ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

BÁO CÁO MÔN HỌC

TÍNH TOÁN ĐA PHƯƠNG TIỆN

CS232.N21.KHCL

ĐỀ TÀI: NÉN VÀ GIẢI NÉN DỮ LIỆU VĂN BẢN VÀ HÌNH ẢNH

GIẢNG VIÊN HƯỚNG DẪN: **ĐỖ VĂN TIẾN**

SINH VIÊN THỰC HIỆN: **NGUYỄN VŨ DƯƠNG – 20520465**

LÊ TRẦN QUỐC KHÁNH - 20520574

TP. HỒ CHÍ MINH, 06/2023

LỜI CẢM ƠN

Mặc dù hiện nay dung lượng lưu trữ tuy không còn là một vấn đề lớn so với trước tuy nhiên việc nén dữ liệu vẫn đóng một vai trò vô cùng quan trọng trong nhiều lĩnh vực, đặc biệt là trong lưu trữ và truyền thông dữ liệu. Với dữ liệu được nén thì mình sẽ ít thời gian tải hơn so với dữ liệu gốc giúp tiết kiệm được rất nhiều thời gian. Việc nén dữ liệu cũng giúp việc truy cập và xử lý dữ liệu nhanh hơn.

Là sinh viên ngành KHMT, trong đồ án môn Tính toán đa phương tiện này, nhóm chúng em đã chọn và thực hiện đồ án "Nén và giải nén trên hai loại dữ liệu là văn bản và hình ảnh" trên các thuật toán để tìm hiểu xem cách thức nén các tập tin này như thế nào.

Nhóm xin gửi lời cảm ơn chân thành đến thầy Đỗ Văn Tiến đã tận tình giảng dạy, hướng dẫn chúng em trong suốt thời gian học vừa qua và các bạn học đã góp ý và giúp đỡ nhóm trong quá trình thực hiện đồ án này.

Do kiến thức và thời gian thực hiện hạn chế, đồ án của nhóm vẫn còn nhiều thiếu sót. Nhóm rất mong nhận được góp ý của thầy và các bạn để đồ án của nhóm được hoàn thiện.

TP. Hồ Chí Minh, tháng 6 năm 2023.

MỤC LỤC

I) Giới thiệu

1) Tổng quan

- Trong máy tính, có nhiều loại tập tin khác nhau được sử dụng để lưu trữ và tổ chức dữ liệu. Một số loại tập tin phổ biến:
    - Văn bản: Tập tin văn bản chứa các đoạn văn bản và thông tin chỉ với các ký tự và định dạng đơn giản, ví dụ như .txt (văn bản thuần túy), .doc hoặc .docx (Microsoft Word), .odt (OpenDocument Text).
    - Hình ảnh: Tập tin hình ảnh lưu trữ các hình ảnh và đồ họa, ví dụ như .jpg, .png, .gif, .bmp. Các định dạng khác nhau có độ nén và chất lượng khác nhau.
    - Âm thanh: Tập tin âm thanh chứa âm thanh và âm nhạc, ví dụ như .mp3 (MPEG Audio Layer-3), .wav (Waveform Audio File Format), .flac (Free Lossless Audio Codec).
    - Video: Tập tin video chứa dữ liệu video và âm thanh, ví dụ như .mp4 (MPEG-4), .avi (Audio Video Interleave), .mkv (Matroska Multimedia Container).
    - Bảng tính: Tập tin bảng tính lưu trữ dữ liệu trong một bảng dữ liệu có cấu trúc, ví dụ như .xlsx (Microsoft Excel), .ods (OpenDocument Spreadsheet).
    - Cơ sở dữ liệu: Tập tin cơ sở dữ liệu chứa dữ liệu có cấu trúc, có thể được truy cập và quản lý bằng các hệ quản trị cơ sở dữ liệu như MySQL, Oracle, hoặc Microsoft Access. Các định dạng phổ biến bao gồm .sql, .mdb (Microsoft Access Database), .db (SQLite Database).
    - File nén: Tập tin nén chứa một hay nhiều tệp tin hoặc thư mục đã được nén lại để giảm kích thước. Một số định dạng phổ biến bao gồm .zip, .rar, .7z.
    - Chương trình thực thi: Tập tin chương trình thực thi là các tệp tin chứa mã máy tính có thể được thực thi. Các định dạng phổ biến bao gồm .exe (Windows Executable), .app (Application Bundle trên macOS).
- Có thể thấy có nhiều loại tập tin trong máy tính tuy nhiên nhóm tụi em chọn ra hai loại tập tin mà phổ biến và sử dụng rộng rãi nhất để thực hiện quá trình nén và giải nén dữ liệu.

2) Dữ liệu văn bản

- Dữ liệu văn bản là các thông tin được biểu diễn dưới dạng văn bản, gồm các ký tự, từ ngữ, và câu. Đây là dạng dữ liệu chứa thông tin và ý nghĩa của ngôn ngữ tự nhiên được sử dụng trong việc truyền tải và lưu trữ thông tin.
- Dữ liệu văn bản có thể bao gồm nhiều loại tài liệu, bao gồm văn bản thông thường, email, bài viết, sách, báo cáo, blog, trang web, v.v. Dữ liệu văn bản thường chứa các đoạn văn, câu, từ ngữ và các yếu tố ngữ pháp khác.
- Dữ liệu văn bản được sử dụng rộng rãi trong nhiều lĩnh vực và ứng dụng. Ví dụ, trong xử lý ngôn ngữ tự nhiên, dữ liệu văn bản được sử dụng để phân loại văn bản, trích xuất thông tin, dịch thuật, phân tích ý kiến, v.v. Trong lĩnh vực trích xuất thông tin, dữ liệu văn bản được sử dụng để tìm kiếm và trích xuất thông tin cụ thể từ các tài liệu. Trong lĩnh vực truyền thông và mạng xã hội, dữ liệu văn bản được sử dụng để phân tích nội dung, phát hiện tin giả và phân loại tâm trạng, v.v.
- Có 3 loại dữ liệu văn bản:
    - Dữ liệu văn bản cấu trúc (Structured Text Data): Đây là loại dữ liệu văn bản được tổ chức theo một cấu trúc nhất định, ví dụ như bảng dữ liệu hoặc các tệp tin văn bản có định dạng cụ thể. Dữ liệu văn bản cấu trúc thường được biểu diễn bằng các dạng dữ liệu như CSV (Comma-Separated Values), JSON (JavaScript Object Notation), XML (eXtensible Markup Language) và các tệp tin cơ sở dữ liệu. Loại dữ liệu này thường dễ dàng xử lý và phân tích, và được sử dụng rộng rãi trong lĩnh vực phân tích dữ liệu và trích xuất thông tin.
    - Dữ liệu văn bản bán cấu trúc (Semi-Structured Text Data): Đây là loại dữ liệu văn bản không tuân theo một cấu trúc nghiêm ngặt, nhưng vẫn chứa các thông tin quan trọng được gắn kết với các nhãn hoặc thẻ. Ví dụ, các tài liệu HTML, XHTML hoặc tệp tin log có thể được coi là dữ liệu văn bản bán cấu trúc. Loại dữ liệu này thường yêu cầu các kỹ thuật xử lý và trích xuất thông tin đặc biệt để trích xuất dữ liệu từ các nhãn hoặc thẻ.
    - Dữ liệu văn bản không cấu trúc (Unstructured Text Data): Đây là loại dữ liệu văn bản không có cấu trúc rõ ràng, không tuân theo bất kỳ định dạng hay quy tắc cụ thể nào. Đây là dạng dữ liệu tự do, không giới hạn và đa dạng như email, tin nhắn văn bản, bài viết blog, sách, trang web, v.v. Dữ liệu văn bản không cấu trúc thường chứa thông tin phong phú và có thể chứa nhiều định dạng khác nhau như văn bản, hình ảnh, đường dẫn, biểu

đồ, v.v. Xử lý dữ liệu văn bản không cấu trúc đòi hỏi các kỹ thuật và công cụ phức tạp như xử lý ngôn ngữ tự nhiên và khai thác dữ liệu văn bản để trích xuất thông tin và hiểu nội dung của văn bản.

- Với đồ án này:
    - Huffman: Thực hiện nén và giải nén trên cả ba loại dữ liệu văn bản.
    - LZW: Thực hiện nén và giải nén chỉ trên loại dữ liệu không cấu trúc.

3) Dữ liệu hình ảnh

- Dữ liệu hình ảnh là dạng dữ liệu biểu diễn thông tin dưới dạng hình ảnh hoặc đồ họa. Nó chứa các dữ liệu liên quan đến màu sắc, độ phân giải, chiều cao, chiều rộng và các thông tin khác liên quan đến hình ảnh.
- Trong máy tính, dữ liệu hình ảnh là một loại dữ liệu được biểu diễn và xử lý bằng các phương pháp và công cụ đặc biệt dành cho xử lý hình ảnh và đồ họa. Dữ liệu hình ảnh trong máy tính thường được biểu diễn dưới dạng các tệp hình ảnh kỹ thuật số, chẳng hạn như JPEG, PNG, GIF, BMP, TIFF và nhiều định dạng khác.
- Hình ảnh màu trong máy tính thường được biểu diễn bằng mô hình màu RGB (Red, Green, Blue). Theo mô hình này, mỗi pixel trong hình ảnh chứa thông tin về mức độ đỏ (R), mức độ xanh lá cây (G) và mức độ xanh dương (B). Các giá trị R, G và B thường được biểu diễn bằng các con số từ 0 đến 255, trong đó 0 đại diện cho mức đen tuyệt đối và 255 đại diện cho mức trắng tuyệt đối. Bằng cách kết hợp các giá trị R, G và B, máy tính tạo ra một màu sắc cụ thể cho mỗi pixel.
- Hình ảnh đơn sắc (grayscale) trong máy tính được biểu diễn bằng một kênh duy nhất, thể hiện mức độ ánh sáng của từng pixel.
- Có 3 loại kiểu dữ liệu phổ biến về hình ảnh:
    - Raster (Bitmap): Đây là kiểu dữ liệu hình ảnh phổ biến nhất, biểu diễn hình ảnh dưới dạng một mảng hai chiều các điểm ảnh (pixels). Mỗi điểm ảnh có thông tin về màu sắc và độ sáng.
    - Vector: Kiểu dữ liệu hình ảnh vector biểu diễn hình ảnh dưới dạng các đối tượng hình học, ví dụ như đường thẳng, đa giác, v.v. Thông tin hình ảnh được lưu trữ dưới dạng các công thức toán học, cho phép hình ảnh có thể được phóng to hoặc thu nhỏ mà không mất đi chất lượng. Các định dạng hình ảnh vector phổ biến bao gồm SVG (.svg), AI (.ai), EPS (.eps) và PDF (.pdf).
    - Hình ảnh động (Animated): Đây là kiểu dữ liệu hình ảnh được sử dụng để tạo ra hình ảnh chuyển động. Nó chứa một chuỗi các hình ảnh tĩnh liên tiếp, được phát lại theo một tốc độ nhất định

tạo thành một hiệu ứng chuyển động. Định dạng hình ảnh động phổ biến bao gồm GIF (.gif) và APNG (.apng).

- Loại kiểu dữ liệu hình ảnh tụi em chọn để làm đồ án là bitmap(Raster).

