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 VĂN BẢN VÀ ẢNH VỚI THUẬT TOÁN HUFFMAN, LZW, SHANNON VÀ FANO

GIẢNG VIÊN HƯỚNG DẪN: **TH.S ĐỖ VĂN TIẾN**

SINH VIÊN THỰC HIỆN: **LỮ THỊ THÚY QUỲNH - 20521826**
BÙI VĂN THUẬN - 20521990
TRẦN VĨNH TUÂN - 20522127

TP. HỒ CHÍ MINH, 7/2023

LỜI CẢM ƠN

Sau quá trình học tập và rèn luyện tại Trường Đại học Công Nghệ Thông Tin, chúng em đã được trang bị các kiến thức cơ bản, các kỹ năng thực tế để có thể hoàn thành đồ án môn học của mình.

Chúng em xin gửi lời cảm ơn chân thành đến thầy Đỗ Văn Tiến đã tận tâm hướng dẫn, truyền đạt những kiến thức cũng như kinh nghiệm cho chúng em trong suốt thời gian học tập môn truy xuất thông tin.

Trong quá trình làm đồ án môn học, chúng em chắc chắn sẽ không tránh được những sai sót không đáng có. Mong nhận được sự góp ý cũng như kinh nghiệm quý báu của các các thầy để được hoàn thiện hơn và rút kinh nghiệm cho những môn học sau. Chúng em xin chân thành cảm ơn!

TP. Hồ Chí Minh, tháng 7 năm 2023.

MỤC LỤC

BẢNG PHÂN CHIA CÔNG VIỆC

MSSV	Tên	Phần công việc	Mức độ hoàn thành
20521826	Lữ Thị Thúy Quỳnh	Làm slide, báo cáo, code và demo phần LZW.	100%
20521990	Bùi Văn Thuận	Làm slide, báo cáo, code và demo phần Huffman. Xây dựng chương trình ứng dụng.	100%
20522127	Trần Vĩnh Tuân	Làm slide, báo cáo, code và demo phần Shannon Fano	100%

Link github của nhóm: [Link]

I. Giới thiệu

Ở đồ án lần này, chúng tôi tiến hành thực nghiệm các thuật toán nén Huffman, LZW và Shannon-Fano trên hai tác vụ chính là nén tệp văn bản và ảnh. Từ đó đưa ra số liệu nhằm so sánh và đánh giá độ tối ưu của các thuật toán.

1.1. Tại sao phải nén dữ liệu?

1.1.1. Khái niệm về dữ liệu

Trong một bài toán, dữ liệu bao gồm một tập các phần tử cơ sở mà ta gọi là dữ liệu nguyên tử (atoms). Nó có thể là một chữ số, một ký tự,... nhưng cũng có thể là một con số hay một từ,... điều đó tùy thuộc vào từng bài toán.

1.1.2. Mục đích của việc nén dữ liệu

Một trong những chức năng chính của máy tính là xử lý dữ liệu và lưu trữ. Bên cạnh việc xử lý nhanh, người ta còn quan tâm đến việc lưu trữ được nhiều dữ liệu nhưng lại tiết kiệm được vùng nhớ và giảm chi phí lưu trữ. Về mặt lý thuyết thì các thiết bị lưu trữ là không có giới hạn nhưng ngày nay do nhu cầu xử lý nhiều tập tin, nhiều loại dữ liệu trong cùng một tệp do vậy mà kích thước tệp trở nên khá lớn.

Trong nhiều năm gần đây, mạng máy tính đã trở nên phổ biến trên thế giới. Sự ra đời của mạng đã thực hiện được ước mơ chinh phục khoảng cách giữa con người. Những lợi ích mà mạng cung cấp rất đa dạng và phong phú trên các lĩnh vực khác nhau của xã hội như cung cấp, trao đổi thông tin giữa các máy tính, giữa máy tính với server hoặc giữa các server với nhau. Điều này dẫn đến phải làm như thế nào để giảm thiểu thời gian, chi phí sửa dụng để trao đổi dữ liệu trên mạng. Nó cũng đồng nghĩa với việc bên cạnh nâng cao chất lượng của các thiết bị truyền dữ liệu trên mạng thì mặt khác chúng ta phải nghĩ ra một phương pháp nào không để sao cho việc truyền dữ liệu có hiệu quả hơn.

