**ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA HỌC MÁY TÍNH**

**--o0o--**

**BÁO CÁO**

**DATA COMPRESSION WITH HUFFMAN, LZW Alg.**

Môn: **Tính toán đa phương tiện – CS232**

Lớp: **CS232.M21.KHCL**

GVHD: **TS. Đỗ Văn Tiến**

Sinh viên thực hiện :

**Hồ Thịnh** - **19522274**

**Lê Vinh Quang - 19522093**

**Nguyễn Thành Trung** - **19522432**

***Thành phố Hồ Chí Minh, ngày 2 tháng 7 năm 2022***

**Lời cảm ơn**

Đầu tiên, chúng em xin gửi lời cảm ơn và lòng biết ơn sâu sắc nhất tới TS. Đỗ Văn Tiến – giảng viên phụ trách bộ môn *“Tính toán đa phương tiện”,* trực thuộc Khoa Khoa học Máy Tính, đã trang bị cho em những kýến thức trong môn học này.

Tuy nhiên trong quá trình nghiên cứu đề tài, do kiến thức chuyên ngành còn hạn chế nên chúng em vẫn còn nhiều thiếu sót khi tìm hiểu, đánh giá, trình bày về đề tài. Những góp ý của thầy trong buổi báo cáo là những kinh nghiệm quý báu mà chúng em học hỏi, khắc phục cho quá trình học tập và làm việc sau này.

Xin chân thành cảm ơn Thầy.

Nội dung

[**Chương 1 Tổng quan** 4](#_Toc107690681)

[**1** **Mục tiêu đồ án** 4](#_Toc107690682)

[**2** **Đặt vấn đề** 4](#_Toc107690683)

[**2.1** **Ngữ cảnh bài toán** 4](#_Toc107690684)

[**2.2** **Input/ Output bài toán** 4](#_Toc107690685)

[**3** **Thách thức bài toán** 5](#_Toc107690686)

[**Chương 2 Các nghiên cứu liên quan** 6](#_Toc107690687)

[**1 WinRAR**[1] 6](#_Toc107690688)

[**2 Unarchiver**[2] 6](#_Toc107690689)

[**Chương 3 Nội dung chính** 8](#_Toc107690690)

[**1. Nén dữ liệu** 8](#_Toc107690691)

[**1.1 Định nghĩa** 8](#_Toc107690692)

[**1.2 Phương thức hoạt động của nén dữ liệu** 8](#_Toc107690693)

[**1.3 Nén có mất mát** 8](#_Toc107690694)

[**1.4 Nén không mất mát** 8](#_Toc107690695)

[**2 Mã hoá dữ liệu trong máy tính** 9](#_Toc107690696)

[**3 Huffman Coding (Static Huffman)** 10](#_Toc107690697)

[**4** **Lempel-Ziv Welch (LZW)** 11](#_Toc107690698)

[**4.1 Giới thiệu** 11](#_Toc107690699)

[**4.2** **Cách hoạt động** 12](#_Toc107690700)

[**4.3 Thuật toán nén** 13](#_Toc107690701)

[**4.4 Thuật toán giải nén**[5] 15](#_Toc107690702)

[**Chương 4 Thực nghiệm và đánh giá thuật toán** 17](#_Toc107690703)

[**1** **Thực nghiệm** 17](#_Toc107690704)

[**2** **Đánh giá thuật toán** 17](#_Toc107690705)

[**2.1** **Huffman Coding** 17](#_Toc107690706)

[**2.2** **Lempel-Ziv Welch** 18](#_Toc107690707)

[**Chương 5 Tổng kết** 19](#_Toc107690708)

[**Tài liệu tham khảo** 20](#_Toc107690709)

# **Chương 1 Tổng quan**

## **1** **Mục tiêu đồ án**

Để thực hiện đồ án, nhóm đã đặt ra những mục tiêu như sau:

* Hiểu được ý tưởng của nén dữ liệu.
* Hiểu được ý tưởng và cách thức hoạt động của hai thuật toán Huffman coding và LZW.
* Cách đánh giá thuật toán nén.
* Ưu/nhược điểm của từng thuật toán.