4) Nén dữ liệu

- Nén dữ liệu (Data compression) là quá trình giảm kích thước dữ liệu để tiết kiệm không gian lưu trữ hoặc băng thông mạng mà không làm mất đi thông tin quan trọng. Khi dữ liệu được nén, các thuật toán nén được áp dụng để loại bỏ các dạng lặp lại, sự thừa thãi hoặc các mẫu không cần thiết khác trong dữ liệu. Kết quả là một phiên bản nén của dữ liệu gốc có kích thước nhỏ hơn.
- Có hai loại chính của nén dữ liệu:
    - Nén dữ liệu không mất mát (Lossless Compression): Loại nén này giữ nguyên tất cả thông tin ban đầu khi giảm kích thước của dữ liệu. Khi giải nén, dữ liệu được khôi phục chính xác như trước khi nén. Phương pháp nén không mất mát được sử dụng cho các tập tin và dữ liệu mà việc mất mát thông tin là không chấp nhận được, ví dụ như tập tin văn bản, hồ sơ, tệp tin lưu trữ, v.v. Các phương pháp nén không mất mát phổ biến bao gồm gzip, ZIP, RAR, …
    - Nén dữ liệu mất mát (Lossy Compression): Loại nén này giảm kích thước của dữ liệu bằng cách loại bỏ một phần thông tin không cần thiết hoặc ít quan trọng. Khi giải nén, dữ liệu không thể được khôi phục chính xác như ban đầu và có thể mất một phần thông tin gốc. Phương pháp nén mất mát thường được sử dụng cho dữ liệu nhạc, video, hình ảnh, nơi một mức độ nhất định của mất mát không ảnh hưởng nhiều đến chất lượng. Các phương pháp nén mất mát phổ biến bao gồm MPEG, JPEG, và MP3.
- Tầm quan trọng của việc nén dữ liệu:
    - Tiết kiệm không gian lưu trữ: Khi dữ liệu được nén, kích thước của nó giảm đi. Điều này giúp tiết kiệm không gian lưu trữ trên ổ cứng, bộ nhớ hoặc các phương tiện lưu trữ khác. Đặc biệt trong các ứng dụng yêu cầu lưu trữ lớn như hệ thống tập tin, cơ sở dữ liệu hoặc lưu trữ đám mây, việc nén dữ liệu là rất quan trọng để giảm chi phí và tăng khả năng lưu trữ.
    - Tăng tốc độ truyền dữ liệu: Khi dữ liệu được nén, kích thước truyền đi giảm đi. Điều này có nghĩa là việc truyền dữ liệu qua mạng hoặc qua các kênh truyền thông sẽ nhanh hơn. Nén dữ liệu

làm giảm lượng dữ liệu cần truyền, giúp giảm thời gian truyền tải và tăng tốc độ truyền dữ liệu.

- Tiết kiệm băng thông: Khi truyền dữ liệu qua mạng, việc nén dữ liệu giúp giảm lượng dữ liệu cần truyền đi. Điều này đồng nghĩa với việc tiết kiệm băng thông mạng. Đối với các hệ thống mạng có băng thông hạn chế hoặc đắt đỏ, việc nén dữ liệu làm giảm lượng dữ liệu truyền qua mạng và giúp tăng hiệu suất mạng.
- Bảo vệ dữ liệu: Một khía cạnh quan trọng khác của nén dữ liệu là bảo vệ dữ liệu. Trong quá trình nén, dữ liệu được biến đổi và mã hóa, làm cho nó khó hiểu và khó bị tấn công bởi người không có quyền truy cập. Điều này cung cấp một mức độ bảo mật nhất định cho dữ liệu, đặc biệt là khi chúng được truyền qua mạng công cộng hoặc lưu trữ trong các hệ thống không đáng tin cậy.
- Tối ưu hóa hiệu suất hệ thống: Khi dữ liệu được nén, việc truy cập và xử lý dữ liệu trở nên nhanh hơn. Thời gian truy cập vào dữ liệu được rút ngắn, và các hoạt động như sao chép, di chuyển và sao lưu cũng trở nên nhanh chóng hơn. Nén dữ liệu giúp tối ưu hóa hiệu suất hệ thống và cải thiện trải nghiệm người dùng.

5) Giải nén dữ liệu

- Giải nén dữ liệu (Data decompression) là quá trình khôi phục lại dữ liệu gốc từ một tệp tin hoặc dữ liệu đã được nén trước đó. Khi dữ liệu được nén, nó được biến đổi hoặc mã hóa thành một định dạng khác có kích thước nhỏ hơn, nhưng vẫn chứa thông tin cần thiết để khôi phục lại dữ liệu gốc khi được giải nén.
- Quá trình giải nén dữ liệu thường được thực hiện bằng cách sử dụng các thuật toán và công cụ giải nén tương ứng với phương pháp nén đã được sử dụng. Ví dụ, nếu dữ liệu đã được nén bằng phương pháp nén không mất mát, quá trình giải nén sẽ khôi phục lại dữ liệu gốc mà không mất đi bất kỳ thông tin nào. Trong trường hợp dữ liệu đã được nén bằng phương pháp nén mất mát, quá trình giải nén sẽ khôi phục lại dữ liệu gốc nhưng có thể mất một phần thông tin ban đầu.

II) Thuật toán nén Huffman

1) Giới thiệu

- Thuật toán Huffman là một thuật toán nén dữ liệu không mất mát, được phát triển bởi David Huffman vào đầu những năm 50 khi ông là một sinh viên nghiên cứu tiến sĩ tại MIT. Thuật toán này dựa trên phương pháp sắp xếp theo tần suất của cây nhị phân, cho phép mã hóa bất kỳ chuỗi thông điệp nào thành các thông điệp mã hóa ngắn hơn và có phương pháp để tái lắp thành thông điệp ban đầu mà không mất mát dữ liệu nào.

2) Nguyên lý

- Nguyên lý của phương pháp Huffman là mã hóa các bytes trong tệp dữ liệu nguồn bằng biến nhị phân. Nó tạo mã độ dài biến thiên là một tập hợp các bits. Đây là phương pháp nén kiểu thống kê, những ký tự xuất hiện nhiều hơn sẽ có mã ngắn hơn.
- Thông thường, dữ liệu văn bản được lưu trữ ở định dạng chuẩn 8 bit cho mỗi ký tự, sử dụng mã hóa ASCII ánh xạ mỗi ký tự thành một giá trị số nguyên nhị phân từ 0-255. Ý tưởng của mã hóa Huffman là từ bỏ yêu cầu cứng nhắc 8 bit cho mỗi ký tự và thay vào đó sử dụng mã hóa nhị phân có độ dài khác nhau cho các ký tự khác nhau. Ưu điểm của việc này là nếu một ký tự xuất hiện thường xuyên trong tệp, chẳng hạn như chữ cái 'e' rất phổ biến, thì nó có thể được mã hóa ngắn hơn (tức là ít bit hơn), làm cho tệp tổng thể nhỏ hơn. Sự cân bằng là một số ký tự có thể cần phải sử dụng các mã hóa dài hơn 8 bit, nhưng điều này được dành riêng cho các ký tự xảy ra không thường xuyên, vì vậy, số dư phải trả thêm.

3) Thuật toán

3.1) Thuật toán nén

1: Xây dựng bảng thống kê tần số xuất hiện của các ký tự trong dữ liệu cần nén.
2: Tạo một nút lá cho mỗi ký tự và gán tần số xuất hiện của ký tự đó cho nút lá tương ứng.
3: Xếp các nút lá vào một hàng đợi ưu tiên (priority queue), sắp xếp theo tần số xuất hiện. Nút lá với tần số xuất hiện thấp nhất có độ ưu tiên cao nhất.

4: Lặp lại quá trình sau cho đến khi chỉ còn lại một nút duy nhất trong hàng đợi ưu tiên:

+ Lấy ra hai nút có độ ưu tiên cao nhất từ hàng đợi ưu tiên, tạo một nút mới là nút cha của hai nút này.

+ Tạo một nút mới là nút cha của hai nút này và gán tần số của nút cha bằng tổng tần số của hai nút con.

+ Đưa nút cha vào hàng đợi ưu tiên.

5: Nút cuối cùng trong hàng đợi ưu tiên sẽ là nút gốc của cây.

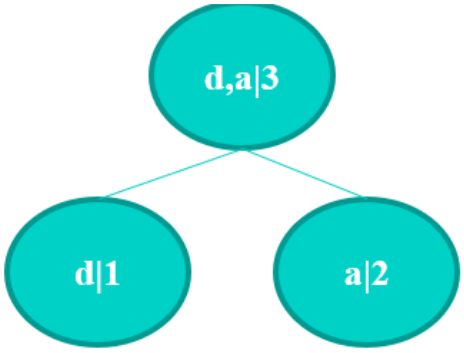
Để dễ hình dung cách tạo ra cây Huffman thì cho ví dụ cho chữ "abbcadeec"

- Bước 1: Xây dựng bảng thống kê.

Ký tự	Số lần xuất hiện
e	2
c	2
b	2
a	2
d	1

- Bước 2: Chọn 2 ký tự có số lần xuất hiện bé nhất (nếu cùng thì chọn theo bảng alphabet đứng trước thì chọn trước) để xây dựng xây xong thì cập nhật lại cây và cập nhật lại bảng.

Cây huffman:

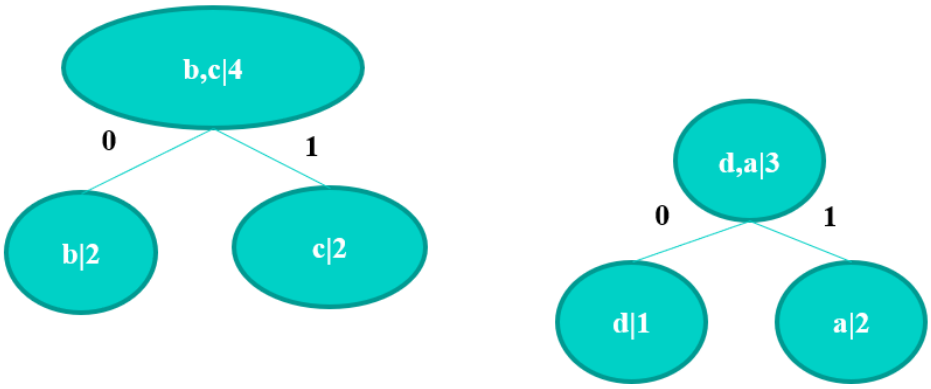


Bảng thống kê

Ký tự	Số lần xuất hiện
d,a	3
e	2
c	2
b	2

- Bước 3: Làm tương tự bước 2 ta có

Cây Huffman

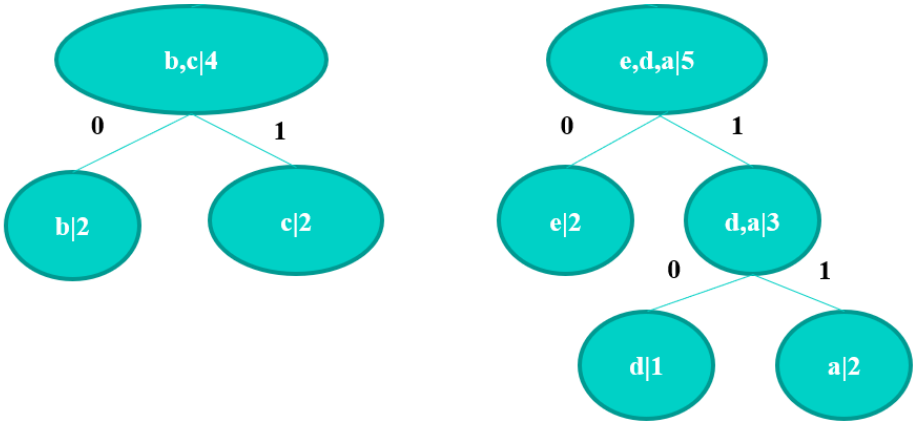


Bảng thống kê

Ký tự	Số lần xuất hiện
b.c	4
d,a	3
e	2

- Bước 4:

Cây Huffman

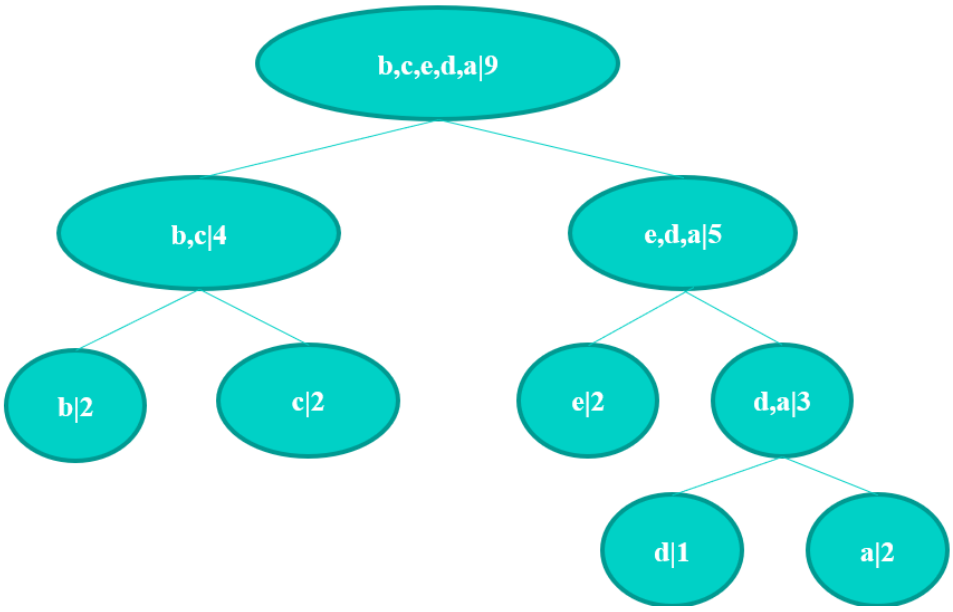


Bảng thống kê

Ký tự	Số lần xuất hiện
e,d,a	5
b.c	4

- Bước 5:

Cây Huffman

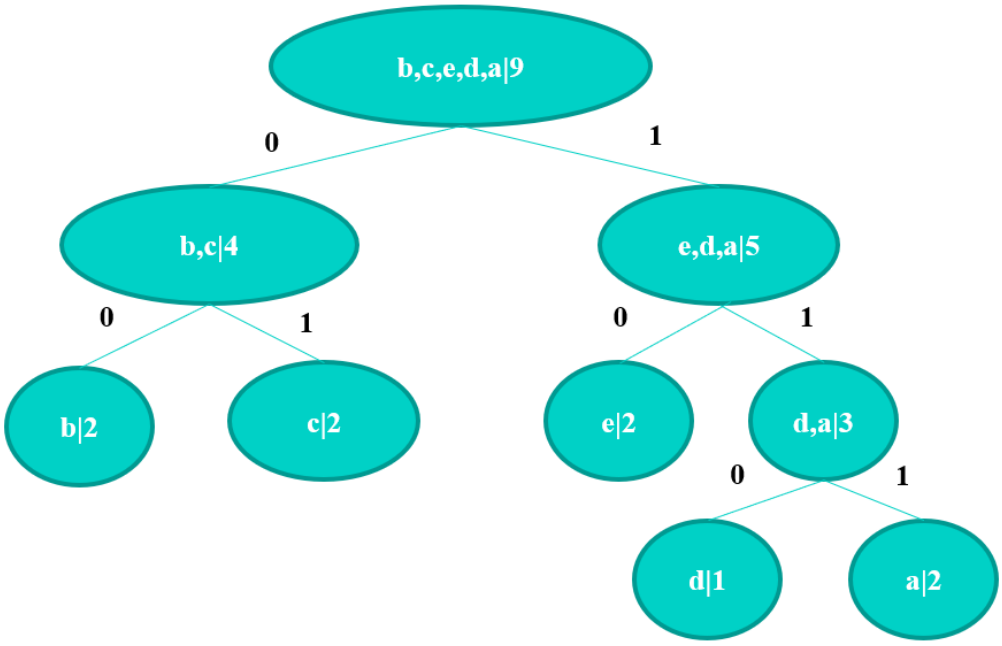


Bảng thống kê

Ký tự	Số lần xuất hiện
b.c,e,d,a	9

- Bước 6: Thêm các giá trị 0 và 1 ứng với cây con trái và cây con phải

Cây Huffman



Bảng thống kê

Ký tự	Số lần xuất hiện	Code
e	2	10
c	2	01
b	2	00
a	2	111
d	1	110

➔ Kết quả: 111000001111110101001

3.2 Thuật toán giải nén

Các bước giải nén của thuật toán Huffman:

+ Đọc dữ liệu nén: Đầu tiên, đọc dữ liệu nén đã được lưu trữ hoặc truyền tải.

+ Đọc bảng mã Huffman: Tiếp theo, đọc bảng mã Huffman đã được tạo ra trong quá trình nén dữ liệu. Bảng mã này phải đi kèm với dữ liệu nén hoặc được lưu trữ/được truyền tải một cách riêng biệt.

+ Tạo cây Huffman: Sử dụng thông tin từ bảng mã Huffman, cần tạo cây Huffman ban đầu. Cây Huffman được tạo bằng cách sắp xếp các ký tự theo tần suất xuất hiện và xây dựng cây từ các nút lá đến nút gốc. Cách xây dựng cây Huffman đã được sử dụng trong quá trình nén phải được lặp lại ở đây để tạo cấu trúc cây giống nhau.

+ Giải mã dữ liệu: Bắt đầu từ gốc của cây Huffman, duyệt qua cây bằng cách di chuyển xuống theo các nhánh trái (bit 0) và phải (bit 1) dựa trên các bit trong dữ liệu nén. Khi đạt được một nút lá, hãy ghi ký tự tương ứng trong bảng mã Huffman. Tiếp tục lặp lại quá trình này cho đến khi đã giải mã toàn bộ dữ liệu nén.

+ Ghi dữ liệu giải nén: Khi đã giải mã toàn bộ dữ liệu nén, hãy ghi dữ liệu giải nén ra file hoặc hiển thị nó trên màn hình.

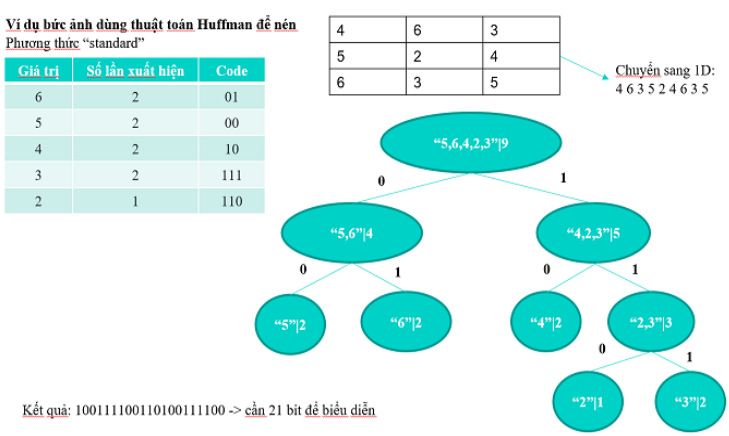
- Để dễ hình dung ta sẽ giải mã chuỗi bit sau “11100000111111010101001” và ta có bảng từ điển như sau

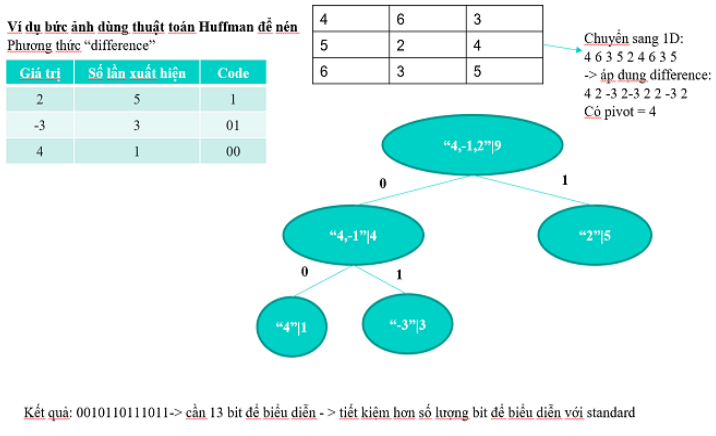
Ký tự	Số lần xuất hiện	Code
e	2	10
c	2	01
b	2	00
a	2	111
d	1	110

- Bước 1: Đầu tiên ta duyệt từ đầu bit 1 11100000111111010101001 ta thấy bit 1 không có trong từ điển do đó dịch thêm 1 bit nữa.
- Bước 2: Từ bước trên bit tiếp theo sẽ là 11 11100000111111010101001 ta thấy nó y chang bước 1 tiếp tục dịch thêm 1 bit.
- Bước 3: Bit tiếp theo là 111 11100000111111010101001 ta thấy bit “111” ứng với chữ “a” ta chuyển đổi thành a và đặt lại bit bắt đầu tại 111 0 0000111111010101001.
- Thực hiện liên tục đến cuối chuỗi ta sẽ được y như ký tự ban đầu là “abbcadeec”.

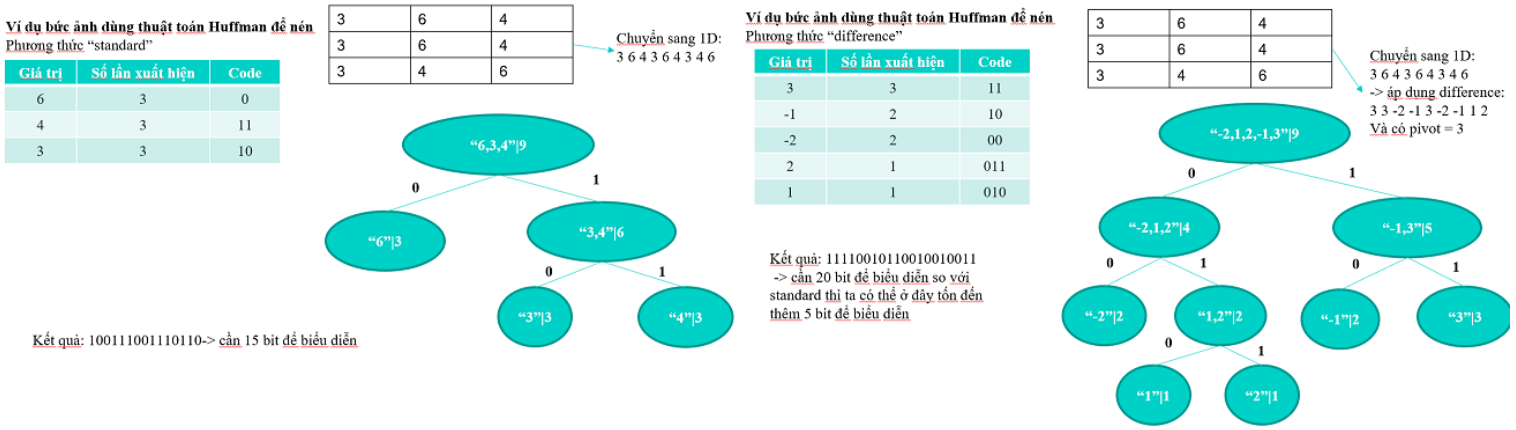
4) Nén dữ liệu hình ảnh

- Trong máy tính thì hình ảnh sẽ được biểu diễn theo ma trận có số dòng và cột tương ứng với chiều dài và chiều rộng của bức ảnh .Tại mỗi vị trị trong ma trận này chứa 1 giá trị pixel (giá trị của điểm ảnh) và giá trị của nó nằm trong khoảng từ 0 đến 255. Vì các giá trị này là các con số nên sẽ có 2 cách thức để nén bằng Huffman:
    - Cách thức "Standard": Với các thức này thì ta cần chuyển ma trận hình ảnh sang thành ma trận 1 hàng sau đó thực hiện nén thì y chang với thuật toán nén của Huffman. Còn đối với giải nén thì ta giải nén bình thường sau khi giải nén xong thì ta phải reshape lại kết quả đó với kích thước ban đầu của bức ảnh
    - Cách thức "Difference": Đầu tiên ta cũng chuyển đổi thành ma trận 1 dòng sau đó ta lấy giá trị pixel của điểm đầu tiên của ma trận dòng đó có tên là pivot và lưu trữ nó . Sau đó, tính toán sự khác biệt giữa các giá trị pixel liền kề. Điều này có thể được thực hiện bằng cách lấy hiệu của giá trị pixel tiếp theo và giá trị pixel hiện tại trên cùng một hàng hoặc cột -> kết quả sẽ tạo ra một ma trận các giá trị khác biệt. Lấy ma trận khác biệt này áp dụng thuật toán Huffman để nén và làm ngược lại để giải nén.
- Tuy nhiên 2 phương thức này phương thức nào hiệu quả hơn tùy thuộc vào bức ảnh
  Để hình dung dễ dàng sẽ có 2 VD sau:
- "Difference" tốt hơn "Standard":





- “Standard” tốt hơn “Difference”



5) Dữ liệu hình ảnh màu

- Với phương pháp trình bày bằng cách reshape ma trận thành ma trận một dòng thì chỉ phù hợp với với bức ảnh trắng đen còn đối với bức ảnh màu thì nó sẽ có ma trận của nó sẽ có kích thước là m*n*3 (với m , n là chiều dài và chiều rộng của bức ảnh và 3 thể hiện cho 3 kênh màu là red green blue). Nếu ta sử dụng phương pháp trên thì sẽ không giúp tối ưu cho việc nén giữ liệu bằng Huffman vì

  + Các kênh màu trong mô hình RGB tạo ra một loạt các giá trị màu từ 0 đến 255. Bằng cách chia thành 3 kênh màu riêng biệt, mỗi kênh chỉ chứa thông tin về một màu cụ thể, điều này cho phép chúng ta áp dụng các kỹ thuật nén dữ liệu độc lập lên từng kênh màu.

