

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO CUỐI KÌ**

**MÔN HỌC: TÍNH TOÁN ĐA PHƯƠNG TIỆN**

**ĐỀ TÀI: TEXT COMPRESSION USE HUFFMAN AND LEMPEL-ZIV-WELCH**

|  |  |  |
| --- | --- | --- |
| Giảng viên hướng dẫn | : | Gv. Mai Xuân Hùng |
| Sinh viên thực hiện | **:** | **Đỗ Trọng Khánh – 19521676**  **Võ Phạm Duy Đức – 19521383**  **Trịnh Công Danh – 19521326** |
| Lớp | **:** | **CS232.M21.KHCL** |

**Thành phố Hồ Chí Minh, ngày 12 tháng 06 năm 2022**.

MỤC LỤC

[Chương 1: Tổng quan - 3 -](#_Toc106917919)

[1. Đặt vấn đề - 3 -](#_Toc106917920)

[2. Thách thức bài toán - 3 -](#_Toc106917921)

[3. Mục tiêu và phạm vi - 3 -](#_Toc106917922)

[Chương 2: Các nghiên cứu và hệ thống liên quan - 4 -](#_Toc106917923)

[1. Thuật toán Huffman Coding - 4 -](#_Toc106917924)

[2. Thuật toán LZW - 4 -](#_Toc106917925)

[Chương 3: Nội dung chính - 4 -](#_Toc106917926)

[1. Thuật toán Huffman Coding - 4 -](#_Toc106917927)

[1.1. Giới thiệu: - 4 -](#_Toc106917928)

[1.2. Nguyên lý - 4 -](#_Toc106917929)

[1.3. Thuật toán - 6 -](#_Toc106917930)

[1.4. Một số những hạn chế của mã Hufman - 11 -](#_Toc106917931)

[2. Thuật toán Lempel-Ziv-Welch (LZW) - 11 -](#_Toc106917932)

[2.1. Giới thiệu - 11 -](#_Toc106917933)

[2.2. Nguyên lí - 11 -](#_Toc106917934)

[2.3. Thuật toán - 12 -](#_Toc106917935)

[2.4. Ưu điểm, nhược điểm của phương pháp mã hoá LZW - 16 -](#_Toc106917936)

[Chương 4: Thực nghiệm và đánh giá - 17 -](#_Toc106917937)

[1. Huffman Coding - 17 -](#_Toc106917938)

[2. Lempel-Ziv-Welch (LZW) - 20 -](#_Toc106917939)

[3. Demo - 21 -](#_Toc106917940)

[Chương 5: Kết luận: - 22 -](#_Toc106917941)

[Chương 6: Tài liệu tham khảo - 23 -](#_Toc106917942)

[1. Thuật toán Huffman: - 23 -](#_Toc106917943)

[2. Thuật toán Lempel-Ziv-Wel: - 23 -](#_Toc106917944)

# Chương 1: Tổng quan

## 1. Đặt vấn đề

* Trong các lĩnh vực của công nghệ thông tin –viễn thông hiện nay, việc truyền tải tin tức đã là một công việc xảy ra thường xuyên. Do vậy, trong đề tài đồ án môn học này chúng tôi xây dựng một ứng dụng web có khả năng nén các text thành các dạng như bit, thập phân và 2 thuật toán chúng tôi sử dụng để nén đó là LZW (Lempel-Ziv-Welch) và Huffman.
* Lợi ích của việc nén text là: làm giảm lượng bộ nhớ để lưu trữ một văn bản, làm giảm thời gian truyền văn bản qua mạng máy tính và gần đây đã được triển khai để tăng tốc độ tính toán thuật toán vì chúng có thể tốt hơn là khai thác hệ thống phân cấp bộ nhớ có sẵn trong các PC hiện đại bằng cách giảm thời gian truy cập đĩa, bằng cách tăng gần như băng thông và kích thước của đĩa (hoặc bộ nhớ, bộ nhớ cache) và bằng cách chi phí không đáng kể vì tốc độ đáng kể của CPU hiện tại.
* Input: Một đoạn text được người dùng nhập
* Output: Ứng dụng sẽ thực hiện nén đoạn text thành dạng nhị phân (Huffman) hoặc thập phân (LZW).

## 2. Thách thức bài toán

* Một trong những khó khăn của bài toán này là khi sử dụng thuật toán LZW thì sẽ không xử lí được với dữ liệu text với ngôn ngữ tiếng việt.

## 3. Mục tiêu và phạm vi

* Mục tiêu chung của bài báo cáo là xây dựng được một ứng dụng web có khả năng nén văn bản thành các dạng có kích thước nhẹ hơn như nhị phân, thập phân với độ chính xác cao và tốc độ xử lí nhanh chóng.
* Trong nội dung này chúng tôi giới hạn phạm vi của ứng dụng về các nội dung sau: Dữ liệu chúng tôi sử dụng trong ứng dụng này là dữ liệu dạng text.

# Chương 2: Các nghiên cứu và hệ thống liên quan

## 1. Thuật toán Huffman Coding

* Link bài báo: <https://pdfcoffee.com/bao-cao-huffman-coding-pdf-free.html>
* Bài báo này được thực hiện bởi 1 nhóm sinh viên gồm: Nguyễn Trung Dũng, Phạm Minh Trị, Phạm Ngọc Minh Tân.
* Đánh giá chi phí giải thuật:
* Ta ký hiệu T(n) là hàm thời gian thực hiện một chương trình, trong đó n là kích thước độ lớn của dữ liệu vào, và T(n)≥ 0 , Với n ≥ 0.
* Chi phí cho giải thuật xử lí Heap tốn **T(n)=O(logn)**.
* Chi phí cho giải thuật tạo Heap tốn **T(n)=O(n/2).**
* Chi phí cho giải thuật tạo cây **T(n)=nlogn**.