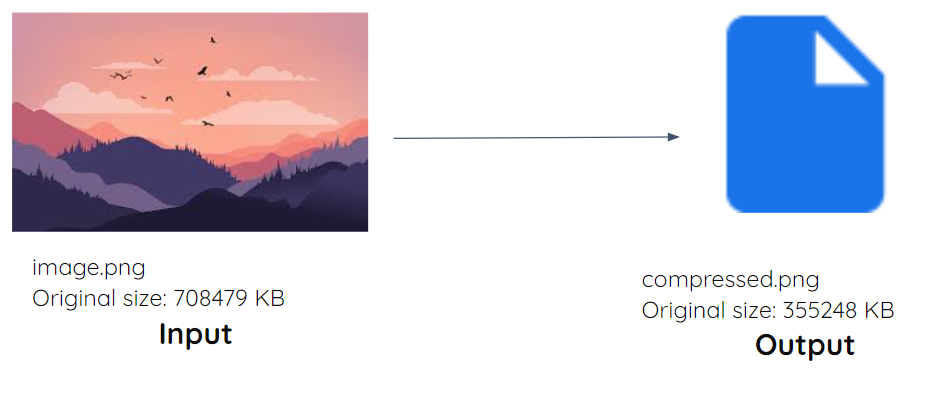
## **2** **Đặt vấn đề**

### **2.1** **Ngữ cảnh bài toán**

Trong thế giới 4.0, dữ liệu đang là nguồn tài nguyên hữu ích, nó chứa được rất nhiều thông tin quan trọng. Khi mà xã hội càng phát triển, các thông tin cần lưu trữ càng nhiều, càng phức tạp thì việc tận dụng, sử dụng các bộ nhớ máy một cách tối ưu trở nên là một vấn đề cấp bách. Các ổ cứng hiện nay thì có dung lượng nhất định. Muốn sử dụng ổ cứng có dung lượng lớn thì bạn phải trả ra rất nhiều tiền và ngược lại, bạn sẽ trả ít tiền hơn nên ổ cứng bạn có dung lượng nhỏ, tuy nhiên với dung lượng nhỏ thì việc lưu trữ dữ liệu trở nên rất khó khăn. Việc truyền tải dữ liệu hiện nay với một tệp dữ liệu có dung lượng lớn cũng có rất nhiều khó khăn về mặt thời gian và dễ mất mát dữ liệu. Vì vậy ta cần một phương pháp xử lý dữ liệu mà làm cho nó có dung lượng nhỏ đi nhưng vẫn giữ được thông tin có bên trong nó. Đó là nén dữ liệu (data compression).

### **2.2** **Input/ Output bài toán**

* Input: Một tệp ảnh hoặc văn bản có dung lượng n <đơn vị đo dung lượng>.
* Output : Một tệp ảnh hoặc văn bản đã được nén tương ứng với input, có dung lượng nhỏ hơn n<đơn vị đo dung lượng>.



*Hình 1. Input/ Output của quá trình nén ảnh.*

## **Thách thức bài toán**

Dữ liệu ngày nay có thể có cấu trúc, không cấu trúc và bán cấu trúc, với nhiều cấu trúc rất phức tạp làm cho việc nén dữ liệu đôi khi không mang lại hiệu quả. Các thuật toán dùng trong nén dữ liệu có những ưu và nhược điểm riêng, nó không mang lại tính hiệu quả toàn diện đối với mọi kiểu dữ liệu.

Cụ thể hơn đối với chủ đề nghiên cứu lần này, nhóm phải đối mặt với một số trường hợp ngoài mong muốn đó là những ảnh có màu phức tạp khiến cho cài đặt hiện tại của nhóm không thu được kết quả mã hoá hoặc tệ hơn là bị lỗi code một phần là source code cài đặt chưa thực sự tối ưu trong việc thiết kế cũng như sử dụng các cấu trúc dữ liệu. Đây là thách thức lớn nhất đối với nhóm nghiên cứu bởi vì điều này xuất hiện ở cả 2 cài đặt thuật toán của nhóm chính vì thế mà nhóm sẽ vẫn còn nghiên cứu và cài đặt trong tương lai.

Một vấn đề thách thức khác, trong nghiên cứu lần này của nhóm đó chính là nhóm muốn đưa ra một loại file nén để lưu trữ cho kết quả mã hoá ở từng loại thuật toán nghiên cứu. Tuy nhiên, đây cũng là một việc hết sức khó khăn vì nếu lưu dưới dạng một file lưu trữ text thông thường thì thường sẽ làm cho kích thước có khi còn to hơn gấp đôi so với dữ liệu đầu vào ban đầu (bởi vì bitstring cho dữ liệu đầu vào là những ảnh phức tạp không hề nhỏ nên kéo theo nội dung lưu trữ cũng dài).

# **Chương 2 Các nghiên cứu liên quan**

## **1 WinRAR**[[1]](https://www.win-rar.com/fileadmin/downloads/WinRAR_Product_Information.pdf)

WinRAR ra đời vào ngày 22 tháng 4 năm 1995, được phát triển bởi Eugene Roshal và phân phối bởi Alexander Roshal. WinRAR là một công cụ nén dữ liệu, đây là chương trình chỉ dành cho Window. Nó có thể tạo và xem các tệp lưu trữ ở định dạng tệp RAR hoặc ZIP, và giải nén nhiều định dạng tệp lưu trữ. Với công cụ này, chúng ta có thể sẽ tiết kiệm không gian có giá trị trên ổ cứng máy tính, ổ USB hoặc thiết bị di động và quản lý các tệp của chúng ta hiệu quả hơn. WinRAR sử dụng thuật toán Lempel–Ziv–Storer–Szymanski (LZSS) làm phương pháp nén dữ liệu chính.

WinRAR có thể thực hiện :

* Nén và giải nén tệp
* Nén tệp email đính kèm
* Sử dụng mã bảo vệ cho tệp và tệp đính kèm
* Khóa tệp
* Chia tệp
* Tạo các tệp tự trích xuất
* Làm tệp dự phòng
* Bảo vệ tệp

Các định dạng tệp được hỗ trợ bao gồm: zip, 7z, rar, gz, bz2, xz, tar, tar.gz, tar.bz2, tar.xz, tar.Z, iso-9660..