  + Khi áp dụng Huffman lên từng kênh màu, ta có thể tận dụng tính chất thống kê của dữ liệu màu trong kênh đó và xây dựng bảng mã Huffman tối ưu cho từng kênh riêng biệt. Điều này cho phép ta mã hóa các giá trị màu xuất hiện thường xuyên trong mỗi kênh màu bằng các mã ngắn hơn, trong khi các giá trị màu xuất hiện ít thường xuyên sẽ được mã hóa bằng các mã dài hơn. Quá trình này giúp giảm kích thước tập tin ảnh màu mà vẫn giữ lại chất lượng hình ảnh đáng kể.

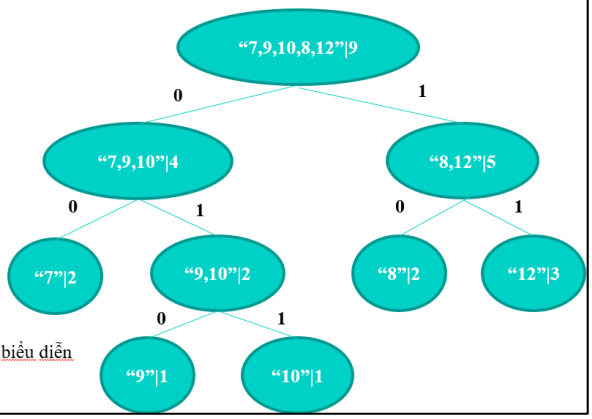
  -> Nén ảnh màu bằng huffman phải chia thành 3 kênh và nén trên từng kênh.
- Với nén ảnh màu ta cũng có 2 phương pháp nén là tương tự là “standard” và “difference” và áp dụng 2 phương pháp này cho 3 kênh tuy nhiên trong từ điển ta cần phải thêm đó chính là độ dài của bit sau khi được nén trên từng kênh dể là công cụ cho quá trình giải nén và đầu ra sẽ là tổng tất cả các bit của các kênh khi được nén.
- Có thể dễ dàng chứng minh được bằng ví dụ sau đây:

Ví dụ ảnh màu dùng thuật toán Huffman để nén

Phương thức reshape thành 1D

Giá trị	Số lần xuất hiện	Code
12	3	11
8	2	10
7	2	00
10	1	011
9	1	010

Các giá trị của các kênh màu:
R([12,7,8]) ; G([12,9,10]) ; B([8,7,12]) → Chuyển sang 1D:
12 7 8 12 9 10 8 7 12



Kết quả: 11001011010011100011 -> cần 20 bit để biểu diễn

Ví dụ ảnh màu dùng thuật toán Huffman để nén

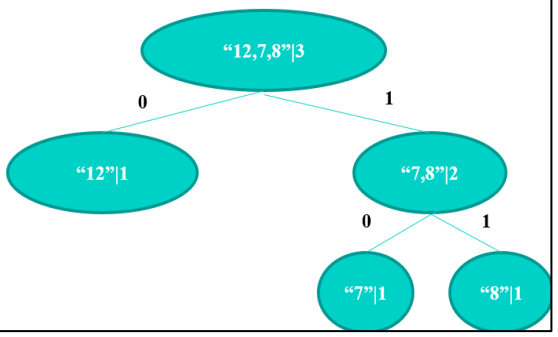
Phương thức chia thành 3 kênh và nén

Giá trị	Số lần xuất hiện	Code
12	1	0
8	1	11
7	1	10

Các giá trị của các kênh màu:
R([12,7,8]) ; G([12,9,10]) ; B([8,7,12])

Lấy kênh màu đỏ và Chuyển sang 1D:
12 7 8

Kết quả:01011 -> 5 bit để nén kênh đỏ



Ví dụ ảnh màu dùng thuật toán Huffman để nén

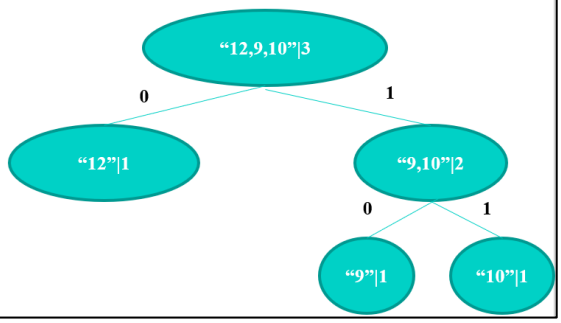
Phương thức chia thành 3 kênh và nén

Giá trị	Số lần xuất hiện	Code
12	1	0
10	1	11
9	1	10

Các giá trị của các kênh màu:
R([12,7,8]) ; G([12,9,10]) ; B([8,7,12])

Lấy kênh màu xanh lá và Chuyển sang 1D:
12 9 10

Kết quả:01011 -> 5 bit để nén kênh xanh lá



Ví dụ ảnh màu dùng thuật toán Huffman để nén

Phương thức chia thành 3 kênh và nén

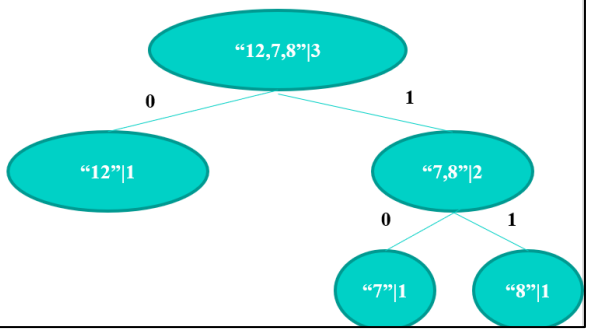
Giá trị	Số lần xuất hiện	Code
12	1	0
8	1	11
7	1	10

Các giá trị của các kênh màu:
R([12,7,8]) ; G([12,9,10]) ; B([8,7,12])

Lấy kênh màu xanh dương và Chuyển sang 1D:
8 7 12

Kết quả:11100 -> 5 bit để nén kênh xanh dương

Sau khi thực hiện nén từng kênh xong ta gộp các giá trị bit sẽ được giá trị của bức hình biểu diễn bằng Huffman như sau:
01011 01011 11100 -> cần 15 bit để biểu diễn tiết kiệm đến 5 bit so với việc mình chỉ chuyển đổi thành 1 chiều.



6) Lưu trữ file nén

- File nén thuật toán của em sẽ gồm 2 thứ:
  + File bin: chứa các thông tin của input sau khi qua thuật toán Huffman nén thành các bit.
  + File .csv: đây chính là file từ điển chứa các thông tin giúp cho quá trình giải nén dữ liệu
- Có một vấn đề xảy ra là trong máy tính, các tệp tin thường được lưu trữ dưới dạng byte. Byte là một đơn vị thông tin kỹ thuật số bao gồm 8 bit. Mỗi bit là một chữ số nhị phân, đại diện cho giá trị 0 hoặc 1. Byte được sử dụng để đo kích thước của các tệp tin, cũng như dung lượng lưu trữ trên các thiết bị lưu trữ của máy tính như ổ cứng, ổ đĩa rắn (SSD) hoặc ổ đĩa flash
- Để giải quyết vấn đề đó thì em chỉ cần kiểm tra giá trị bit sau khi thực nén có chia hết cho 8 hay không:
  + Nếu chia hết thì ta chuyển chuỗi bit thành byte.
  + Nếu chưa thì ta thêm bit 0 và cuối dãy bit mã hóa cho đến khi nào vừa đủ để thành byte sau đó chuyển sang dạng byte.
- Vậy quá trình giải nén giải quyết vấn đề thêm bit này ra sao đơn giản trong từ điển ta thêm 1 phần là extra_length nó chứa số lượng bit 0 được thêm vào.

7) Hàm đánh giá

7.1) Quá trình nén

- Average Length (Độ dài trung bình) để đo lường độ dài trung bình của mã bit được sử dụng để biểu diễn mỗi ký tự sau khi áp dụng thuật toán Huffman. Average length càng nhỏ thì thuật toán nén càng tối ưu. Khi average length nhỏ, có nghĩa là số lượng bit cần thiết để biểu diễn mỗi ký tự trong tập dữ liệu sau khi áp dụng thuật toán nén càng ít. Điều này có nghĩa là thuật toán nén giảm kích thước tệp tin gốc một cách hiệu quả. (đơn vị bits/symbol)

Công thức tính:

$$\text{Average Length} = \sum P(x) * l(x)$$

Trong đó: P(x) là xác suất xuất hiện của ký tự x trong tập dữ liệu

l(x) là độ dài của mã bit biểu diễn ký tự x.

- Entropy được sử dụng để đánh giá khả năng nén của thuật toán. Nếu entropy của dữ liệu ban đầu là cao, tức là thông tin không chắc chắn và không có sự xuất hiện đồng đều của các ký tự. Ngược lại, nếu entropy thấp, tức là thông tin có sự xuất hiện đồng đều của các ký tự, thuật toán Huffman có thể không đạt được mức nén tốt nhất. Mức độ nén tốt nhất sẽ được đạt khi entropy là thấp nhất.(đơn vị bits/symbol)

Công thức tính:

$$\text{Entropy} = \sum P(x) * log_2(P(x))$$

Trong đó: P(x) là xác suất xuất hiện của ký tự x trong tập dữ liệu

Nếu Average Length càng gần đến Entropy của dữ liệu ban đầu, tức là mã hóa Huffman đã gần tiếp cận giới hạn tối ưu của việc nén.

- Compression ratio (tỷ lệ nén) là một độ đo quan trọng để đánh giá hiệu quả của thuật toán nén dữ liệu. Nó đo lường mức độ nén được đạt được bằng cách so sánh kích thước tệp tin gốc và kích thước tệp tin sau khi áp dụng thuật toán nén.

Công thức tính:

Compression ratio = (Kích thước tệp tin gốc) / (Kích thước tệp tin sau nén)

Nếu giá trị của compression ratio > 1 thì thể hiện rằng kích thước của tệp tin nén nhỏ hơn tệp gốc tỉ lệ này càng cao thì file nén có kích thước càng nhỏ. Nếu compression ratio < 1 thì nói lên rằng kích thước của tệp tin nén lớn hơn tệp tin gốc -> thuật toán nén không được hiệu quả.

- Tuy nhiên để xác định kích thước tệp tin gốc của bức ảnh là hơi khó vì các bức ảnh chủ yếu được lưu dưới dạng png và jpeg. Thế nhưng các loại tập tin này lại chính là file nén thế nhưng vẫn có cách để tính được bằng cách nhân kích thước ảnh với độ sâu bit và số kênh màu(đơn vị là bit)

Dung lượng (byte) = (Chiều rộng) x (Chiều cao) x (Độ sâu bit) x (Số kênh màu)

- Execution time (Thời gian thực thi) là thời gian mà thuật toán nén dữ liệu mất để thực hiện quá trình nén và giải nén trên tệp tin. Thời gian thực thi càng ngắn thì tốc độ nén hoặc giải nén của thuật toán nén càng nhanh và ngược lại

7.2) Quá trình giải nén

+ Đối với file dạng văn bản:

Bước 1: Thử đọc file gốc. Nếu thành công, tiếp tục sang bước 2. Nếu không, trả về giá trị -1 để biểu thị lỗi đọc file.