## 2. Thuật toán LZW

# Chương 3: Nội dung chính

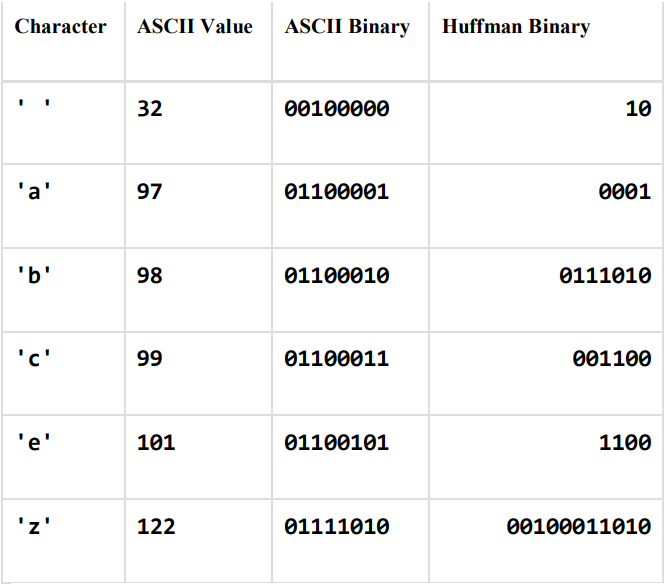
## 1. Thuật toán Huffman Coding

### 1.1. Giới thiệu:

* Phương pháp Huffman là một thuật toán do David A. Huffman của MIT đưa ra vào năm 1952 để nén dữ liệu dạng văn bản, 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. Mặc dù là một thuật toán nén tương đối đơn giản, nhưng Huffman đủ mạnh để các biến thể của nó vẫn được sử dụng ngày nay trong mạng máy tính, máy fax, modem, HDTV và các lĩnh vực khác.

### 1.2. Nguyên lý

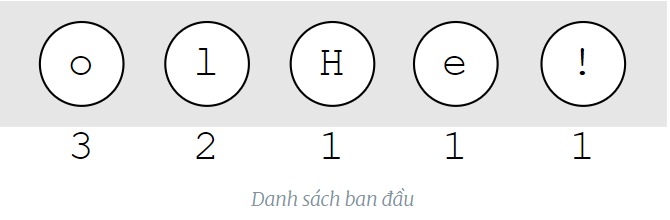
* Nguyên lý của phương pháp Huffman là mã hóa các bytes trong tệp dữ liệu nguồn bằng biến nhị phân. Nó tạo mã độ dài biến thiên là một tập hợp các bits. Đây là phương pháp nén kiểu thống kê, những ký tự xuất hiện nhiều hơn sẽ có mã ngắn hơn.
* Thông thường, dữ liệu văn bản được lưu trữ ở định dạng chuẩn 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.
* Bảng dưới đây so sánh các giá trị ASCII của các ký tự khác nhau với các bảng mã Huffman có thể có cho một số văn bản tiếng Anh. Các ký tự thông thường như dấu cách và 'e' có mã hóa ngắn, trong khi các ký tự hiếm hơn (như 'z') có mã hóa dài hơn.



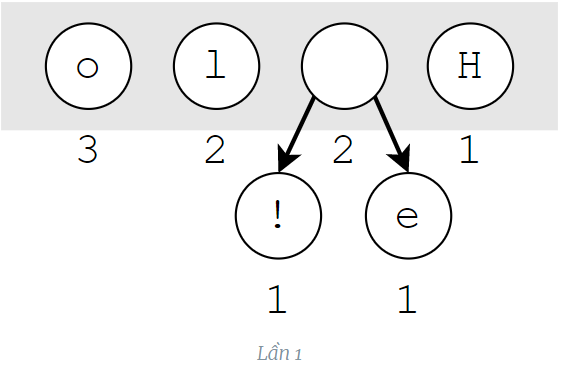
### 1.3. Thuật toán

#### 1.3.1. Thuật toán nén

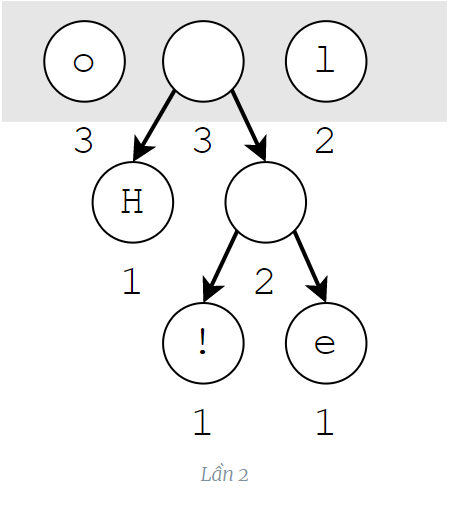
* Với ý tưởng trên, thuật toán Huffman coding gồm 3 bước:
* Bước 1: Đếm tần suất xuất hiện của các phần tử trong chuỗi đầu vào.
* Bước 2: Xây dựng cây Huffman (cây nhị phân mã hóa) với quy ước bên trái mã 0, bên phải mã 1.
* Bước 3: Từ cây Huffman, ta có được các giá trị mã hóa. Lúc này, ta có thể xây dựng chuỗi mã hóa từ các giá trị này.
* Quá trình xây dựng cây Huffman gồm các bước sau:
* 2.1. Tạo danh sách chứa các nút lá bao gồm phần tử đầu vào và trọng số nút là tần suất xuất hiện của nó.
* 2.2. Từ danh sách này, lấy ra 2 phần tử có tần suất xuất hiện ít nhất. Sau đó gắn 2 nút vừa lấy ra vào một nút gốc mới với trọng số là tổng của 2 trọng số ở nút vừa lấy ra để tạo thành một cây.
* 2.3. Đẩy cây mới vào lại danh sách.
* 2.4. Lặp lại bước 2 và 3 cho đến khi danh sách chỉ còn 1 nút gốc duy nhất của cây.
* 2.5. Nút còn lại chính là nút gốc của cây Huffman.
* Để dễ tiếp cận các bước thực hiện xây dựng cây Huffman, ví dụ cho chữ ***“Hellooo!”.***
* Bước 2.1: Sau khi đếm tần suất xuất hiện các phần tử đầu vào. Chúng ta tạo danh sách các nút lá với trọng số là tần suất xuất hiện. Danh sách sẽ có 5 phần tử như bên dưới.