Những lợi ích mà WinRAR có thể mang lại:

* Miễn phí với mọi phiên bản ngôn ngữ.
* Miễn phí cho mọi thiết bị thuộc quyền sở hữu của một cá nhân có giấy phép sử dụng nó.

## **2 Unarchiver**[[2]](https://en.wikipedia.org/wiki/The_Unarchiver)

Unarchiver ban đầu được phát triển bởi Dag Ågren, sau đó nó tiếp tục được phát triển bởi Circlesoft, MacPaw. Hiện tại, Unarchiver đã có sẵn trong 18 ngôn ngữ. Unarchiver là công cụ dùng để đọc và giải nén tệp, nó được sử dụng trên Mac OS, Ubuntu, Linux và cả Window. Các định dạng tệp được hỗ trợ bao gồm Zip, Tar-GZip, Tar-BZip2, Rar, 7-zip, LhA, StuffIt và nhiều định dạng,.. Unarchiver không nén tệp. Unarchiver có sử dụng nhiều thuật toán giải nén dữ liệu như: Lempel–Ziv–Storer–Szymanski (LZSS), Lempel–Ziv–Welch (LZW), Burrows–Wheeler transform (BWT), … để xây dựng nên thư viện XAD-Master hỗ trợ trong việc đọc và giải nén dữ liệu.

Những lợi ích mà Unarchiver mang lại:

* Đây là phần mềm hoàn toàn miễn phí.
* Hỗ trợ người dùng mở và giải nén tệp với nhiều định dạng được sử dụng nhiều hiện nay : Rar, 7-zip,.. và cả các định dạng cũ :  StuffIt, DiskDoubler, ARJ, ARC, LZH, định dạng BIN, ISO và một số ít định dạng chỉ dành riêng cho Windows nữa.

# **Chương 3 Nội dung chính**

## **1. Nén dữ liệu**

### **1.1 Định nghĩa**

Nén dữ liệu là làm giảm số lượng bit cần thiết để biểu diễn dữ liệu. Nén dữ liệu có thể tiết kiệm dung lượng lưu trữ, tăng tốc độ truyền tệp và giảm chi phí cho phần cứng lưu trữ và [băng thông](https://www.techtarget.com/searchnetworking/definition/bandwidth) mạng.

### **1.2 Phương thức hoạt động của nén dữ liệu**

Nén được thực hiện bởi một chương trình sử dụng công thức hoặc [thuật toán](https://www.techtarget.com/whatis/definition/algorithm) để xác định cách thu nhỏ kích thước của dữ liệu.

 Ví dụ: Nén văn bản có thể đơn giản như sử dụng một thuật toán, thuật toán này sử dụng phép tính toán giữa 2 hoặc nhiều ký tự liên tiếp theo một quy luật nhất định, loại bỏ tất cả các [ký tự](https://www.techtarget.com/whatis/definition/character) có thuộc quy luật nhất định, chèn một ký tự đại diện duy nhất để chỉ ra một chuỗi ký tự thỏa quy luật của thuật toán và thay thế một chuỗi bit nhỏ hơn cho một chuỗi bit thường xuyên xảy ra. Nén dữ liệu có thể giảm tệp văn bản xuống theo tỷ lệ phần trăm so với kích thước ban đầu của nó.

### **1.3 Nén có mất mát**

Vào cuối những năm 1980, hình ảnh kỹ thuật số trở nên phổ biến hơn và các tiêu chuẩn nén hình ảnh [[3]](https://en.wikipedia.org/wiki/Image_compression)không mất dữ liệu xuất hiện. Vào đầu những năm 1990, phương pháp nén tổn hao bắt đầu được sử dụng rộng rãi. Trong các kế hoạch này, một số thông tin mất mát được chấp nhận vì việc giảm chi tiết không cần thiết có thể tiết kiệm dung lượng lưu trữ. Có một sự [đánh đổi](https://en.wikipedia.org/wiki/Trade-off) tương ứng giữa việc bảo toàn thông tin và giảm kích thước.

Hầu hết các dạng nén tổn hao đều dựa trên transform coding, đặc biệt là discrete cosine transform (DCT). Nó được đề xuất lần đầu tiên vào năm 1972 bởi [Nasir Ahmed](https://en.wikipedia.org/wiki/N._Ahmed) và được công bố vào tháng 1 năm 1974. DCT là phương pháp nén tổn hao được sử dụng rộng rãi nhất, và được sử dụng ở các định dạng đa phương tiện cho [hình ảnh](https://en.wikipedia.org/wiki/Image_compression) (chẳng hạn như [JPEG](https://en.wikipedia.org/wiki/JPEG) và [HEIF](https://en.wikipedia.org/wiki/HEIF) ), [[17]](https://en.wikipedia.org/wiki/Data_compression#cite_note-JPEG-17) [video](https://en.wikipedia.org/wiki/Video_compression) (chẳng hạn như [MPEG](https://en.wikipedia.org/wiki/MPEG) , [AVC](https://en.wikipedia.org/wiki/H.264/AVC) và [HEVC](https://en.wikipedia.org/wiki/HEVC)) và âm thanh (chẳng hạn như [MP3](https://en.wikipedia.org/wiki/MP3), [AAC](https://en.wikipedia.org/wiki/Advanced_Audio_Coding) và [Vorbis](https://en.wikipedia.org/wiki/Vorbis)).