Tất cả những vấn đề trên nảy sinh ra khái niệm nén dữ liệu. Một trong những hình thức nén dữ liệu đầu tiên là hệ chữ Braille, là một hệ chữ dùng phương pháp mã hóa ký hiệu cho người mù có thể đọc và viết. Ngày nay, nén dữ liệu mang lại rất nhiều lợi ích khác nhau mà điển hình là:

- Đối với việc tìm kiếm thì đôi khi ta tìm kiếm thông tin trên dữ liệu nén lại nhanh hơn so với việc tìm kiếm thông tin trên dữ liệu không nén vì do dữ liệu lưu trữ ít nên số phép toán để tìm kiếm giảm và lượng thông tin cao.
- Nén dữ liệu đặc biệt hiệu quả với việc truyền dữ liệu trên mạng. Khi nén dữ liệu thì chi phí cho việc truyền dữ liệu trên mạng sẽ giảm mặt khác tốc đường truyền sẽ tăng lên bởi vì cùng một lượng thông tin đó thời gian truyền dữ liệu sẽ giảm.
- Nén file sẽ giúp người dùng giảm dung lượng cần thiết để lưu trữ dữ liệu. Sử dụng tệp nén có thể giải phóng dung lượng quý giá trên ổ cứng hoặc máy chủ web.
- Các phần mềm nén file thông dụng hiện nay đều tích hợp mật khẩu, ta có thể dùng mật khẩu để khóa file nén lại và chỉ cần nhập đúng mật khẩu để mở file. Nhờ vậy, ta dễ dàng bảo vệ những dữ liệu quan trọng hay những dữ liệu nhạy cảm một cách dễ dàng, giúp dữ liệu được tăng tính bảo mật.

Với những ưu điểm trên thì do đó, nén dữ liệu lầ giải pháp hợp lý nhất nhằm mục đích giảm chi phí cho người sử dụng.

Tóm lại những lợi ích mà nén dữ liệu mang lại xoay quanh vấn đề tiết kiệm tối đa chi phí cho việc mua các thiết bị lưu trữ dữ liệu và cho việc luân chuyển thông tin trên mạng.

Một số vấn đề gặp phải khi nén dữ liệu là:

- Các thuật toán thực hiện trước hết phải giảm chi phí lưu trữ.
- Các thuật toán được thực hiện nhanh, hiệu quả.

Với nhiều loại thông tin khác nhau mà ta có các kỹ thuật nén khác nhau, có hiệu quả khác nhau, ví dụ như nén tệp văn bản thường tiết kiệm 20% - 50%, còn đối với tập tin nhị phân khoảng 50% - 90%, tuy nhiên đối với các tập tin ngẫy nhiên thì luognwj không gian tiết kiệm được rất tít hoặc hầu như không tiết kiệm được (chẳng hạn như tệp *.exe,...). Do các dữ liệu có độ dư thừa khác nhau nên mỗi phương pháp nén nếu áp dụng đúng sẽ trở nên không cần thiết, không có độ linh hoạt.

1.2. Phương pháp nén không mất thông tin*

1.2.1. Đặc điểm

- Phương pháp nến không mất thôn tin cho phép khôi phục lại hoàn toàn khối dữ liệu ban đầu qua các chu trình nén – giải nén.
- Đòi hỏi phải có thiết bị lưu trữ và đường truyền lớn.
- Các thuật toán của nén không mất dữ liệu dựa vào việc thay thế một nhóm các ký tự trùng lặp bởi một nhóm các ký tự đặc biệt khác ngắn hơn không quan tâm tới ý nghĩa của dữ liệu.

1.2.2. Ba dạng thuật toán nén không mất thông tin

- Các thuật toán mã hóa thống kê:
  - Các thuật toán này hoạt động dựa trên tần suất xuất hiện của các ký tự mã trong khối thông tin. Giảm số lượng bit dùng để biểu diễn các ký tự mã xuất hiện thường xuyên.
  - Tăng số lượng bit dùng để biểu diễn những ký tự mã ít xuất hiện.
- Các thuật toán dựa trên sự thay thế các chuỗi: các thuật toán này nén cá chuỗi chứa các ký tự đồng nhất.
- Các thuật toán dựa trên từ điển:
  - Giảm số lượng các bit dùng để chứa các từ xuất hiện thường xuyên.
  - Tăng số lượng các bit dùng để chứa các từ xuất hiện thưa thớt.

