and (b)lossy [2]. As the name suggests, lossless compression techniques do not lead to any loss of data or information. The compression is done by representing the file using less number of bits, without any loss of information.

lossy compression techniques remove the unnecessary bits, reducing the size of the file. The compressed file cannot be converted back to the original file as some part of the information is lost in the compression. JPEG [5] and MP3 [6] are examples of lossy compression techniques.