### **1.4 Nén không mất mát**

Nén không mất dữ liệu cho phép [khôi phục](https://www.techtarget.com/searchdatabackup/definition/restore) tệp về trạng thái ban đầu mà không làm mất một bit dữ liệu nào khi tệp được giải nén.

Nén không mất dữ liệu là cách tiếp cận điển hình với tệp thực thi, cũng như tệp văn bản và bảng tính, trong đó việc mất từ ​​hoặc số sẽ thay đổi thông tin.

Phương pháp nén Lempel-Ziv (LZ) [[4]](https://en.wikipedia.org/wiki/LZ77_and_LZ78) là một trong những thuật toán phổ biến nhất để lưu trữ không mất dữ liệu. Ngoài ra còn có các phương pháp như Huffman coding, LZW,.. Huffman coding và LZW là hai thuật toán mà nhóm sử dụng trong đồ án nén dữ liệu của mình.

## **2 Mã hoá dữ liệu trong máy tính**

Để mã hóa các ký hiệu (bao gồm ký tự, chữ số, …) trong máy tính, ta thay chúng bằng các chuỗi nhị phân, được gọi là mã ký tự của ký hiệu đó. Ví dụ bộ mã ASCII ra đời năm 1967, mã hóa cho 256 ký hiệu là biểu diễn nhị phân của các số từ 0 đến 255, mỗi mã ký tự gồm 8 bit. Trong ASCII mã ký tự của ký tự “a” là 1100001, của ký tự “A” là 1000001. Trong cách mã hóa này các mã ký tự của tất cả 256 ký hiệu có độ dài bằng nhau (mỗi mã ký tự 8 bit). Nó được gọi là mã hóa với độ dài không đổi.

Khi mã hóa một tài liệu có thể không sử dụng đến tất cả 256 ký hiệu. Hơn nữa trong tài liệu chữ cái "a" chỉ có thể xuất hiện 1000000 lần còn chữ cái "A" có thể chỉ xuất hiện 2, 3 lần. Như vậy ta có thể không cần dùng đủ 8 bit để mã hóa cho một ký hiệu, hơn nữa độ dài (số bit) dành cho mỗi ký hiệu có thể khác nhau, ký hiệu nào xuất hiện nhiều lần thì nên dùng số bit ít, ký hiệu nào xuất hiện ít thì có thể mã hóa bằng mã ký tự dài hơn. Như vậy ta có việc mã hóa với độ dài thay đổi. Tuy nhiên, nếu mã hóa với độ dài thay đổi, khi giải mã ta làm thế nào phân biệt được chuỗi bit nào là mã hóa của ký hiệu nào. Một trong các giải pháp là dùng các dấu phẩy (",") hoặc một ký hiệu quy ước nào đó để tách mã ký tự của các ký tự đứng cạnh nhau. Nhưng như thế số các dấu phẩy sẽ chiếm một không gian đáng kể trong bảng mã. Một cách giải quyết khác dẫn đến khái niệm mã tiền tố. Mã tiền tố là bộ các mã ký tự của một tập hợp các ký hiệu sao cho mã ký tự của mỗi ký hiệu không là tiền tố (phần đầu) của mã ký tự một ký hiệu khác trong bộ mã ấy.

Ví dụ: Giả sử mã hóa từ "ARRAY", tập các ký hiệu cần mã hóa gồm 3 chữ cái "A","R","Y".

* Nếu mã hóa bằng các mã ký tự có độ dài bằng nhau ta dùng ít nhất 2 bit cho một chữ cái chẳng hạn "A"=00, "R"=01, "Y"=10. Khi đó mã hóa của cả từ là 0001010010. Để giải mã ta đọc hai bit một và đối chiếu với bảng mã.
* Nếu mã hóa "A"=0, "R"=01, "Y"=11 thì bộ mã ký tự này không là mã tiền tố vì mã ký tự của "A" là tiền tố của mã ký tự của "R". Để mã hóa cả từ ARRAY phải đặt dấu ngăn cách vào giữa các mã ký tự 0,01,01,0,11
* Nếu mã hóa "A"=0, "R"=10, "Y"=11 thì bộ mã này là mã tiền tố. Với bộ mã tiền tố này khi mã hóa chuỗi "ARRAY" ta có 01010011.

Để dễ dàng hơn trong việc biểu diễn mã tiền tố ta biểu diễn các ký tự lên cây nhị phân. Nếu có một cây nhị phân n lá ta có thể tạo một bộ mã tiền tố cho n ký hiệu bằng cách đặt mỗi ký hiệu vào một lá. Mã ký tự của mỗi ký hiệu được tạo ra khi đi từ gốc tới lá chứa ký hiệu đó, nếu đi qua cạnh trái thì ta thêm số 0, đi qua cạnh phải thì thêm số 1. Theo như cách biểu diễn này, biểu diễn cho ví dụ ở trên ta được mã ký tự của "A" là 0, của "R" là 10, của "Y" là 11.