1.2.3. Đặc tính

- Thuật toán đơn giản.
- Tỷ lệ nén thấp.
- Thích hợp với nén ảnh và văn bản.

**Do hai thuật toán chính được nghiên cứu trong đồ án này là Huffman và Shannon-Fano nên chúng ta chỉ tìm hiểu về phương pháp nén không mất thông tin.*

1.3. Mô tả bài toán

Ở đồ án lần này, chúng tôi quyết định đánh giá và so sánh các thuật toán Huffman, LZW và Shannon-Fano trên hai tác vụ là nén tệp văn bản và ảnh. Từ đó, đưa ra kết luận cũng như so sánh ưu nhược điểm của từng thuật toán.

Nén tệp văn bản:

- Input: Một tập tin văn bản có định dạng là *.txt.
- Output: Một file *.txt chứa thông tin đã mã hóa của tệp văn bản đầu vào với kích thước file nhỏ hơn.

Nén ảnh:

- Input: Một ảnh màu có định dạng là *.jpg.
- Output: Một file *.txt chứa thông tin đã mã hóa của ảnh đầu vào với kích thước file nhỏ hơn.

II. Sơ lược về các thuật toán dùng để thực nghiệm:

2.1. Thuật toán Huffman

2.1.1. Giới thiệu

- Huffman là một thuật toán mã hóa dùng để nén dữ liệu, thuộc hình thức nén không mất thông tin. Dựa trên bảng tần suất xuất hiện các kí tự cần mã hóa để xây dựng một bộ mã nhị phân cho các kí tự đó sao cho dung lượng (số bit) sau khi mã hóa là nhỏ nhất.
- Giảm số bit để biểu diễn 1 ký tự.
- Dùng chuỗi bit ngắn hơn để biểu diễn ký tự xuất hiện nhiều lần.
- Sử dụng mã tiền tố để phân cách các ký tự.

2.1.2. Quá trình nén

B1: Duyệt file, lập bảng thống kê tần suất xuất hiện của mỗi ký tự.

B2: Xây dựng cây Huffman dựa vào bảng thống kê.

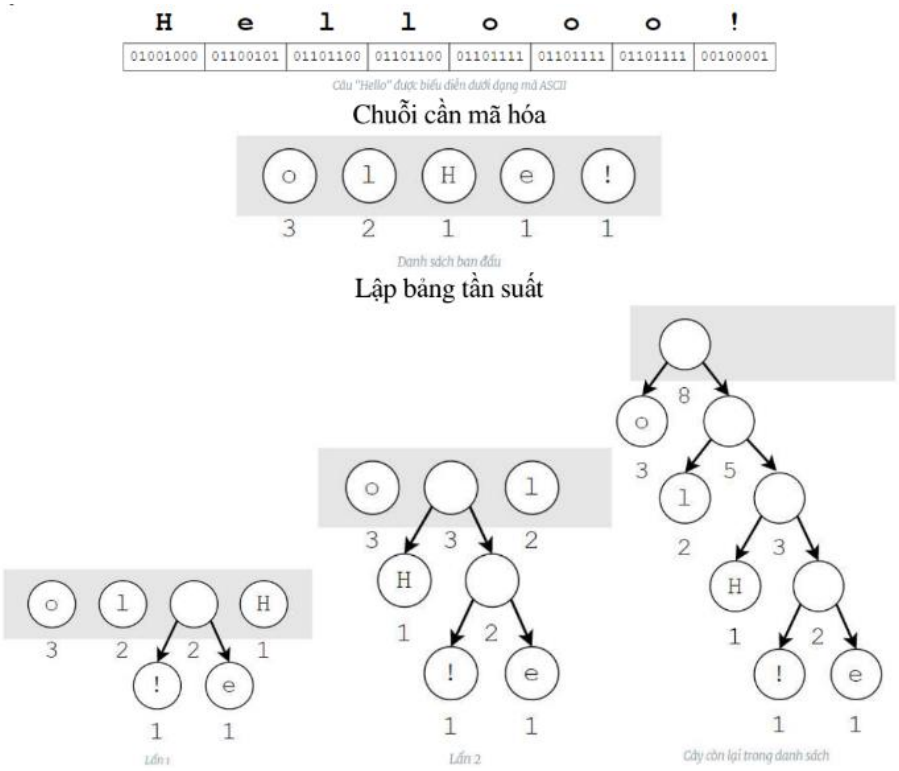
- B2.1: Tạo danh sách chứa các nút lá bao gồm phần tử đầu vào và trọng số nút là tần suất xuất hiện của nó. Sắp xếp các nút lá theo thứ tự giảm dần.
- B2.2: Từ danh sách này, lấy ra 2 phần tử có tần suất xuất hiện ít nhất. Sau đó gắn 2 nút vừa lấy ra vào một nút gốc mới với trọng số là tổng của 2 trọng số ở nút vừa lấy ra để tạo thành một cây.
- B2.3: Đẩy cây mới vào lại danh sách. Sau đó, sắp xếp lại thứ tự các nút lá.
- B2.4: Lặp lại bước 2 và 3 cho đến khi danh sách chỉ còn một nút gốc duy nhất của cây.
- B2.5: Nút còn lại chính là nút gốc của cây Huffman.