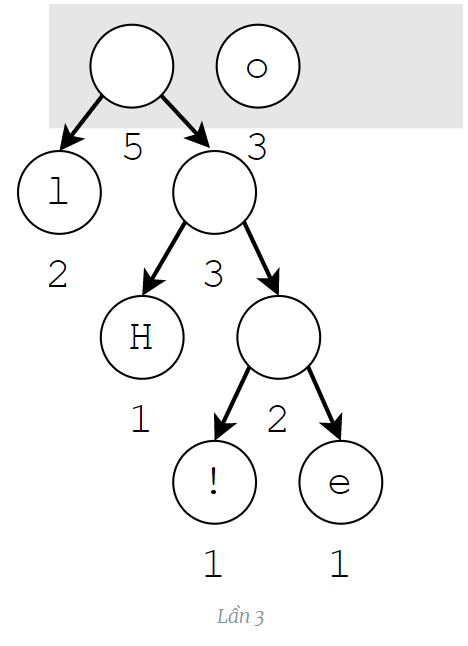
* Bước 2.2 và 2.3: Chọn 2 nút có trọng số thấp nhất, tạo nút gốc mới có trọng số bằng tổng 2 trọng số nút con. Sau đó gắn 2 nút con vào nút gốc và đẩy lại vào danh sách. Danh sách cần được biểu diễn đặc biệt để có thể lấy ra các nút trọng số nhỏ nhất một cách tối ưu nhất.



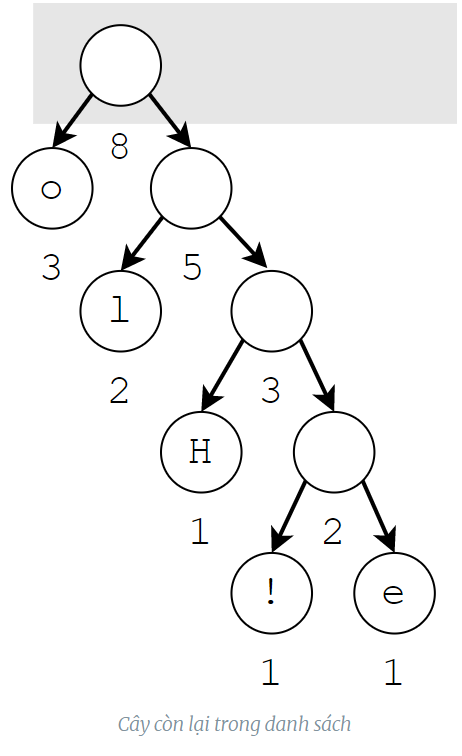
* Bước 2.4: Lặp lại các bước 2.2 và 2.3.



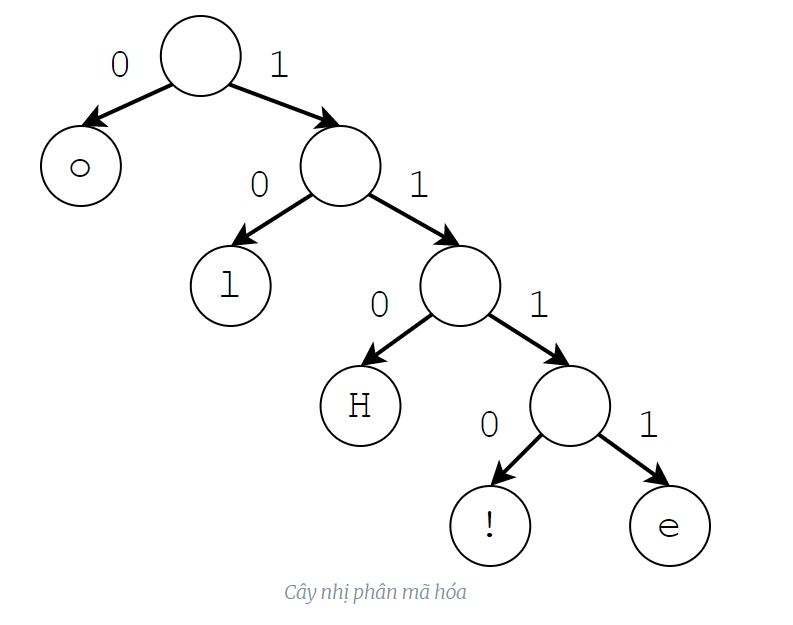
* Bước 2.4: Lặp lại các bước 2.2 và 2.3.



* Bước 2.5: Chỉ còn lại 1 nút trong danh sách, nút này chính là cây Huffman.



* Từ cây Huffman, ta có thể suy ra các giá trị mã hóa của từng phần tử bằng cách duyệt cây nhị phân mã hóa.