Từ ví dụ trên thấy mã hóa của chuỗi "ARRAY" bằng mã độ dài cố định mất 10 bit, bằng mã tiền tố đã đưa ra mất 8 bit, tiết kiệm được 20%. Bài toán đặt ra là bộ mã tiền tố đã tối ưu chưa?

## **3 Huffman Coding (Static Huffman)**

Các tập tin của máy tính được lưu dưới dạng các ký tự có chiều dài không đổi là 8 bits. Trong nhiều tập tin, xác suất xuất hiện các ký tự này là nhiều hơn các ký tự khác, từ đó ta thấy ngay rằng nếu chỉ dùng một vài bit để biểu diễn cho các ký tự có xác suất xuất hiện lớn và dùng nhiều bit hơn để biểu diễn cho các ký tự có xác suất xuất hiện nhỏ thì có thể tiết kiệm được độ dài tập tin một cách đáng kể. Đây cũng chính là ý tưởng của thuật toán Huffman Coding được đề xuất bởi David A. Huffman vào năm 1952.

Để thực hiện nén dữ liệu với Huffman Coding thì cần phải tiến hành theo các bước như sau:

* Bước 1: Duyệt dữ liệu để lập bảng thống kê số lần xuất hiện của mỗi ký tự
* Bước 2: Tạo cây Huffman từ bảng thống kê
* Bước 3: Tạo bảng mã cho các ký tự
* Bước 4: Duyệt dữ liệu để thay thế các ký tự bằng mã bit tương ứng
* Bước 5: Lưu lại thông tin của cây Huffman dùng để giải nén

Trong các bước thực hiện trên ta có đề cập tới cây Huffman, thì đây là một cấu trúc lưu trữ quan trọng của thuật toán nơi lưu lại thông tin chủ yếu của dữ liệu nén cần cho việc giải nén sau này. Cây Huffman là binary tree với giá trị 0 thể hiện cho nhánh trái và 1 cho nhánh phải. Những giá trị trên nút của cây đại diện cho chuỗi ký tự (tổng hợp từ các nút con) và tổng tần số (tổng giá trị của các nút con). Nút lá của cây đại diện cho các ký tự có tần suất xuất hiện ít nhất. Từ việc cài đặt cây ta có thể thấy cây Huffman là cây nhị phân đầy đủ, các nút có tần suất cao nằm gần gốc, các nút có tần suất thấp nằm xa gốc, tổng số nút của cây là 2n − 1 với n là số ký tự.

Các bước tạo cây Huffman: Từ bảng thống kê tần suất xuất hiện của các ký tự ta khởi tạo cây T gồm có n nút là các ký tự với các trọng số của chúng.

* Bước 1: Chọn hai phần tử có trọng số thấp nhất x và y
* Bước 2: Tạo một phần tử z từ x và y sao cho trọng số của z là tổng của x và y và chuỗi của z là tổng của x và y
* Bước 3: Loại bỏ x và y khỏi bảng thống kê
* Bước 4: Thêm z vào bảng thống kê và thêm nút z vào cây T với x và y là nút con trái và phải
* Lặp lại các bước trên cho đến khi chỉ còn một phần tử

Vd minh hoạ tạo cây Huffman: [link](../OneDrive/Desktop/CreateHuffmanTreeExample.docx)

Từ ví dụ trên ta có thể nhận thấy được đoạn mã hoá theo thuật toán Huffman Coding tiết kiệm 1 lượng lớn bits để lưu trữ 1 chuỗi đơn giản. Tuy nhiên, thuật toán này cũng có một số mặt hạn chế, dễ thấy nhất trong ví dụ trên đó là việc nó phải duyệt qua toàn bộ chuỗi để thống kê được tần suất xuất hiện của ký tự, và việc phải tạo 1 cây nhị phân cho lưu trữ dữ liệu nén, 2 điều này có thể dẫn đến tốc độ xử lý thuật toán đối với những dữ liệu phức tạp hơn (như văn bản, ảnh màu,…) có thể không hiệu quả và dữ liệu nén lưu trữ cũng có thể lớn. Do đó mà một thuật toán cải tiến dựa trên thuật toán này được để xuất bởi [Newton Faller](https://en.wikipedia.org/wiki/Newton_Faller) (1973) và [Robert Gray Gallager](https://en.wikipedia.org/wiki/Robert_G._Gallager) (1978) sau đó được [Donald Ervin Knuth](https://en.wikipedia.org/wiki/Donald_Knuth) hoàn thiện (1985), đó là Adaptive Huffman Coding (FGK). Về ưu điểm, FGK khắc phục được các nhược điểm của Huffman Coding, hơn nữa thuật toán còn cho phép ta nén dữ liệu dựa trên dữ liệu phát sinh ở thời gian thực mà không cần đến thống kê như Huffman.