Bước 2: Duyệt qua từng ký tự trong file gốc và file được giải nén, và kiểm tra xem hai ký tự tương ứng có khác nhau hay

không. Nếu hai ký tự khác nhau, tăng giá trị biến "different" lên một đơn vị.

Bước 3: Trả về giá trị biến "different", đại diện cho số lượng ký tự khác nhau giữa hai file.

+ Đối với file hình ảnh:

Bước 1: Thử đọc file gốc. Nếu thành công, tiếp tục sang bước 2. Nếu không, trả về giá trị -1 để biểu thị lỗi đọc file.

Bước 2: Kiểm tra xem file ảnh giải nén và file gốc có cùng loại ảnh hay không. Nếu file ảnh giải nén là ảnh xám trong khi file gốc là ảnh màu, chuyển đổi ảnh gốc sang ảnh xám để so sánh.

Bước 3: Duyệt qua từng pixel của ảnh gốc và ảnh giải nén, và kiểm tra xem hai giá trị pixel tương ứng có khác nhau hay không. Nếu hai giá trị pixel khác nhau, tăng giá trị biến "different" lên một đơn vị.

Bước 4: Trả về giá trị biến "different", đại diện cho số lượng pixel khác nhau giữa hai ảnh.

- Quá trình này sẽ giúp đánh giá chất lượng kết quả nén bằng cách đếm số lượng ký tự hoặc pixel khác nhau giữa file gốc và file giải nén. Kết quả càng gần 0, tức là ít khác biệt, thì thuật toán nén càng hiệu quả.

III) LZW

1) Giới thiệu

Khái niệm nén từ điển được Jacob **L**ampel và Abraham **Z**iv đưa ra lần đầu tiên vào năm 1977. Sau đó phát triển thành một họ giải thuật nén từ điển LZ. Năm1984, Terry **W**elch đã cải tiến thuật giải LZ thành một họ giải thuật mới hiệu quả hơn và đặt tên là **LZW.** LZW là thuật toán nén không mất dữ liệu được sử dụng phổ biến trong định dạng file GIF.

2) Nguyên lý

Ý tưởng chính: LZW xây dựng một từ điển lưu các chuỗi có tần suất xuất hiện cao trong dữ liệu. Từ điển là tập hợp những cặp mã và giá trị của chuỗi. Đặc biệt LZW không dùng lại bộ từ điển trong quá trình mã hóa.

1. Mã từ 0 đến 255 miêu tả 1 dãy ký tự thay thế cho ký tự 8-bit tương ứng.

2. Mã từ 256 được tạo bên trong từ điển cho trường hợp lặp chuỗi trong dữ liệu.

3. Mỗi bước trong khi nén, giá trị nhập vào được tập hợp lại thành chuỗi cho đến khi ký tự tiếp theo sẽ tạo thành chuỗi chưa có trong từ điển, và 1 mã mới cho chuỗi được tạo sẽ được thêm vào từ điển,và mã đó bỏ đi giá trị cuối sẽ được xuất ra output.

3) Thuật toán

3.1) Thuật toán nén

Các bước thực hiện của thuật toán nén LZW:

1. Đọc lần lượt từng giá trị đầu vào vào chuỗi đang xét cho đến khi chuỗi chưa có trong từ điển.

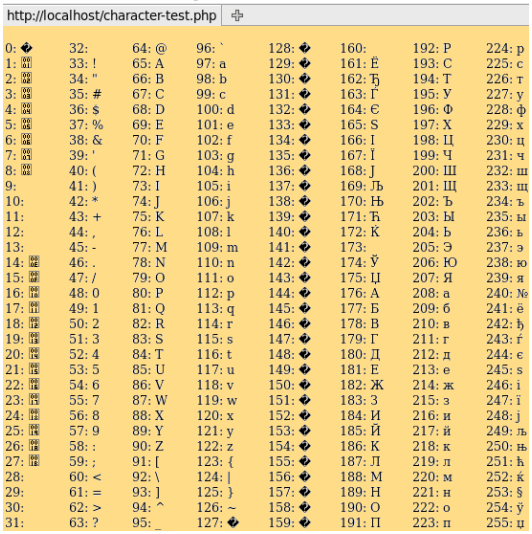
2. Tạo từ mới trong từ điển với giá trị là chuỗi đang xét và tổng số mã trong từ điển tăng lên 1 và gán cho mã mới.

3. Xuất ra output là giá trị chuỗi đang xét (bỏ đi ký tự cuối).

4. Ký tự cuối tiếp tục kết hợp với từng giá trị đầu vào cho đến khi hết chuỗi đầu vào.

Ví dụ minh họa:

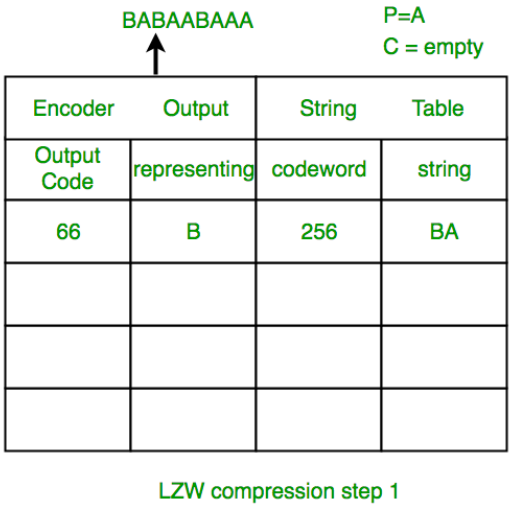
Cho bộ từ điển ban đầu có giá trị từ 0 – 255 như sau:



Hãy nén chuỗi : “BABAABAAA” bằng thuật toán LZW

Bước 1:

- Đọc lần lượt từng từ trái sang phải của chuỗi đầu vào, ta có ký tự “B”
- Do “B” đã có trong từ điển nên ta tiếp tục thêm ký tự tiếp theo của chuỗi đầu vào là “A”
- Chuỗi “BA” chưa có sẽ được thêm vào từ điển với mã 256



LZW compression step 1

- Chuỗi xuất ra sẽ là “BA” bỏ đi ký tự cuối là “A” nên “B” sẽ được xuất ra, “A” sẽ được tiếp tục đi xét

Bước 2:

- “A” + ký tự tiếp theo của chuỗi đầu vào là “B” tạo thành “AB” chưa có trong từ điển nên mã mới được tạo.

BABAABAAA

P=B
C = empty

Encoder	Output	String	Table
Output Code	representing	codeword	string
66	B	256	BA
65	A	257	AB

LZW compression step 2

- Chuỗi xuất ra sẽ là “AB” bỏ đi ký tự cuối là “B” nên “A” sẽ được xuất ra, “B” sẽ được tiếp tục đi xét.

Bước 3:

- “B” + ký tự tiếp theo của chuỗi đầu vào là “A” tạo thành “BA” dã có trong từ điển nên tiếp tục xét.
- “BA” + ký tự tiếp theo của chuỗi đầu vào là “A” tạo thành “BAA” chưa có trong từ điển nên mã mới được tạo.

BABAABAAA

P=A
C = empty

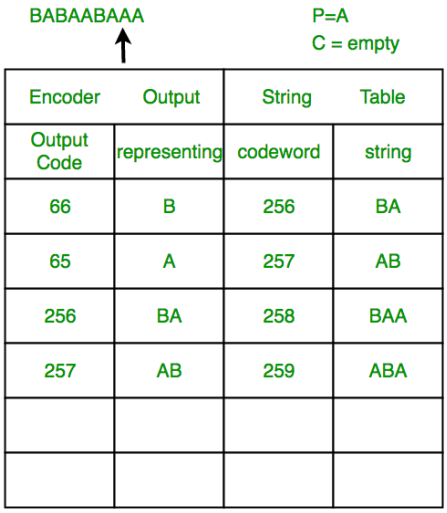
Encoder	Output	String	Table
Output Code	representing	codeword	string
66	B	256	BA
65	A	257	AB
256	BA	258	BAA

LZW compression step 3

- Chuỗi xuất ra sẽ là “BAA” bỏ đi ký tự cuối là “A” nên “BA” sẽ được xuất ra, “A” sẽ được tiếp tục đi xét.

Bước 4:

- “A” + ký tự tiếp theo của chuỗi đầu vào là “B” tạo thành “AB” dã có trong từ điển nên tiếp tục xét.
- “AB” + ký tự tiếp theo của chuỗi đầu vào là “A” tạo thành “ABA” chưa có trong từ điển nên mã mới được tạo



Encoder	Output	String	Table
Output Code	representing	codeword	string
66	B	256	BA
65	A	257	AB
256	BA	258	BAA
257	AB	259	ABA

LZW compression step 4

- Chuỗi xuất ra sẽ là “ABA” bỏ đi ký tự cuối là “A” nên “AB” sẽ được xuất ra, “A” sẽ được tiếp tục đi xét.

Bước 5:

- “A” + ký tự tiếp theo của chuỗi đầu vào là “A” tạo thành “AA” chưa có trong từ điển nên mã mới được tạo.

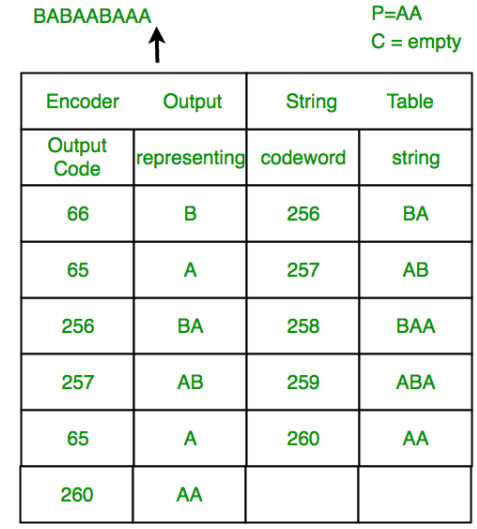
Encoder	Output	String	Table
Output Code	representing	codeword	string
66	B	256	BA
65	A	257	AB
256	BA	258	BAA
257	AB	259	ABA
65	A	260	AA

LZW compression step 5

- Chuỗi xuất ra sẽ là “AA” bỏ đi ký tự cuối là “A” nên “A” sẽ được xuất ra, “A” sẽ được tiếp tục đi xét.

Bước 6:

- “A” + ký tự tiếp theo của chuỗi đầu vào là “A” tạo thành “AA” dã có trong từ điển và là ký tự cuối cùng của chuỗi đầu vào nên sẽ output ra mã có trong từ điển.



Encoder	Output	String	Table
Output Code	representing	codeword	string
66	B	256	BA
65	A	257	AB
256	BA	258	BAA
257	AB	259	ABA
65	A	260	AA
260	AA		

LZW compression step 6

Output sau khi thực hiện mã hóa bằng LZW: 66, 65, 256, 257, 65, 260

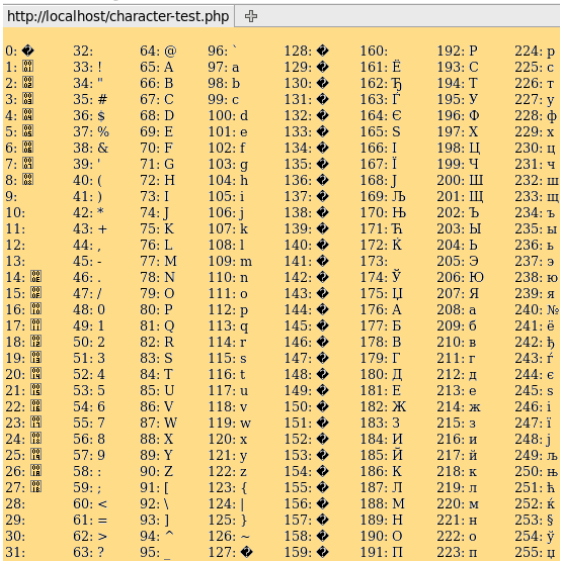
3.2) Thuật toán giải nén

Các bước thực hiện của thuật toán giải nén LZW:

1. Mã giá trị đầu tiên của chuỗi đầu vào luôn luôn có trong bộ từ điển ban đầu nên giá trị của mã sẽ được xuất ra output.
2. Lần lượt xét từng mã giá trị tiếp theo của chuỗi mã hóa đầu vào.
3. Nếu mã giá trị đang xét có trong từ điển thì từ mới được tạo bằng chuỗi giá trị phía trước + ký tự đầu tiên của mã đang xét và xuất ra output giá trị của mã đang xét.
4. Nếu mã giá trị đang xét chưa có trong từ điển thì từ mới được tạo bằng ký tự đầu tiên của chuỗi giá trị phía trước + chuỗi giá trị phía trước và xuất ra output.
5. Quay lại bước 2 đến hết chuỗi mã hóa đầu vào.