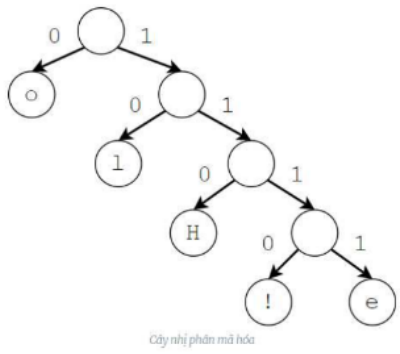
B3: Sinh mã Huffman cho mỗi ký tự dựa vào cây Huffman.

B4: Duyệt file, thay toàn bộ ký tự bằng mã Huffman tương ứng.

B5: Lưu lại cây Huffman (bảng mã) dùng cho việc giải nén. Xuất file nén.

Ví dụ:





Đánh mã cho cây Huffman

H	110
e	1111
l	10
o	0
!	1110

Mã của mỗi ký tự

H	e	l	l	o	o	o	!
110	1111	10	10	0	0	0	1110

Chuỗi ký tự được mã hóa hoàn chỉnh

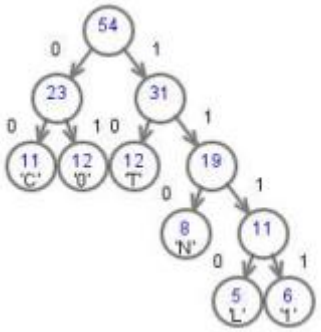
2.1.3. Quá trình giải nén

B1: Xây dựng lại cây Huffman từ thông tin giải mã đã lưu.
B2: Duyệt file, đọc lần lượt từng bit trong file nén và duyệt cây.
B3: Xuất ký tự tương ứng khi duyệt hết nút lá.
B4: Thực hiện B2, B3 cho đến khi duyệt hết file.

B5: Xuất file đã giải nén.

Ví dụ:

- Input:00110101011110111111111010011100000110110101010111101010
  10100000000111011101110111000000010101010111111110101011101011
  10110110010101101101 0



Vậy giải nén được là :
F=”CNTT10110CLCCNNTTT10000CCCCLLLLCCCTTTT11000
NTNNN000TNT”

2.1.4. Ưu nhược điểm

2.1.4.1. Ưu điểm

- Hệ số nén tương đối cao.
- Phương pháp thực hiện đơn giản.
- Đòi hỏi ít bộ nhớ.

2.1.4.2. Nhược điểm

- Mất hai lần duyệt file khi nén.
- Phải lưu trữ thông tin giải mã vào file nén.
- Phải xây dựng lại cây Huffman khi giải nén.