1. **Lempel-Ziv Welch (LZW)**

**4.1 Giới thiệu**

LZW là một phương pháp nén được phát minh bởi Lempel - Zip và Welch, là thuật toán nén dữ liệu không mất mát. Ý tưởng dựa vào các mẫu lặp lại để tiết kiệm dung lượng dữ liệu. LZW là kỹ thuật hàng đầu để nén dữ liệu có mục đích chung do tính đơn giản và linh hoạt của nó. Nó là cơ sở của nhiều tiện ích PC tuyên bố "tăng gấp đôi dung lượng ổ cứng của bạn. huật toán này lưu trữ bằng việc chuẩn bị một danh sách các đoạn bao gồm rât nhiều ký tự trong đầu vào là một bảng chữ cái nào đó, nó thực hiện một quá trình mở rộng các bảng chữ cái hay nói cách khác là nó dùng ký tự bổ sung để biểu diễn lại các chuỗi của ký tự chính quy. Để nén LZW trên mã ASCII 8 bits ta cần mở rộng bảng chữ cái bằng cách dùng 9 bits hay nhiều hơn 256 ký tự bổ sung mà mã 9 bits cung cấp được dùng để lưu trữ các chuỗi mã được quyết định từ các chuỗi trong nguồn tin.

**4.2** **Cách hoạt động**

Nó hoạt động dựa trên một ý tưởng rất đơn giản là người mã hoá và người giải mã cùng xây dựng bảng mã. Bảng mã này không cần được lưu kèm với dữ liệu trong quá trình nén, mà khi giải nén, người giải nén sẽ xây dựng lại nó.

Khi nén LZW hoạt động bằng cách đọc một chuỗi ký hiệu, nhóm các ký hiệu thành chuỗi và chuyển đổi chuỗi thành mã. Bởi vì các mã chiếm ít không gian hơn các chuỗi mà chúng thay thế, chúng tôi nhận được nén. Nó sẽ hoạt động trên nguyên tắc như sau:

* Một xâu ký tự là một tập hợp từ hai ký tự trở lên.
* Nhớ tất cả các xâu ký tự đã gặp và gán cho nó một dấu hiệu (token) riêng.
* Nếu lần sau gặp lại xâu ký tự đó, xâu ký tự sẽ được thay thế bằng dấu hiệu của nó.

Phần quan trọng nhất của phương pháp nén này là phải tạo một bảng mã rất lớn dùng để lưu giữ các xâu ký tự đã gặp (Mảng này được gọi là "Từ điển"). Khi các byte dữ liệu cần nén được đem đến, chúng liền được giữ lại trong một bộ đệm chứa (Accumulator) và đem so sánh với các chuỗi đã có trong "từ điển". Nếu chuỗi dữ liệu trong bộ đệm chứa không có trong "từ điển" thì nó được bổ sung thêm vào "từ điển" và chỉ số của chuỗi ở trong "từ điển" chính là dấu hiệu của chuỗi. Nếu chuỗi trong bộ đệm chứa đã có trong "từ điển" thì dấu hiệu của chuỗi được đem ra thay cho chuỗi ở dòng dữ liệu ra. Có bốn quy tắc để thực hiện việc nén dữ liệu theo thuật toán LZW là:

* Quy tắc 1: 256 dấu hiệu đầu tiên được dành cho các ký tự đơn (0 - 0ffh).
* Quy tắc 2: Cố gắng so sánh với "từ điển" khi trong bộ đệm chứa đã có nhiều hơn hai ký tự.
* Quy tắc 3: Các ký tự ở đầu vào (Nhận từ tập tin sẽ được nén) được bổ sung vào bộ đệm chứa đến khi chuỗi ký tự trong bộ đệm chứa không có trong "từ điển".
* Quy tắc 4: Khi bộ đệm chứa có một chuỗi mà trong "từ điển" không có thì chuỗi trong bộ đệm chứa được đem vào "từ điển". Ký tự cuối cùng của chuỗi ký tự trong bộ đệm chứa phải ở lại trong bộ đệm chứa để tiếp tục tạo thành chuỗi mới.

### **4.3 Thuật toán nén**

PSEUDOCODE

Khởi tạo “từ điển” với 256 ký tự đơn

P = first input character

While p not end character:

C = next input character

If P + C not in “ từ điển “:

P=P+C

Else:

Output the code for p

Add P + C to “từ điển”

P=C

End while

Output the code for P

Ví dụ các bước để mã hóa chuỗi “BABAABAAA”:

Bước 1: Ký tự “B” được đưa vào trong bộ đệm để chuẩn bị tạo chuỗi

Bước 2: “B” + kí tự tiếp theo = “BA”, vì “BA” chưa có trong “từ điển”, nên “BA” sẽ được thêm vào từ điển và gán mã code riêng(<256>,khác với 256 ký tự đầu trong từ điển), sau đó kí tự “B” sẽ được đưa ra và “A” ở lại trong bộ đệm, đồng thời in ra mã code của “B” là : <66>

<66>

Bước 3: “A” + kí tự tiếp theo (là B) = “AB” chưa có trong từ điển , nên tiếp tục được thêm vào và gán mã code riêng(<257>), sau đó “B” ở lại bộ đệm và A được đẩy ra và in ra mã code của “A” là <65>

<66><65>

Bước 4:”B”+ kí tự tiếp theo (là A) =”BA”, ở đây thì “BA” đã có tồn tại trong từ điển nên sẽ không thêm vào từ điển , và cũng không có kí tự nào được gửi ra, “BA” tiếp tục ở trong bộ đệm để tạo thành chuỗi mới.