Ví dụ minh họa:

Cho bộ từ điển ban đầu có giá trị từ 0 – 255 như sau:



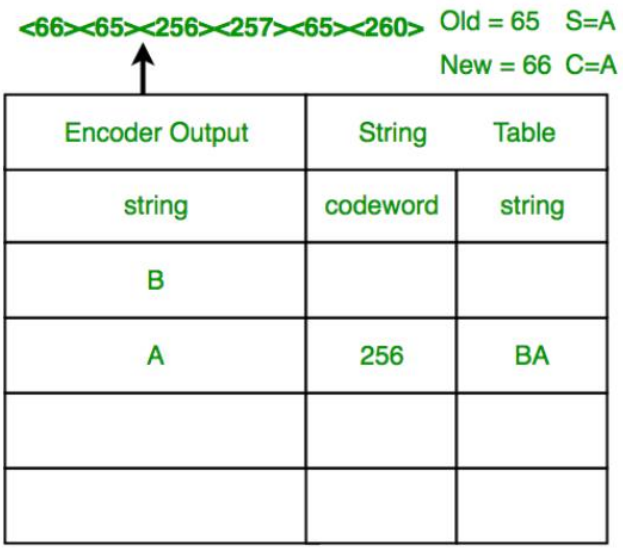
Hãy giải nén chuỗi : “66, 65, 256, 257, 65, 260” bằng thuật toán LZW

Bước 1:

- Mã giá trị “66” tương ứng với ký tự “B” nên được xuất ra output: “B”

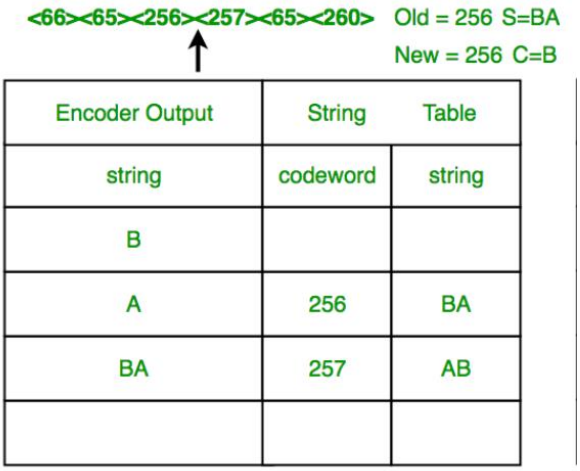
Bước 2:

- Mã giá trị “65” tương ứng với ký tự “A” có trong từ điển nên được xuất ra output: “A”
- Chuỗi ký tự mới: tạo bằng giá trị chuỗi phía trước “B” + ký tự đầu tiên của chuỗi đang xét “A” là “A” thành “BA” được thêm vào từ điển với mã 256.



Bước 3:

- Mã giá trị “256” tương ứng với ký tự “BA” có trong từ điển nên được xuất ra output: “BA”
- Chuỗi ký tự mới: tạo bằng giá trị chuỗi phía trước “A” + ký tự đầu tiên của chuỗi đang xét “BA” là “B” thành “AB” được thêm vào từ điển với mã 257.



Encoder Output	String	Table
string	codeword	string
B		
A	256	BA
BA	257	AB

Bước 4:

- Mã giá trị “257” tương ứng với ký tự “AB” có trong từ điển nên được xuất ra output: “AB”
- Chuỗi ký tự mới: tạo bằng giá trị chuỗi phía trước “BA” + ký tự đầu tiên của chuỗi đang xét “AB” là “A” thành “BAA” được thêm vào từ điển với mã 258.

Encoder Output	String	Table
string	codeword	string
B		
A	256	BA
BA	257	AB
AB	258	BAA

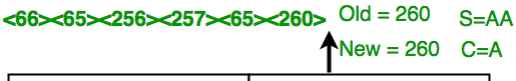
Bước 5:

- Mã giá trị “65” tương ứng với ký tự “A” có trong từ điển nên được xuất ra output: “A”
- Chuỗi ký tự mới: tạo bằng giá trị chuỗi phía trước “AB” + ký tự đầu tiên của chuỗi đang xét “A” là “A” thành “ABA” được thêm vào từ điển với mã 259.

Encoder Output	String	Table
string	codeword	string
B		
A	256	BA
BA	257	AB
AB	258	BAA
A	259	ABA

Bước 6:

- Mã giá trị “260” chưa có trong từ điển nên mã mới được tạo bằng giá trị chuỗi phía trước “A” + ký tự đầu tiên chuỗi phía trước “A” là “A” thành “AA” được thêm vào từ điển với mã 260 và xuất ra output.



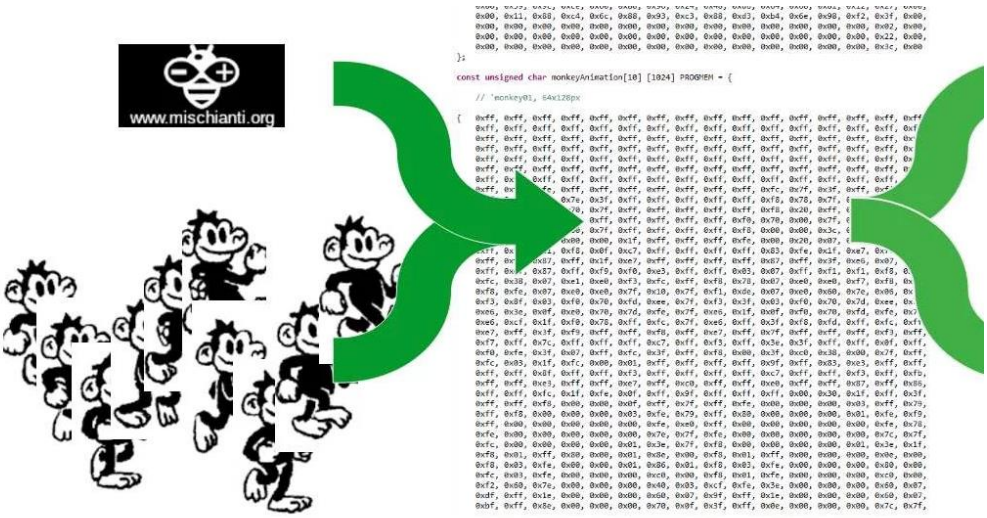
Encoder Output	String	Table
string	codeword	string
B		
A	256	BA
BA	257	AB
AB	258	BAA
A	259	ABA
AA	260	AA

Chuỗi sau khi được giải nén: “BABAABAAA” tương ứng với ví dụ mã hóa phía trên.

4) Nén dữ liệu hình ảnh

4.1) Phương pháp 1

Hình ảnh được lưu trữ dưới dạng một chuỗi các bytes kết hợp với nhau, do đó ta có thể sử dụng hàm của thư viện PIL (Python) để chuyển ảnh về dạng này và áp dụng thuật toán LZW tương tự cho text.



Bộ từ điển ban đầu tương tự như dạng text:

http://localhost/character-test.php

0:	32:	64: @	96: `	128:	160:	192: Р	224: р
1:	33: !	65: A	97: a	129:	161: Ё	193: С	225: с
2:	34: "	66: B	98: b	130:	162: Ђ	194: Т	226: т
3:	35: #	67: C	99: c	131:	163: Ґ	195: У	227: у
4:	36: $	68: D	100: d	132:	164: Є	196: Ф	228: ф
5:	37: %	69: E	101: e	133:	165: S	197: Х	229: х
6:	38: &	70: F	102: f	134:	166: I	198: Ц	230: ц
7:	39: '	71: G	103: g	135:	167: Ї	199: Ч	231: ч
8:	40: (	72: H	104: h	136:	168: J	200: Ш	232: ш
9:	41:)	73: I	105: i	137:	169: Љ	201: Щ	233: щ
10:	42: *	74: J	106: j	138:	170: Њ	202: Ъ	234: ъ
11:	43: +	75: K	107: k	139:	171: Ћ	203: Ы	235: ы
12:	44: ,	76: L	108: l	140:	172: Ќ	204: Ь	236: ь
13:	45: -	77: M	109: m	141:	173:	205: Э	237: э
14:	46: .	78: N	110: n	142:	174: Ў	206: Ю	238: ю
15:	47: /	79: O	111: o	143:	175: Џ	207: Я	239: я
16:	48: 0	80: P	112: p	144:	176: A	208: а	240: №
17:	49: 1	81: Q	113: q	145:	177: Б	209: б	241: ё
18:	50: 2	82: R	114: r	146:	178: В	210: в	242: ђ
19:	51: 3	83: S	115: s	147:	179: Г	211: г	243: ґ
20:	52: 4	84: T	116: t	148:	180: Д	212: д	244: є
21:	53: 5	85: U	117: u	149:	181: Е	213: е	245: s
22:	54: 6	86: V	118: v	150:	182: Ж	214: ж	246: i
23:	55: 7	87: W	119: w	151:	183: З	215: з	247: ї
24:	56: 8	88: X	120: x	152:	184: И	216: и	248: j
25:	57: 9	89: Y	121: y	153:	185: Й	217: й	249: љ
26:	58: :	90: Z	122: z	154:	186: К	218: к	250: њ
27:	59: ;	91: [	123: {	155:	187: Л	219: л	251: ћ
28:	60: <	92: \	124: |	156:	188: М	220: м	252: ќ
29:	61: =	93:]	125: }	157:	189: Н	221: н	253: §
30:	62: >	94: ^	126: ~	158:	190: О	222: о	254: ў
31:	63: ?	95: _	127:	159:	191: П	223: п	255: џ

4.2) Phương pháp 2

Tận dụng đặc điểm của ảnh màu: mỗi ảnh màu là sự kết hợp của 3 kênh màu riêng biệt (Red, Green, Blue), với mỗi kênh màu là một ma trận kích thước (chiều cao x chiều rộng của ảnh). Tại mỗi vị trị trong ma trận này chứa 1 giá trị pixel (giá trị của điểm ảnh) và giá trị của nó nằm trong khoảng từ 0 đến 255

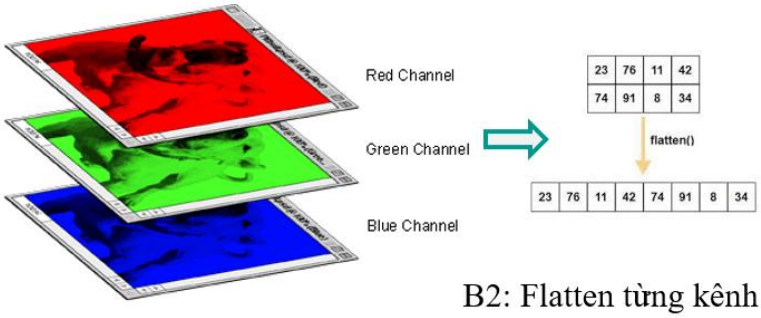
Nếu ta sử dụng phương pháp này thì sẽ giúp tối ưu cho việc nén giữ liệu bằng LZW vì :

+ Các kênh màu trong mô hình RGB tạo ra một loạt các giá trị màu từ 0 đến 255. Bằng cách chia thành 3 kênh màu riêng biệt, mỗi kênh chỉ chứa thông tin về một màu cụ thể, điều này cho phép chúng ta áp dụng các kỹ thuật nén dữ liệu độc lập lên từng kênh màu.