#### 1.3.2. Thuật toán giải nén

* Bạn có thể sử dụng cây Huffman để giải mã văn bản đã được mã hóa trước đó bằng các mẫu nhị phân của nó. Thuật toán giải mã là đọc từng bit từ tệp, từng bit một và sử dụng bit này để duyệt qua cây Huffman. Nếu bit là 0, bạn di chuyển sang trái trong cây. Nếu bit là 1, bạn di chuyển sang phải. Bạn làm điều này cho đến khi bạn chạm vào một nút lá. Các nút lá đại diện cho các ký tự, vì vậy khi bạn đến một lá, bạn sẽ xuất ký tự đó. Ví dụ: giả sử chúng ta được cung cấp cùng một cây mã hóa ở trên và chúng ta được yêu cầu giải mã một tệp chứa các bit sau:



* Sử dụng cây Huffman, chúng tôi đi bộ từ gốc cho đến khi tìm thấy các ký tự, sau đó xuất chúng và quay trở lại gốc.
* Chúng tôi đọc 1 (phải), sau đó là 1 (phải). Chúng tôi đạt đến 'b' và đầu ra b. Quay lại gốc. 1110010001001010011.
* Chúng tôi đọc 1 (phải), sau đó là 0 (trái). Chúng tôi đạt đến 'a' và đầu ra a. Quay lại thư gốc. 1110010001001010011
* Chúng ta đọc số 0 (trái), sau đó là 1 (phải), rồi đến 0 (trái). Chúng tôi đạt đến 'c' và đầu ra c. 1110010001001010011
* Chúng tôi đọc 0 (trái), sau đó là 0 (trái). Chúng tôi đạt ' 'và xuất ra một khoảng trắng. 1110010001001010011
* Chúng tôi đọc 1 (phải), sau đó là 0 (trái). Chúng tôi đạt đến 'a' và đầu ra a. 1110010001001010011
* Chúng ta đọc số 0 (trái), sau đó là 1 (phải), rồi đến 0 (trái). Chúng tôi đạt đến 'c' và đầu ra c. 1110010001001010011
* Chúng tôi đọc 1 (phải), sau đó là 0 (trái). Chúng tôi đạt đến 'a' và đầu ra a. 1110010001001010011
* Chúng tôi đọc 0, 1, 1. Đây là mẫu mã EOF của chúng tôi, vì vậy chúng tôi dừng lại. Giải mã tổng thể văn bản là ***"bac aca"***

### 1.4. Một số những hạn chế của mã Hufman

* Mã Huffman chỉ thực hiện được khi biết được tần suất xuất hiện của các ký tự.
* Mã Huffman chỉ giải quyết được độ dư thừa phân bố ký tự.
* Huffman tĩnh đòi hỏi phải xây dựng cây nhị phân sẵn chứa các khả năng. Điều này đòi hỏi thời gian không ít do ta không biết trước kiểu dữ liệu sẽ được thực hiện nén.
* Quá trình giải nén phức tạp do chiều dài mã không biết trước cho đến khi ký tự đầu tiên được tìm ra.

## 2. Thuật toán Lempel-Ziv-Welch (LZW)

### 2.1. Giới thiệu

* LZW là một phương pháp nén được phát minh bởi Lempel - Zip và Welch. 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.
* Giải thuật LZW xây dựng một từ điển lưu các mẫu có tần suất xuất hiện cao trong dữ liệu. Từ điển là tập hợp những cặp từ vựng và nghĩa của nó. Trong đó, từ vựng sẽ là các từ mã được sắp xếp theo thứ tự nhất định. Nghĩa là một chuỗi con trong dữ liệu , từ điển được xây dựng đồng thời với quá trình đọc dữ liệu. sự có mặt của một chuỗi con trong từ điển khẳng định rằng chuỗi đó đã từng xuất hiện trong phần dữ liệu đã đọc. Thuật toán liên tục “tra cứu” và cập nhật từ điển sau mỗi lần đọc một ký tự dữ liệu đầu vào.

### 2.2. Nguyên lí

* Nguyên tắc hoạt động của nó như sau:
* Một xâu kí tự là một tập hợp từ hai kí tự trở lên.
* Nhớ tất cả các xâu kí tự đã gặp và gán cho nó một chỉ mục riêng.
* Nếu lần sau gặp lại xâu kí tự đó, xâu kí tự sẽ được thay thế bằng chỉ mục của nó.
* Phần quan trọng nhất của phương pháp nén này là phải tạo một mảng rất lớn dùng để lưu giữ các xâu kí 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 qui tắc để thực hiên việc nén dữ liệu theo thuật toán LZW là:
* Qui tắc 1: 255 dấu hiệu đầu tiên được dành cho các kí tự đơn.
* Qui tắc 2: Cố gắng so sánh với "từ điển" khi trong bộ đệm chứa đã có nhiều hơn hai kí tự.
* Qui tắc 3: Các kí tự ở đầu vào (Nhận từ tập tin sẽ được nén) được bổ sung vào bộ đệm chứa đến khi chuỗi kí tự trong bộ đệm chứa không có trong "từ điển".
* Qui 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". Kí tự cuối cùng của chuỗi kí tự trong bộ đệm chứa phải ở lại trong bộ đệm chứa để tiếp tục tạo thành chuỗi mới.