Bước 5: “BA” + kí tự tiếp theo = “BAA” chưa có trong từ điển, nên sẽ được thêm vào và gán mã code(<258>), kí tự sau cùng “A” sẽ ở lại bộ đệm và “BA” được đưa ra, sau đó in ra mã code của “BA” đã thêm vào trong từ điển trước đó là <257>

<66><65><256>

Bước 6: “A” + kí từ tiếp theo = “AB” đã có trong từ điển nên giữ nguyên chuỗi “AB” trong bộ đệm và tiếp tục

Bước 7:”AB” + kí tự tiếp theo = “ABA” chưa có trong từ điển nên được thêm vào và gán mã(<257>), “A” sẽ ở lại bộ đệm và “AB” được đẩy ra và in mã code đã thêm trong từ điển trước đó (<257>)

<66><65><256><257>

Cứ tiếp tục lặp lại các bước đến hết chuỗi ta sẽ được đoạn mã

<66><65><256><257><65><260>

Trong thuật toán nén này, phần lớn thời gian khi bắt đầu nén chủ yếu mất vào việc tạo "từ điển". Khi "từ điển" đủ lớn, xác suất gặp chuỗi ở bộ đệm chứa trong "từ điển" tăng lên và càng nén được nhiều hơn. Một điều cần chú ý ở đây là mỗi một dấu hiệu, ta phải lưu một chuỗi trong "từ điển" để so sánh. Vì dấu hiệu được biểu diễn bằng một số 12 bits nên "từ điển" sẽ có 4096 lối vào, khi tăng số bit để biểu diễn dấu hiệu lên thì hiệu quả nén sẽ tốt hơn nhưng lại bị giới hạn bởi bộ nhớ của máy tính. Vì dụ, khi dùng 16 bits để biểu diễn một dấu hiệu thì "từ điển" phải có đến 65536 lối vào, nếu mỗi lối vào có khoảng 20 ký tự thì "từ điển" phải lớn khoảng 1,2 MB. Với một từ điển có dung lượng như vậy rất khó có thể thực hiện trên các máy tính PC hoạt động dưới hệ điều hành DOS vì giới hạn của một đoạn (Segment) là 64KB. Ưu điểm của phương pháp nén LZW là bên nhận có thể tự xây dựng bảng mã mà không cần bên gửi phải gửi kèm theo trong tập tin nén.

### **4.4 Thuật toán giải nén**[[5]](https://123docz.net/document/2332954-tim-hieu-thuat-toan-nen-anh-lzw-co-source-code.htm)

Quá trình giải nén được thực hiện tương tự quá trình nén, một từ mã ứng với một chuỗi sẽ được ghi ra tệp khi chuỗi ghép bởi chuổi trên với ký tự vừa đọc chưa có trong từ điển, đồng thời cập nhật ngay vào từ điển từ mã ứng với chuỗi tạo bởi chuỗi cũ với ký tự vừa đọc, và kí tự vừa đọc được dùng làm ký tự đầu trong chuỗi ứng với từ mã sẽ được ghi ra tiếp theo.

PSEUDOCODE

Khởi tạo “từ điển” với 256 ký tự đơn

OLD = first input code

output character of OLD

WHILE not end of input code:

NEW = next input code

If NEW not in the “từ điển”:

S = character of OLD

S=S + C

Else:

S= character of NEW

Output S

C = first character of S

add OLD + C to the “từ điển”

OLD = NEW

END WHILE

Ví dụ các bước giải nén đoạn mã <66><65><256><257><65><260>

Table

Description automatically generated

Các bước giải nén[[6]](https://www.geeksforgeeks.org/lzw-lempel-ziv-welch-compression-technique/)

Thuật toán này nén các chuỗi dữ liệu lặp đi lặp lại rất tốt. Vì các từ mã là 12 bit, bất kỳ ký tự mã hóa đơn nào sẽ mở rộng kích thước dữ liệu hơn là giảm nó.