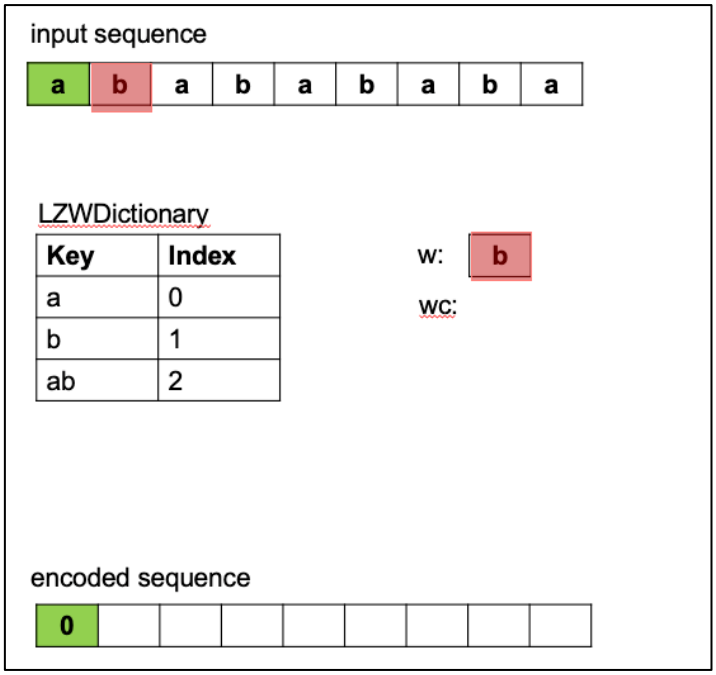
2.2. Thuật toán LZW

2.2.1. Lịch sử

Lempel – Ziv – Welch (LZW) là một thuật toán nén dữ liệu không mất dữ liệu phổ quát được tạo ra bởi Abraham Lempel , Jacob Ziv và Terry Welch . Nó được Welch công bố vào năm 1984 như là một cách triển khai cải tiến của thuật toán LZ78 do Lempel và Ziv xuất bản vào năm 1978. Thuật toán này dễ thực hiện và có tiềm năng cho thông lượng rất cao trong việc triển khai phần cứng. [1] Đây là thuật toán của tiện ích nén tệp Unix được sử dụng rộng rãi và được sử dụng ở định dạng ảnh GIF.

2.2.2. Quá trình nén

- LZW hoạt động đựa 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.



Nguyên tắc hoạt động

- 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 mã hóa riêng.
- Nếu lần sau gặp lại xâu kí tự đó, xâu kí tự sẽ được thay thế bằng mã hóa của nó.
- Tạo một mảng rất lớn dùng để lưu giữ các xâu kí tự đã gặp.(từ điển)
- Khi các byte dữ liệu cần nén được đem đến, chúng liền được giữ lại trong một bộ đệm chứa 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 mã hóa của chuỗi.
- Nếu có rồi thì mã hóa của chuỗi được đem ra thay cho chuỗi ở dòng dữ liệu ra.

Các bước trong quá trình nén:

1. Khởi tạo từ điển ban đầu chứa các ký tự đơn (nếu là mã ASCII thì ban đầu có 256 ký tự).
2. Đọc vào một chuỗi ký tự từ tệp tin ban đầu.
3. Tiến hành nén chuỗi ký tự đó bằng cách tìm chuỗi con trong từ điển trùng với chuỗi được đọc vào.
4. Nếu chuỗi con đó chưa có trong từ điển, thì thêm chuỗi con đó vào từ điển và đưa ra mã hóa của chuỗi ký tự trước đó.
5. Nếu chuỗi con đó đã có trong từ điển, thì sử dụng mã hóa đã có của chuỗi con đó để nén chuỗi ký tự trước đó.
6. Lưu các thông tin về mã hóa vào tệp tin đích.

Ý tưởng thuật toán:

Khởi tạo bảng với các chuỗi ký tự đơn
P = ký tự đầu vào đầu tiên
WHILE không kết thúc luồng đầu vào
    C = ký tự đầu vào tiếp theo
    NẾU P + C nằm trong bảng chuỗi
        P = P + C
    ELSE
        xuất mã cho P
        thêm P + C vào bảng chuỗi
        P = C
END WHILE
xuất mã đầu ra cho P

- Code thực thi cho quá trình trên:

```python
def encoding(file_path):
    with open(file_path, 'r') as f:
        s1 = f.read()
    table = defaultdict(int)
    for i in range(256):
        ch = chr(i)
        table[ch] = i

    p = s1[0]
    code = 256
    output_code = []
    for i in range(len(s1)):
        if i != len(s1) - 1:
            c = s1[i + 1]
        if p + c in table:
            p = p + c
        else:
            output_code.append(table[p])
            #print(p, table[p])
            table[p + c] = code
            code += 1
            p = c
        c = ""
    output_code.append(table[p])
    #print(p, table[p])

    return output_code
```