### 2.3. Thuật toán

#### 2.3.1. Thuật toán nén

* LZW bắt đầu bởi 1 từ điển 255 kí tự (trong trường hợp sử dụng bảng mã 8 bits) và sử dụng chúng như tập kí tự chuẩn. Sau đó mỗi lần đọc nó đọc 8 bits (ví dụ 't', 'r', ...) và mã hóa thành con số tương ứng với chỉ mục của kí tự đó trong từ điển.
* Mỗi khi LZW đi qua 1 chuỗi con mới (giả sử "tr") thì nó thêm chuỗi con đó vào từ điển.
* Mỗi khi nó đi qua 1 chuỗi con mà nó đã thấy trước đó, nó chỉ đọc thêm 1 kí tự mới nữa và cộng với chuỗi con đã biết để tạo ra 1 chuỗi con mới. Lần tiếp theo LZW bắt gặp một chuỗi con đã có, nó chỉ có việc sử dụng số chỉ mục tương ứng trong từ điển.
* Thường thì người ta sẽ định sẵn số lượng lớn nhất các từ trong từ điển (giả sử 4096), vì thế việc nén LZW không làm tiêu tốn hết toàn bộ bộ nhớ. Vì vậy mã của các chuỗi con trong ví dụ này là 12 bits (2 ^ 12 = 4096). Cần thiết phải lập mã dài hơn số bits của một kí tự (12 vs 8 bits), do đo khi rất nhiều chuỗi con lặp lại sẽ được thay thế bởi một mã duy nhất thì việc nén được thực hiện.
* Ví dụ: Các bước thực hiện mã hóa chuỗi “ABCBCABCABCD” như sau:
* Bước 1: Kí tự thứ nhất ‘A’ được cất vào bộ đệm chứa để tạo nên một chuỗi
* Bước 2: Kí tự thứ hai ‘B’ nối thêm vào sau kí tự ‘A’. Vì trong "từ điển" chưa có chuỗi ‘AB’ nên chuỗi này được thêm vào "từ điển" và được gán chỉ mục là 256 (Vì từ 0 đến 255 được dành riêng cho các kí tự đơn: Qui tắc 1). ‘A’ được gửi ra còn ‘B’ phải ở lại trong bộ đệm chứa.
* Bước 3: Kí tự thứ ba ‘C’ thêm vào sau ‘B’. Chuỗi ‘BC’ chưa có trong "từ điển" nên nó được thêm vào "từ điển" và gán chỉ mục là 257. ‘C’ ở lại trong bộ đệm chứa còn ‘B’ được gửi ra.
* Bước 4: Kí tự thứ tư ‘B’ thêm vào sau ‘C’. Chuỗi ‘CB’ chưa có trong "từ điển" nên nó được thêm vào "từ điển" và gán chỉ mục là 258. ‘B’ ở lại trong bộ đệm chứa còn ‘C’ được gửi ra.
* Bước 5: Kí tự thứ năm ‘C’ thêm vào sau ‘B’. Chuỗi "BC" đã có trong "từ điển" nên không có kí tự nào được gửi ra. "BC" tiếp tục ở lại trong bộ đệm chứa để tạo ra chuỗi mới.
* Bước 6: Kí tự thứ sáu ‘A’ thêm vào sau ‘BC’ để tạo thành chuỗi ‘BCA’, do ‘BCA’ không có trong "từ điển" nên nó được thêm vào "từ điển" và gán chỉ mục là 259 đồng thời chỉ mục 257 được gửi ra thay cho "BC" (Qui tắc 4). ‘A’ tiếp tục ở lại trong bộ đệm chứa để tạo thành chuỗi mới.
* Bước 7: Kí tự thứ bảy ‘B’ thêm vào sau ‘A’. Chuỗi ‘AB’ đã có trong "từ điển" nên không có kí tự nào được gửi ra. ‘AB’ tiếp tục ở lại trong bộ đệm chứa để tạo ra chuỗi mới.
* Bước 8: Kí tự thứ tám ‘C’ thêm vào sau ‘AB’ để tạo thành chuỗi ‘ABC’, do ‘ABC’ không có trong "từ điển" nên nó được thêm vào "từ điển" và gán chỉ mục là 260 đồng thời chỉ mục 256 được gửi ra thay cho "AB" (Qui tắc 4). C tiếp tục ở lại trong bộ đệm chứa để tạo thành chuỗi mới.
* Bước 9: Kí tự thứ chín ‘A’ thêm vào sau ‘C’ để tạo thành chuỗi ‘CA’, do ‘CA’ không có trong "từ điển" nên nó được thêm vào "từ điển" và gán chỉ mục là 261. ‘A’ ở lại trong bộ đệm chứa còn ‘C’ được gửi ra.
* Bước 10: Kí tự thứ mười ‘B’ thêm vào sau ‘A’. Chuỗi ‘AB’ đã có trong "từ điển" nên không có kí tự nào được gửi ra. "AB" tiếp tục ở lại trong bộ đệm chứa để tạo ra chuỗi mới.
* Bước 11: Kí tự thứ mười một ‘C’ thêm vào sau ‘AB’. Chuỗi ‘ABC’ đã có trong "từ điển" nên không có kí tự nào được gửi ra. ‘ABC’ tiếp tục ở lại trong bộ đệm chứa để tạo ra chuỗi mới.
* Bước 12: Kí tự thứ mười hai ‘D’ thêm vào sau ‘ABC’ để tạo thành chuỗi ‘ABCD’, do ‘ABCD’ không có trong "từ điển" nên nó được thêm vào "từ điển" và gán chỉ mục là 262 đồng thời chỉ mục 260 được gửi ra thay cho ‘ABC’. ‘D’ tiếp tục ở lại trong bộ đệm chứa để tạo thành chuỗi mới.
* Bước 13: Không còn kí tự nào thêm vào nên kí tự ‘D’ được gửi ra và kết thúc quá trình nén.
* Việc giảm kích thước chỉ thực sự bắt đầu tại bước 6 khi mà một dấu hiệu 12 bits là 257 được gửi ra thay cho hai byte "BC". Các bước được mô tả rõ hơn trong bảng dưới đây:

Text

Description automatically generated with low confidence

#### 2.3.2. Thuật toán giải nén

* Quá trình giải nén LZW cũng không có gì phức tạp. Thêm vào đó nó có nhiều lợi thế hơn so với các phương thức nén tĩnh vì không cần một từ điển hay những thông tin tạp phí cần thiết cho quá trình thuật giải nén.
* Một từ điển mới đồng nhất với từ điển gốc đã tạo trong khi nén được tái tạo lại trong quá trình giải nén này.Quá trình mã hóa và giải mã cần phải sử dụng cùng 1 từ điển khởi đầu, trong trường hợp này là 256 kí tự của bảng mã ASCII Sau đây là cơ chế nó hoạt động. Bộ giải mã LZW trước hết đọc một chỉ mục (là 1 số nguyên), tìm chỉ mục đó trong từ điển, và cho ra chuỗi con gắn với chỉ mục đó. Kí tự đầu tiên của chuỗi con này được cộng thêm vào chuỗi đang làm việc.
* Chuỗi mới tạo ra được thêm vào từ điển (hoàn toàn giống với quá trình nén). Chuỗi đã được giải mã lại trở thành chuỗi đang làm việc và cứ thế quá trình này được tiếp tục.
* Ví dụ: Các bước thực hiện giải mã chuỗi “65 66 67 257 256 67 260 68”
* Bước 1: 2 số chỉ mục đầu tiên sẽ luôn được tồn tại trong “từ điển”. Chỉ mục ‘65’ cho ra ký tự ‘A’, chỉ mục ‘66’ cho ra ký tự ‘B’. Lấy ký tự ‘A’ thêm vào đầu ký tự tiếp theo là ký tự ‘B’ ta được ‘AB’, ‘AB’ không có trong “từ điển” nên gán ‘AB’ cho chỉ mục 256.
* Bước 2: Chỉ mục thứ ba 67 tương ứng trong “từ điển” là ‘C’. ‘C’ sẽ được gửi ra. ‘BC’ không có trong từ điển nên gán ‘BC’ cho chỉ mục 257.
* Bước 3: Chỉ mục thứ tư 257 tương ứng trong “từ điển” là ‘BC’. ‘BC’ sẽ được gửi ra. ‘CB’ không có trong từ điển nên gán ‘CB’ cho chỉ mục 258.
* Bước 4: Chỉ mục thứ năm 256 tương ứng trong “từ điển” là ‘AB’. ‘AB’ sẽ được gửi ra. ‘BCA’ không có trong từ điển nên gán ‘BCA’ cho chỉ mục 259.
* Bước 5: Chỉ mục thứ sáu 67 tương ứng trong “từ điển” là ‘C’. ‘C’ sẽ được gửi ra. ‘ABC’ không có trong từ điển nên gán ‘ABC’ cho chỉ mục 260.
* Bước 6: Chỉ mục thứ bảy 260 tương ứng trong “từ điển” là ‘ABC’. ‘ABC’ sẽ được gửi ra. ‘CA’ không có trong từ điển nên gán ‘CA’ cho chỉ mục 261.
* Bước 7: Chỉ mục thứ tám 68 tương ứng trong “từ điển” là ‘D’. ‘D’ sẽ được gửi ra. ‘ABCD’ không có trong từ điển nên gán ‘ABCD’ cho chỉ mục 262.

### 2.4. Ưu điểm, nhược điểm của phương pháp mã hoá LZW

* Ưu điểm:
* Thuật toán nén LZW có các ưu điểm là hệ số nén tương đối cao, trong tập tin
* Bên nhận có thể tự xây dựng bảng mã mà không cần bên gửi phải gửi kèm theo bản tin nén.
* Thuật toán LZW đã khắc phục được sự lãng phí về bộ nhớ mà các thuật toán trước không tận dụng được hết. Đồng thời khắc phục được sự cứng nhắc của thuật toán nén, góp phần làm thuật toán nén trở nên mềm dẻo hơn, có sức hấp dẫn hơn đối với người sử dụng.
* Nhược điểm:
* Nhược điểm của thuật toán này là tốn nhiều bộ nhớ, khó thực hiện dựa trên các mảng đơn giản (bé hơn 64KB).