**Chương 4 Thực nghiệm và đánh giá thuật toán**

1. **Thực nghiệm**

Trong quá trình thực nghiệm, chúng em đã lần lượt cho hai thuật toán chạy trên bộ data gồm 100 ảnh (.png), kết quả thu được như sau:

Thuật toán Huffman Coding : [Link colab](https://colab.research.google.com/drive/1aB8Xql20PLEgfOOzNFYcemawQfSMtYD9)

* Kích thước dữ liệu sau khi nén của mỗi ảnh giảm trung bình 45.6 % so với dữ liệu gốc
* Thời gian chạy : 36 phút (trung bình khoảng 20 giây cho 1 ảnh)
* Kích thước dữ liệu nén trung bình: 35766 bit

Thuật toán LZW : [Link colab](https://colab.research.google.com/drive/1-zsyZW7egbOo_GUljs2_MTMgTj8m6In9#scrollTo=VpRKZVNYvr19)

* Kích thước dữ liệu sau khi nén của mỗi ảnh giảm trung bình 99.99 % so với dữ liệu gốc
* Thời gian chạy : 4 phút (trung bình khoảng 2.4 giây cho 1 ảnh)
* Kích thước dữ liệu nén trung bình: 2953 bit

1. **Đánh giá thuật toán**
   1. **Huffman Coding**

Trong quá trình cài đặt cũng như chạy thực nghiệm trên bộ dữ liệu gồm 100 ảnh của nhóm thu thập nhóm nhận thấy được thuật toán này có một số nhược điểm cụ thể như:

* Quá trình mã hóa phải duyệt dữ liệu 2 lần: một lần để xác định tần suất & dựng cây mã, một lần nữa để mã hóa. Việc này khiến cho chi phí tính toán cao cũng như thời gian xử lý khá lâu.
* Khi lưu trữ dữ liệu nén phải lưu cả cây Huffman để sử dụng cho việc giải nén. Việc này làm tăng kích thước file mã hoá đáng kể.
* Việc nén cần phải có dữ liệu đầy đủ, không nén được với các dữ liệu phát sinh theo thời gian thực.
  1. **Lempel-Ziv Welch**

Thuật toán LZW đặc biệt có hiệu quả khi sử dụng để nén file văn bản vì độ lặp lại của ký tự là lớn , so với Huffman Coding thì LZW có một số điểm nổi bật hơn, tiêu biểu là sau khi thực nghiệm chúng tôi nhận thấy thuật toán tính toán khá nhanh và hoạt động hiệu quả với hầu hết các dữ liệu đặc biệt là GIF (ảnh động). Ngoài ra LZW còn có một số ưu điểm hơn so với Huffman Coding như:

* LZW không yêu cầu thông tin trước về luồng dữ liệu đầu vào.
* LZW có thể nén luồng đầu vào trong một lần truyền.

**Chương 5 Tổng kết**

Mặc dù còn một số hạn chế trong cài đặt đối với từng thuật toán, đặc biệt là cài đặt và thực nghiệm decoding dành cho thuật toán Lempel-Ziv Welch, trong nghiên cứu lần này nhóm đã cơ bản hiểu được ý tưởng, nguyên tắc hoạt động của hai thuật toán nén dữ liệu là Huffman và Lempel-Ziv Welch. Từ cơ sở đó, rút ra được một số đánh giá và so sánh hai thuật toán này dựa trên thực nghiệm trên bộ dataset nhóm thu thập được. Trong phần cài đặt thuật toán Huffman Coding của nhóm sử dụng phương pháp biến ma trận ảnh đầu vào thành một chuỗi kí tự để xử lý, cách này tuy giúp nhóm đơn giản trong việc xử lý dữ liệu vào cũng như hiện thực hoá mã giả của thuật toán, tuy nhiên đây vẫn chưa là một cách hợp lý và tối ưu nhất cụ thể là đối với những trường hợp ảnh là ảnh chụp có các pixel màu phức tạp. Việc xuất hiện nhiều pixel màu phức tạp sẽ khiến việc xử lý sẽ chậm chạp và có thể tốn rất nhiều thời gian cũng như bộ nhớ để có thể mã hoá dữ liệu dạng ảnh. Đây cũng là một trong những thách thức của nhóm trong nghiên cứu lần này mà nhóm sẽ tiếp tục nghiên cứu cũng như thử nghiệm những cách xử lý khác đối với dữ liệu đầu vào trong tương lai. Mặt khác, như đã đề cập ở trên về phần cài đặt thuật toán Lempel-Ziv Welch, đây cũng là một thách thức đối với nhóm để hiện thực mã giả decoding cho thuật toán Lempel-Ziv Welch cụ thể là cách thức cài đặt này khởi tạo một bảng mã từ những dữ liệu sau khi đã được mã hoá bằng thuật toán Lempel-Ziv Welch, chính vì thế nhóm sẽ tiếp tục nghiên cứu và cài đặt mục này trong tương lai.

# **Tài liệu tham khảo**

[[1] WinRAR](https://www.win-rar.com/fileadmin/downloads/WinRAR_Product_Information.pdf)

[[2] Unarchiver](https://en.wikipedia.org/wiki/The_Unarchiver)

[[3] Wikipedia - Image compression](https://en.wikipedia.org/wiki/Image_compression)

[[4] Wikipedia - LZ](https://en.wikipedia.org/wiki/LZ77_and_LZ78)

[[5] Tài liệu Decode LZW](https://123docz.net/document/2332954-tim-hieu-thuat-toan-nen-anh-lzw-co-source-code.htm)

[Rosettacode – LZW\_compression#Python](https://rosettacode.org/wiki/LZW_compression#Python)

[Huffman Github source code](https://github.com/tjazerzen/Huffman_encoding_decoding)