+ Khi áp dụng LZW lên từng kênh màu, ta có thể tận dụng tính chất thống kê của dữ liệu màu trong kênh đó và xây dựng bảng mã LZW tối ưu cho từng kênh riêng biệt. Điều này cho phép ta mã hóa các giá trị màu xuất hiện thường xuyên trong mỗi kênh màu bằng các mã ngắn hơn, trong khi các giá trị màu xuất hiện ít thường xuyên sẽ được mã hóa bằng các mã dài hơn. Quá trình này giúp giảm kích thước tập tin ảnh màu mà vẫn giữ lại chất lượng hình ảnh đáng kể.

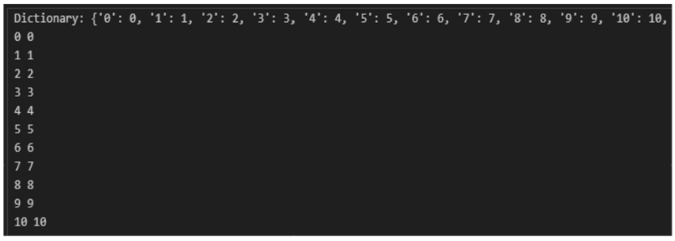
-> Có thể nén ảnh màu bằng LZW bằng cách chia thành 3 kênh và nén trên từng kênh

B1: Tách ảnh thành 3 kênh R, G, B



B2: Flatten từng kênh (R)

Do giá trị điểm ảnh (pixel) chỉ có giá trị từ 0 – 255 nên sẽ được bộ từ điển mới tối ưu hơn:



Bộ từ điển ban đầu
Bao gồm 256 giá trị màu tương ứng với mã:
Vd: ‘0’:0, ‘1’:1

5) Lưu trữ file nén

Ta có kết quả LZW encode cho chuỗi “BABAABAAA”: [66, 65, 256, 257, 65, 260] vì vậy mảng các mã kí tự [66, 65, 256, 257, 65, 260] sẽ được dùng để lưu trữ.

Phương pháp 1: Sử dụng hàm numpy.save thành file như sau: ‘encode.pny’

+ Ưu:

- Đơn giản, dễ thực hiện

- Nhanh

+ Nhược:

- Kích thước file nén lớn so với file gốc

- Tốn nhiều chi phí với chuỗi encode lớn

Phương pháp 2:

- B1: Encode sang dạng unsigned short C ++ (2 byte)
- B2: Lưu file dạng ‘encode.bin’

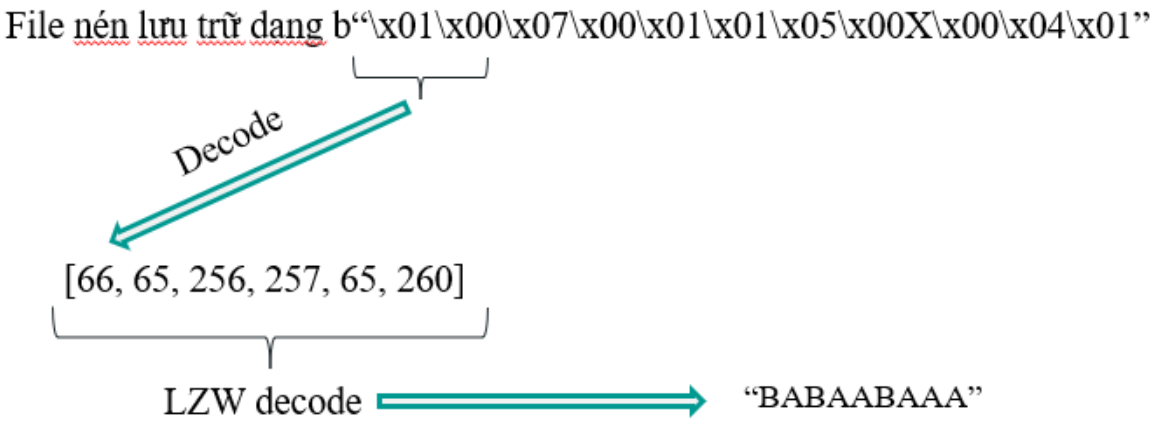
+ Ưu:

- Kích thước file nén nhỏ hơn file gốc

+ Nhược:

- Giá trị max của unsigned short là 65535 -> hạn chế với file gốc lớn

Trong đồ án lần này, phương pháp 2 được sử dụng để lưu trữ file nén.



- Do trong quá trình nén, có sử dụng thêm phần encode sang dạng unsigned short C ++ (2 byte) nên khi giải nén, cần có thêm bước decode về dạng số, sau đó tiếp tục sử dụng LZW decode để giải mã.

File nén thuật toán LZW sẽ gồm 2 thứ:

+ File bin: chứa các thông tin của input sau khi qua thuật toán LZW nén thành các bit.

+ File .csv: đây chính là file từ điển chứa các thông tin giúp cho quá trình kiểm tra, theo dõi từ điển do LZW không cần từ điển này trong quá trình giải mã.

6) Hàm đánh giá

6.1) Quá trình nén

- Average Length (Độ dài trung bình) để đo lường độ dài trung bình của mã bit được sử dụng để biểu diễn mỗi ký tự sau khi áp dụng thuật toán. Average length càng nhỏ thì thuật toán nén càng tối ưu. Khi average length nhỏ, có nghĩa là số lượng bit cần thiết để biểu diễn mỗi ký tự trong tập dữ liệu sau khi áp dụng thuật toán

nén càng ít. Điều này có nghĩa là thuật toán nén giảm kích thước tệp tin gốc một cách hiệu quả. (đơn vị bits/symbol)

Công thức tính:

Average Length = $\sum P(x) * l(x)$

Trong đó: P(x) là xác suất xuất hiện của ký tự x trong tập dữ liệu

l(x) là độ dài của mã bit biểu diễn ký tự x.

- Compression ratio (tỷ lệ nén) là một độ đo quan trọng để đánh giá hiệu quả của thuật toán nén dữ liệu. Nó đo lường mức độ nén được đạt được bằng cách so sánh kích thước tệp tin gốc và kích thước tệp tin sau khi áp dụng thuật toán nén.

Công thức tính:

Compression ratio = (Kích thước tệp tin gốc) / (Kích thước tệp tin sau nén)

Ví dụ: Một ma trận tương ứng với một kênh màu của ảnh:

1	7	7
7	5	88
88	88	88
88	88	9

Sau khi flatten ảnh thành một chuỗi các pixel, tiến hành mã hóa LZW trên chuỗi này.

Kết quả chuỗi mã hóa

Kích thước ảnh trước khi nén: 8 * 3 * 4 = 96 bits

Kích thước ảnh sau khi nén: 8 * 5 + 9 * 3 = 67 bits

(Do giá trị từ 256 – 1023 sử dụng 9 bits để lưu một ký tự, tương tự giá trị từ 1024 – 2047 dùng 10 bits để lưu một ký tự)

Tỉ số nén: Cr = 96/67 ≈ 1,43

Nếu giá trị của compression ratio > 1 thì thể hiện rằng kích thước của tệp tin nén nhỏ hơn tệp gốc tỉ lệ này càng cao thì file nén có kích thước càng nhỏ. Nếu compression ratio < 1 thì nói lên rằng kích thước của tệp tin nén lớn hơn tệp tin gốc -> thuật toán nén không được hiệu quả.

- Tuy nhiên để xác định kích thước tệp tin gốc của bức ảnh là hơi khó vì các bức ảnh chủ yếu được lưu dưới dạng png và jpeg. Thế nhưng các loại tập tin này lại chính là file nén thế nhưng vẫn có cách để tính được bằng cách nhân kích thước ảnh với độ sâu bit và số kênh màu(đơn vị là bit)

Dung lượng (byte) = (Chiều rộng) x (Chiều cao) x (Độ sâu bit) x (Số kênh màu)

- Execution time (Thời gian thực thi) là thời gian mà thuật toán nén dữ liệu mất để thực hiện quá trình nén và giải nén trên tệp tin. Thời gian thực thi càng ngắn thì tốc độ nén hoặc giải nén của thuật toán nén càng nhanh và ngược lại

6.2) Quá trình giải nén

+ Đối với file dạng văn bản:

Bước 1: Thử đọc file gốc. Nếu thành công, tiếp tục sang bước 2. Nếu không, trả về giá trị -1 để biểu thị lỗi đọc file.

Bước 2: Duyệt qua từng ký tự trong file gốc và file được giải nén, và kiểm tra xem hai ký tự tương ứng có khác nhau hay

không. Nếu hai ký tự khác nhau, tăng giá trị biến "different" lên một đơn vị.

Bước 3: Trả về giá trị biến "different", đại diện cho số lượng ký tự khác nhau giữa hai file.

+ Đối với file hình ảnh:

Bước 1: Thử đọc file gốc. Nếu thành công, tiếp tục sang bước 2. Nếu không, trả về giá trị -1 để biểu thị lỗi đọc file.

Bước 2: Kiểm tra xem file ảnh giải nén và file gốc có cùng loại ảnh hay không. Nếu file ảnh giải nén bằng phương pháp đầu tiên, ta sẽ chuyển ảnh về dạng chuỗi các bits.

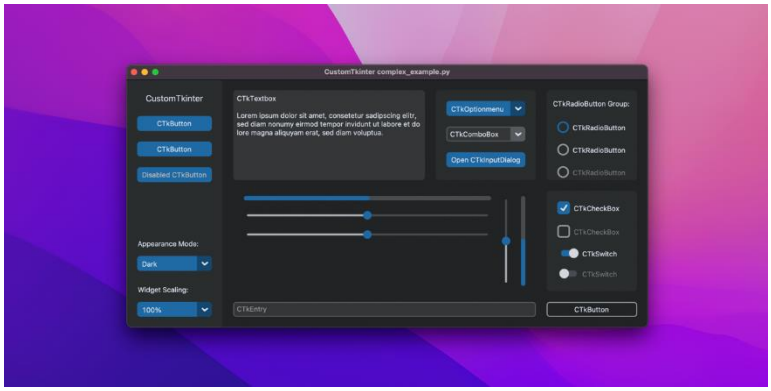
Bước 3: Duyệt qua từng pixel của ảnh gốc và ảnh giải nén, và kiểm tra xem hai giá trị pixel tương ứng có khác nhau hay không. Nếu hai giá trị pixel khác nhau, tăng giá trị biến "different" lên một đơn vị. Nếu sử dụng phương pháp nén ảnh đầu tiên, kiểm tra xem hai ký tự tương ứng có khác nhau hay không. Nếu hai ký tự khác nhau, tăng giá trị biến "different" lên một đơn vị.

Bước 4: Trả về giá trị biến "different", đại diện cho số lượng pixel khác nhau giữa hai ảnh.

- Quá trình này sẽ giúp đánh giá chất lượng kết quả nén bằng cách đếm số lượng ký tự hoặc pixel khác nhau giữa file gốc và file giải nén. Kết quả càng gần 0, tức là ít khác biệt, thì thuật toán nén càng hiệu quả.

IV) Thiết kế hệ thống

1) Tkinter



Tkinter là thư viện GUI tiêu chuẩn cho Python. Tkinter trong Python cung cấp một cách nhanh chóng và dễ dàng để tạo các ứng dụng GUI. Tkinter cung cấp giao diện hướng đối tượng cho bộ công cụ Tk GUI được cài đặt đi kèm với Python.

+ Ưu:

- Có nhiều widget khác nhau như button, canvas, checkbutton, entry, ... chúng được sử dụng để xây dựng các ứng dụng GUI .

- Nhanh chóng, thuận tiện cho quá trình deploy ứng dụng đơn giản với Python.

2) Pyinstaller

- PyInstaller – một công cụ đóng gói chương trình Python thành “stand-alone executable file” có thể thực thi trong các môi trường Windows, Linux, MAC OS, v.v.. mà không cần cài đặt Python Interpreter

- Pyinstaller cung cấp khả năng tạo một thư mục hoặc tệp thực thi mà người dùng có thể chạy ngay lập tức mà không cần cài đặt thêm.