# Chương 4: Thực nghiệm và đánh giá

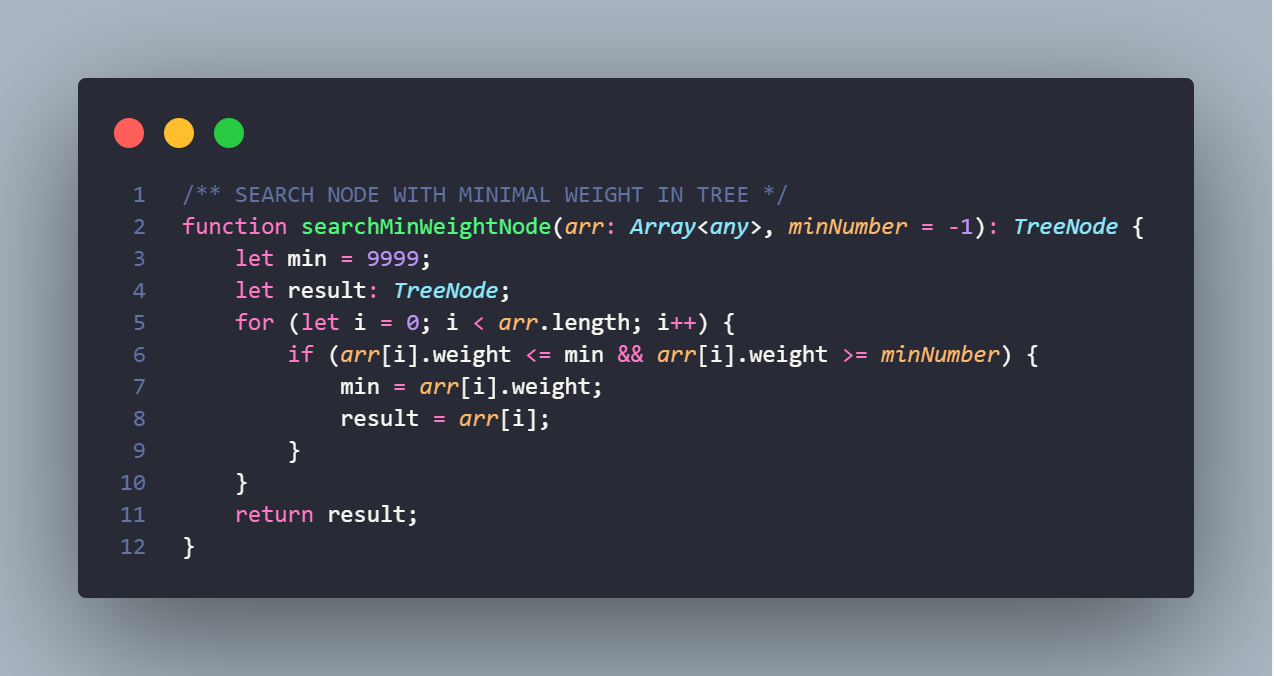
## 1. Huffman Coding

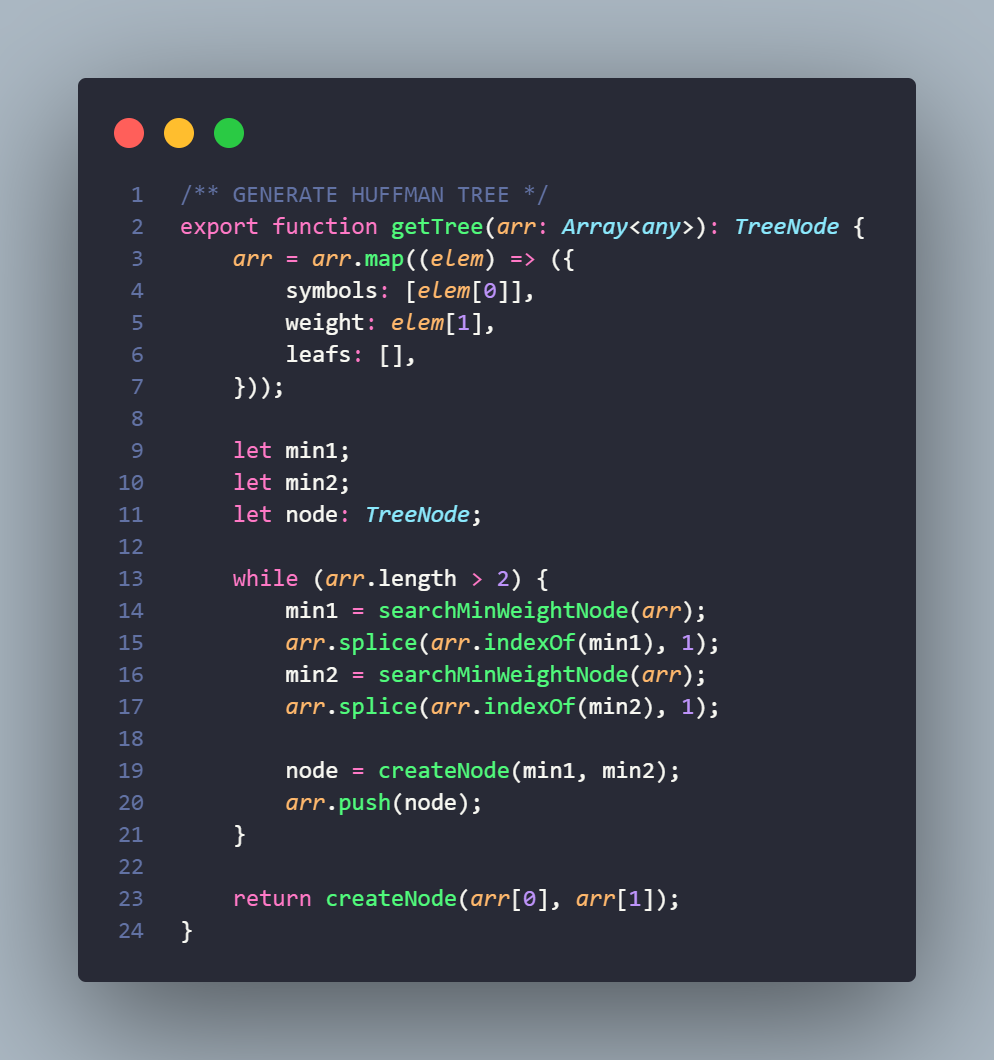






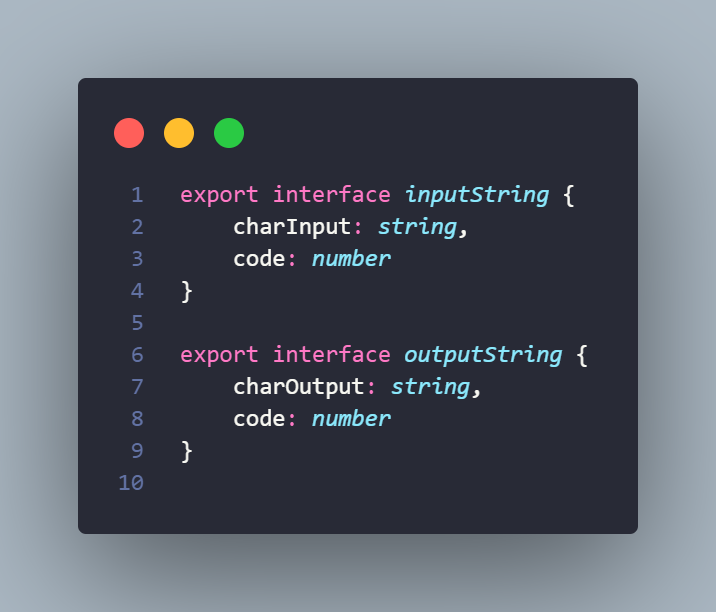








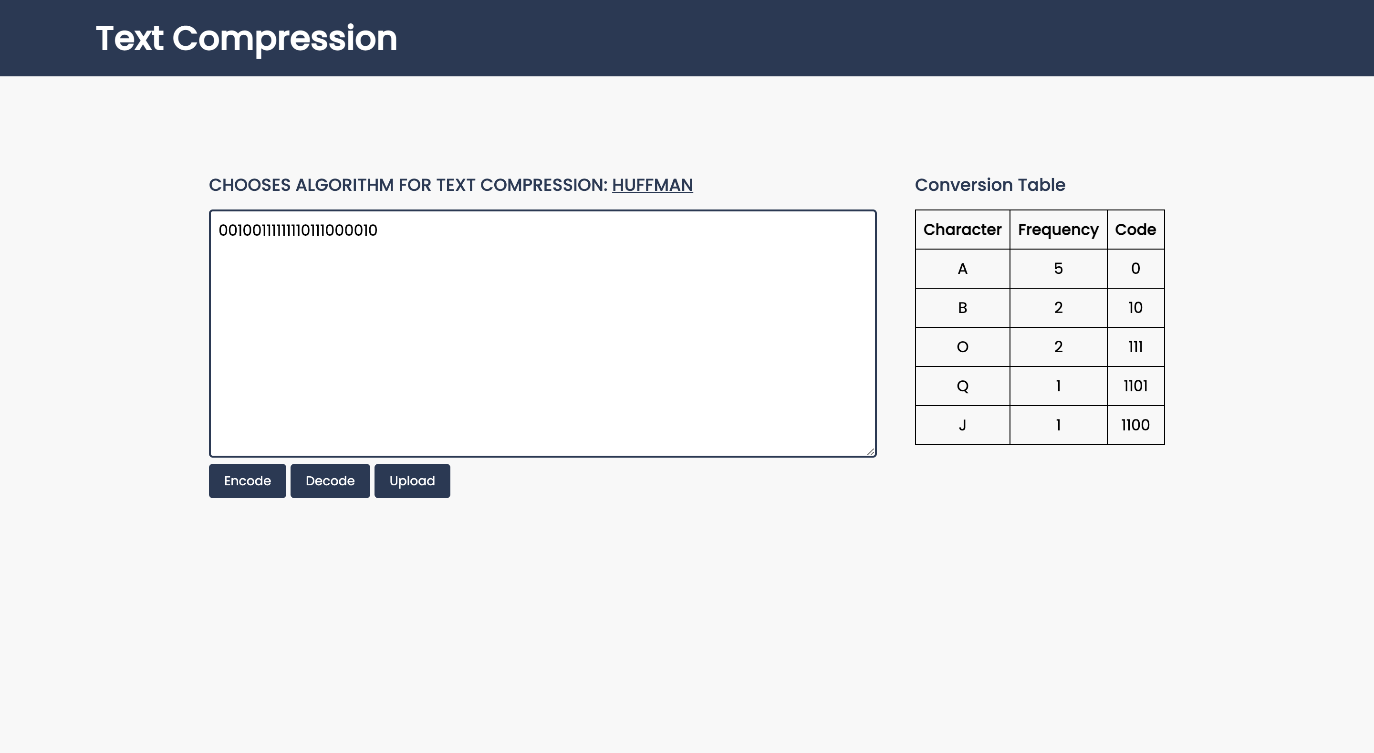
## 2. Lempel-Ziv-Welch (LZW)







## 3. Demo

* Source code: <https://github.com/Karhdo/CS232.M21.KHCL>
* Link demo: https://karhdo.github.io/CS232.M21.KHC

# Chương 5: Kết luận:

* Thuật toán Huffman có ưu điểm là hệ số nén tương đối cao, phương pháp thực hiện tương đối đơn giản, đòi hỏi ít bộ nhớ, có thể xây dựng dựa trên các mảng bé hơn 64KB. Nhược điểm của nó là phải chứa cả bảng mã vào tập tin nén thì phía nhận mới có thể giải mã được do đó hiệu suất nén chỉ cao khi ta thực hiện nén các tập tin lớn .
* Thuật toán nén LZW có các ưu điểm là hệ số nén tương đối cao, trong tập tin nén không cần phải chứa bảng mã. Nhược điểm của thuật toán này là tốn nhiều bộ nhớ, khó thực hiện dựa trên các mảng đơn giản (bé hơn 64KB). Từ các ưu và nhược điểm trên, Em thấy thuật toán nén LZW có các ưu điểm là hệ số nén tương đối cao, trong tập tin nén không cần phải chứa bảng mã.