VD: Mã hóa dãy kí tự sau : BABAABAAA

1. **BA** is not in the Dictionary; insert **BA**, output the code for its prefix: **code(B)**
2. **AB** is not in the Dictionary; insert **AB**, output the code for its prefix: **code(A)**
3. **BA** is in the Dictionary.
   BAA is not in Dictionary; insert **BAA**, output the code for its prefix: **code(BA)**
4. **AB** is in the Dictionary.
   ABA is not in the Dictionary; insert **ABA**, output the code for its prefix: **code(AB)**
5. **AA** is not in the Dictionary; insert **AA**, output the code for its prefix: **code(A)**
6. **AA** is in the Dictionary and it is the last pattern: output its code: **code(AA)**

output	Dictionary	
?	Code Word	String
66	256	BA
65	257	AB
256	258	BAA
257	259	ABA
65	260	AA
260		

The compressed message is: **<66><65><256><257><65><260>**

2.2.3. Quá trình giải nén

Bộ giải nén LZW tạo cùng một bảng chuỗi trong quá trình giải nén. Nó bắt đầu với 256 mục nhập bảng đầu tiên được khởi tạo thành các ký tự đơn. Bảng chuỗi được cập nhật cho từng ký tự trong luồng đầu vào, ngoại trừ ký tự đầu tiên. Giải mã được thực hiện bằng cách đọc mã và dịch chúng thông qua bảng mã đang được xây dựng.

Ý tưởng thuật toán:

Khởi tạo từ điển ban đầu với các kí tự đơn trong ascii

Old= giá trị trong đầu tiên của mã output

S= kí tự tương ứng với mã output old

C= kí tự đầu tiên của chuỗi s

Mở file để ghi dữ liệu

Ghi giá trị đầu tiên vào

Thực hiện vòng lặp qua từng mã trong output code:

n = mã tiếp theo trong mảng

Nếu n không có trong từ điển:

S= mã của kí tự trước đó hay old

S=ký tự trước đó old+ ký tự đầu tiên của old

Nếu n có trong từ điển:

S= kí tự của mã n

Ghi s vào file

Cập nhật ký tự c và thêm cặp khóa mới vào từ điển

Tăng mã count

Cập nhật old = mã vào n đang xét

Lặp lại cho hết vòng lặp

Code cho thuật toán trên:

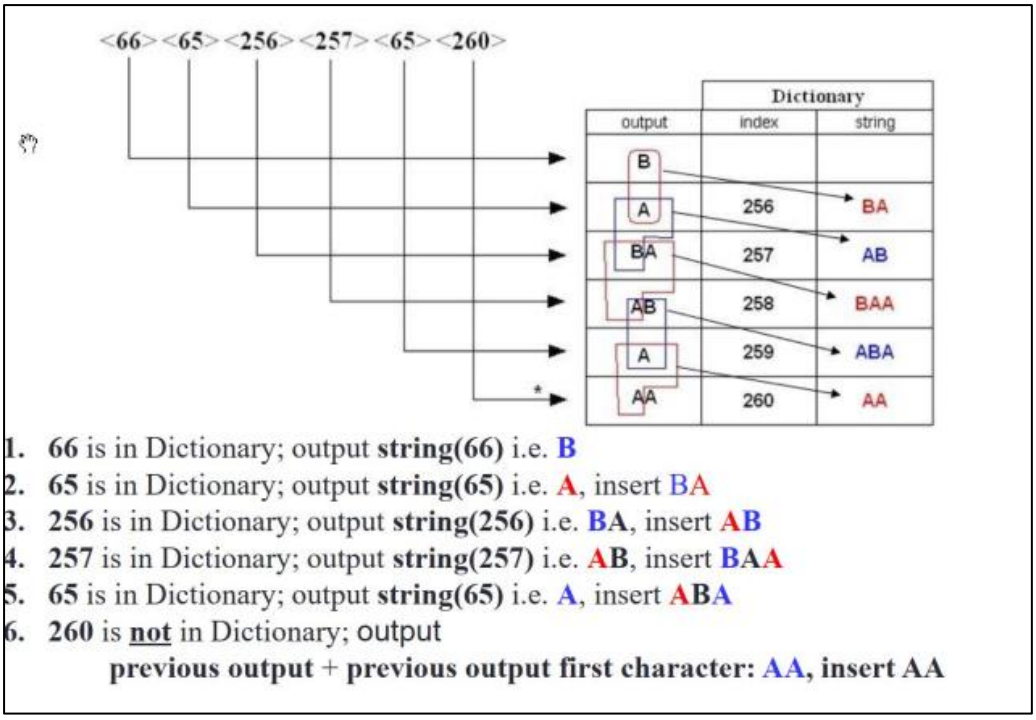
```
def decoding(op, file_path):
    # Khởi tạo từ điển ban đầu với các giá trị mặc định là chuỗi rỗng
    table = defaultdict(str)
    for i in range(256):
        ch = chr(i)
        table[i] = ch
    # Khởi tạo các biến ban đầu
    old = op[0]  # Giá trị đầu tiên trong op
    s = table[old]  # Giá trị tương ứng với old trong từ điển
    c = s[0]  # Ký tự đầu tiên của s
    count = 256  # output code cho những giá trị không có trong từ điển

    # Mở tệp tin để ghi dữ liệu giải nén
    with open(file_path, 'w') as f:
        # Ghi giá trị ban đầu của s vào tệp tin
        f.write(s)
        # Lặp qua các phần tử trong op (trừ phần tử đầu tiên)
        for i in range(len(op) - 1):
            n = op[i + 1]  # Giá trị hiện tại trong op

            # Nếu giá trị n không có trong từ điển
            if n not in table:
                s = table[old]
                s += c
            else:
                s = table[n]
            # Ghi giá trị s vào tệp tin
            f.write(s)

            # Cập nhật ký tự c
            c = s[0]
            # Thêm một cặp khóa-giá trị mới vào từ điển
            table[count] = table[old] + c
            count += 1
            # Cập nhật giá trị old
            old = n
```

- Ví dụ: giải nén <66> <65> <256> <257> <65> <260>



2.2.4. Ưu nhược điểm của LZW

Ưu điểm của LZW:

- Hiệu suất nén tốt: LZW thường có hiệu suất nén tốt, đặc biệt là đối với các loại dữ liệu có sự lặp lại nhiều, như văn bản, tệp tin hình ảnh và âm thanh. Nó có thể nén dữ liệu thành các mã ngắn hơn, giúp giảm kích thước tệp tin.
- Không mất dữ liệu: LZW là một thuật toán nén mất tích (lossless), có nghĩa là nén và giải nén dữ liệu không làm mất bất kỳ thông tin nào. Khi giải nén, dữ liệu ban đầu sẽ được khôi phục chính xác như ban đầu.
- Đơn giản và dễ hiện thực: Thuật toán LZW có cấu trúc đơn giản và dễ hiện thực. Nó chỉ cần một bảng tra cứu (dictionary) để lưu trữ các mẫu đã xuất hiện trước đó trong quá trình nén và giải nén.

Nhược điểm của LZW:

- Tốn không gian lưu trữ: Một nhược điểm của LZW là nó yêu cầu một bảng tra cứu lớn để lưu trữ các chuỗi con đã xuất hiện trước đó trong quá trình nén và giải nén. Điều này có thể tạo ra một overhead không gian lưu trữ đáng kể, đặc biệt là khi làm việc với các tệp tin có kích thước lớn.
- Không tối ưu với dữ liệu ngẫu nhiên: LZW thường hoạt động tốt trên các dữ liệu có sự lặp lại nhiều. Tuy nhiên, đối với các tệp tin có tính ngẫu nhiên cao, LZW có thể không có hiệu suất nén tốt như mong đợi và thậm chí có thể tăng kích thước tệp tin sau quá trình nén.

2.3. Thuật toán Shannon-Fano

2.3.1. Giới thiệu

- Vào khoảng năm 1948, cả Claude E. Shannon (1948) và Robert M. Fano (1949) đã đề xuất độc lập hai thuật toán mã hóa nguồn khác nhau để mô tả hiệu quả nguồn không có bộ nhớ rời rạc. Thật không may, mặc dù khác nhau, cả hai phương pháp đều được biết đến dưới cùng một tên mã Shannon-Fano.
- Có một số lý do cho sự kết hợp này. Có một điều, trong cuộc thảo luận về sơ đồ mã hóa của mình, Shannon đề cập đến sơ đồ của Fano và gọi nó là "về cơ bản là giống nhau" (Shannon, 1948, trang 17). Đối với một cách khác, cả hai chương trình mã hóa của Shannon và Fano đều giống nhau theo nghĩa là cả hai đều hiệu quả, nhưng các chương trình mã hóa không có tiền tố tối ưu có hiệu suất tương tự.