+ Ưu:

- Câu lệnh đơn giản cho quá trình đóng gói ứng dụng
- Kết hợp được với Tkinter
- Dễ dàng tùy chỉnh các thuộc tính của ứng dụng

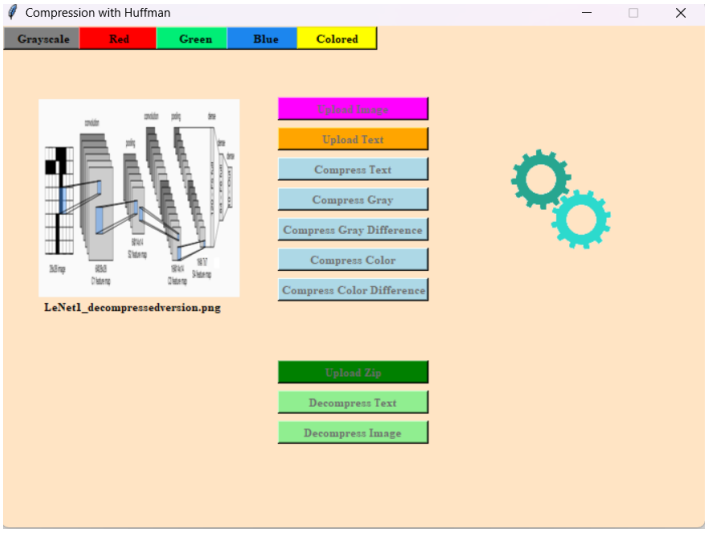
V) Demo

1) Huffman

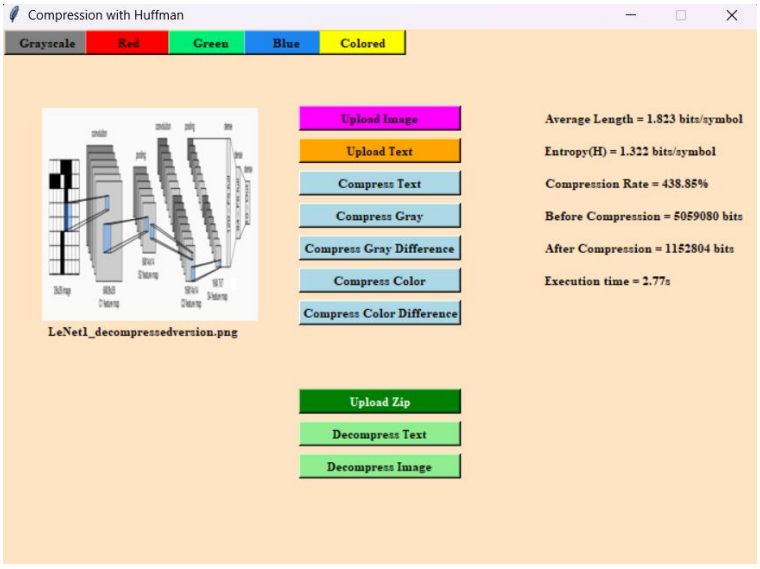
Giới thiệu sơ lược về hệ thống

- Giao diện khởi động:

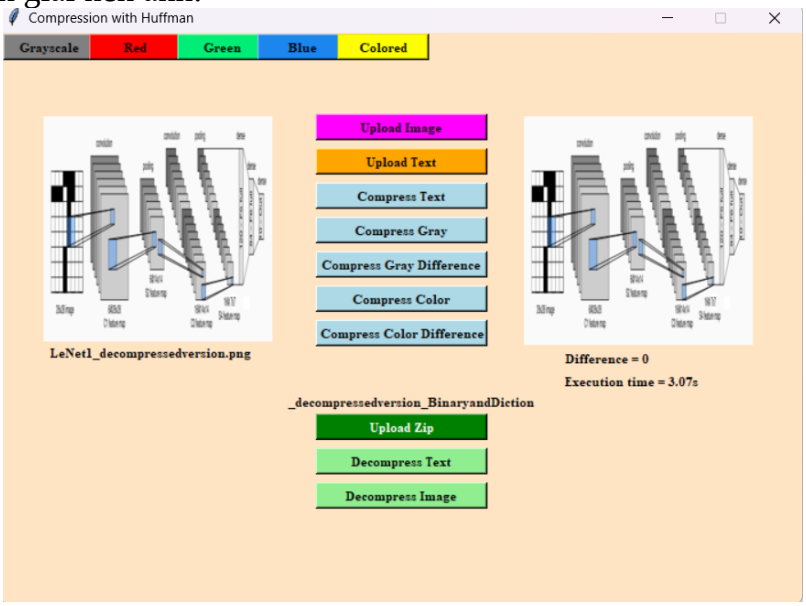
- Quá trình xử lý:



- Thực hiện nén ảnh:

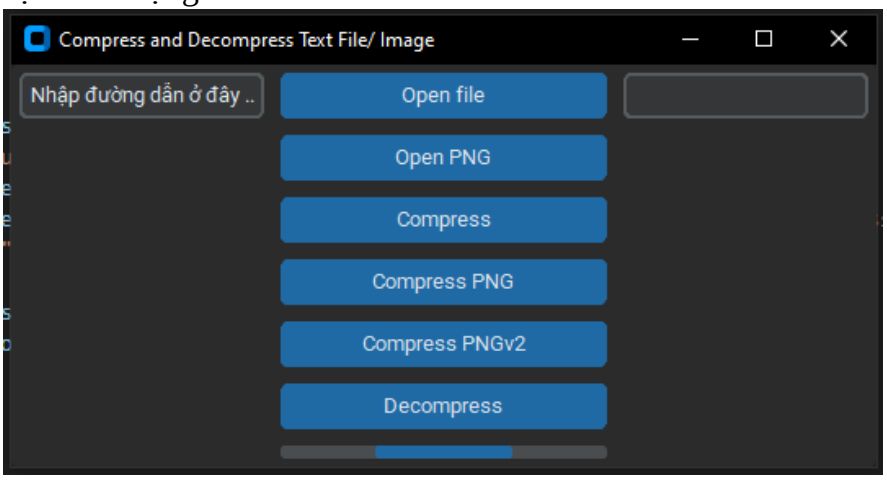


- Thực hiện giải nén ảnh:

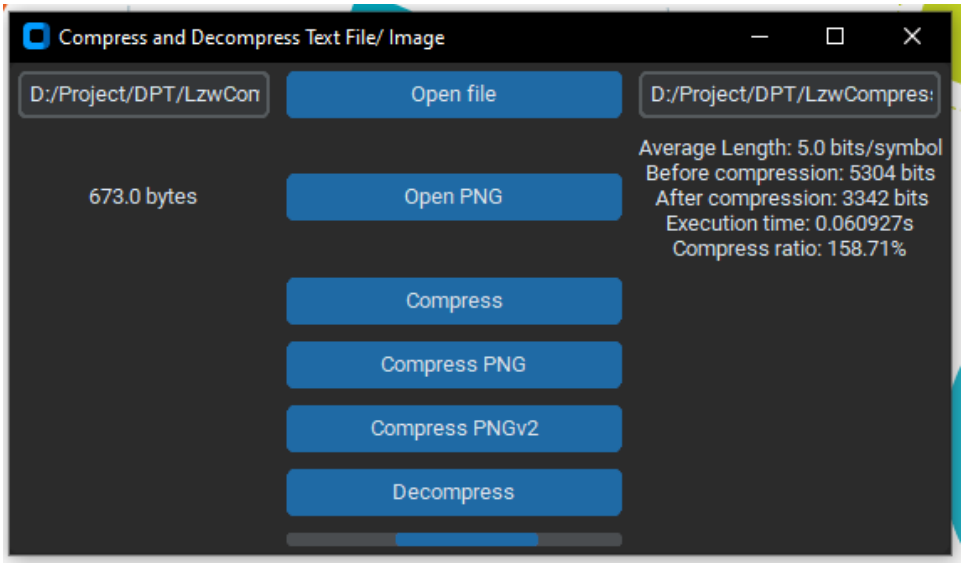


2) LZW

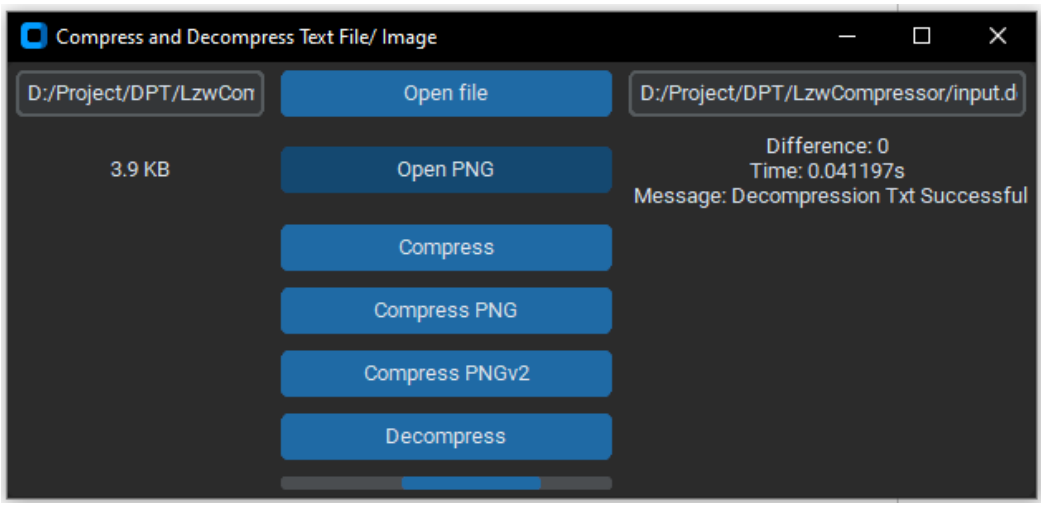
Giao diện khởi động:



Thực hiện nén:



Thực hiện giải nén:



VI) Kết luận

	Huffman	LZW
Hiệu suất nén	Thuật toán Huffman có thể đạt được mức nén tốt cho các tập dữ liệu có tần số xuất hiện của ký tự không đồng đều. Tuy nhiên, nó không hiệu quả cho việc nén chuỗi các ký tự lặp lại.	Thuật toán nén LZW có ưu điểm là hệ số nén tương đối cao, trong tập tin nén không cần phải chứa bảng mã, thích hợp cho chuỗi có ký tự lặp. Tuy nhiên, nó là tốn nhiều bộ nhớ, khó thực hiện dựa trên các mảng đơn giản (bé hơn 64KB)
Độ phức tạp	Thuật toán Huffman có độ phức tạp thời gian O(nlogn) trong việc xây dựng cây Huffman và O(n) trong việc nén và giải nén dữ liệu, trong đó n là số lượng ký tự trong tập dữ liệu.	Thuật toán LZW có độ phức tạp thời gian O(n) trong việc nén và cả giải nén dữ liệu, trong đó n là số lượng ký tự trong tập dữ liệu.
Ứng dụng	Huffman thường được sử dụng cho việc nén dữ liệu nhỏ và không có sự thay đổi thường xuyên. Ví dụ, nó phổ biến trong nén dữ liệu văn bản hoặc dữ liệu hình ảnh tĩnh.	LZW đã được sử dụng trong phần mềm nén mã nguồn mở, LZW đã trở nên phổ biến khi nó được sử dụng làm 1 phần của file GIF năm 1987. Nó cũng có thể được sử dụng trong TIFF và PDF file.

VII) Phân chia công việc

	Vũ Dương	Quốc Khánh
Thuật toán Huffman	100%	
Thuật toán LZW		100%
Slide, báo cáo	50%	50%

VIII) TÀI LIỆU THAM KHẢO

[1] Huffman coding: https://en.wikipedia.org/wiki/Huffman_coding

[2] Thuật toán nén Huffman Coding:
https://chidokun.github.io/2021/07/huffman-coding-p1/

[3] Huffman coding:
https://people.eng.unimelb.edu.au/ammoffat/abstracts/compsurv19moffat.pdf

[4] LZW (Lempel–Ziv–Welch) Compression technique:
https://www.geeksforgeeks.org/lzw-lempel-ziv-welch-compression-technique/

[5] LZW: https://vi.wikipedia.org/wiki/LZW

[6] Nén LZW: https://medium.com/@kongminh/n%C3%A9n-lzw-813b7e4cd7eb