* Bài toán ứng dụng của chúng em mới chỉ dừng lại ở việc nén dữ liệu từ file txt và còn sơ sài, có nhiều thiếu sót. Nhưng không dừng lại ở đó, chúng em sẽ có hướng phát triển để đưa được bài toán vào thực tiễn có thể nén được dữ liệu từ nhiều nguồn dữ liệu hơn. Đồng thời không ngừng học hỏi, tiếp thu kiến thức để có thể sử dụng những thuật toán cải tiến hơn, hiệu quả hơn vào việc xây dựng chương trình giúp chương trình nén nhanh hơn, tỉ lệ nén cao hơn.

# Chương 6: Tài liệu tham khảo

## 1. Thuật toán Huffman:

* <https://chidokun.github.io/2021/07/huffman-coding-p1/>
* <http://dulieu.tailieuhoctap.vn/books/cong-nghe-thong-tin/co-so-du-lieu/file_goc_768105.pdf>
* <https://vi.wikipedia.org/wiki/M%C3%A3_h%C3%B3a_Huffman>

## 2. Thuật toán Lempel-Ziv-Wel:

* <https://tailieutuoi.com/tai-lieu/bai-tap-nhom-mon-ly-thuyet-thong-tin-de-tai-lempel-ziv-encoding>
* <https://tailieu.vn/doc/bao-cao-nghien-cuu-cac-thuat-toan-nen-du-lieu-thuat-toan-lzw--163595.html?fbclid=IwAR0S0qvMDIrHbg-dlmGbqS9ZctDwkZn4N6xjzws_0LNT6Ak8B3wdYInQg84>
* <https://vi.wikipedia.org/wiki/LZW>