2.3.2. Ý tưởng chính

Cả 2 phương pháp này đều xây dựng mã tiền tố dựa trên một tập hợp các ký hiệu và xác suất của chúng (ước tính hoặc đo lường). Cụ thể:

- Phương pháp của Shannon: chọn mã tiền tố trong đó biểu tượng nguồn được cung cấp độ dài từ mã $li = [-log_2 pi\ ]$. Một cách phổ biến để chọn các từ mã là sử dụng phép mở rộng nhị phân của các xác suất tích lũy. Phương pháp này được đề xuất trong " Lý thuyết toán học về truyền thông " của Shannon (1948), bài báo của ông giới thiệu lĩnh vực lý thuyết thông tin.
- Phương pháp của Fano: chia các ký hiệu nguồn thành hai bộ ("0" và "1") với xác suất càng gần 1/2 càng tốt. Sau đó, các tập hợp đó tự được chia làm hai, và cứ tiếp tục như vậy, cho đến khi mỗi tập hợp chỉ chứa một ký hiệu. Từ mã cho biểu tượng đó là chuỗi "0" và "1" ghi lại nửa số chia mà nó nằm trên. Phương pháp này được đề xuất trong một báo cáo kỹ thuật sau đó của Fano (1949).

***Nhận xét:**

- Hai phương pháp Shannon và Fano thực chất là một, đều xây dựng trên cùng một cơ sở đồ dài từ mã tỉ lệ nghịch với xác suất xuất hiện, không cho phép lập mã một cách duy nhất vì sự chia nhóm trên cơ sở đồng đều và tổng xác suất nên có thể có nhiều cách chia.
- Sự lập mã theo cách chia nhóm trên cơ sở đồng xác suất tạo cho bộ mã có tính Prefix.

2.3.3. Shannon

2.3.3.1. Thuật toán nén

- B1: Sắp xếp các xác suất theo thứ tự giảm dần. Không mất tổng quát giả sử $p_1 \geq \cdots . \geq p_K$
- B2: Định nghĩa $q_1 = 0, q_i = \sum_{j=1}^{i-1} p_j \, \forall i = 2, \ldots . , K$
- B3: Đổi q_i sang cơ số 2, (biểu diễn trong cơ số 2) sẽ được một chuỗi nhị phân.
- B4: Từ mã được gán cho a_i là l_i kí hiệu lấy từ vị trí sau dấu phẩy của chuỗi nhị phân tương ứng với q_i, trong đó $li = [-log_2 pi\ ]$.\

2.3.3.2. Ví dụ

Mã hóa nguồn S = {a1, a2, a3, a4, a5, a6} với các xác suất lần lượt là 0,3; 0,25; 0,2; 0,12; 0,08; 0,05.

Tin a_i	Xác suất p_i	$q_i = \sum_{j=1}^{i-1} p_j$	Biểu diễn nhị phân	$l_i = [-log_2 p_i]$	Từ mã w_i
a_1	0,3	0	0,00	2	00
a_2	0,25	0,3	0,01001…	2	01
a_3	0,2	0,55	0,10001…	3	100
a_4	0,12	0,75	0,11000…	4	1100
a_5	0,08	0,87	0,11011…	4	1101
a_6	0,05	0,95	0,111100…	5	11110

2.3.4. Fano

2.3.4.1. Thuật toán nén

- B1: Sắp xếp các xác suất theo thứ tự giảm dần. Không mất tổng quát giả sử $p_1 \geq \cdots . \geq p_K$
- B2: Phân các xác suất thành 2 nhóm có tổng xác suất gần bằng nhau nhất.
- B3: Gán cho hai nhóm lần lượt các kí hiệu 0 và 1 (hoặc ngược lại).
- B4: Lặp lại bước 2 cho các nhóm con cho đến khi không thể tiếp tục được nữa.
- B5: Từ mã ứng với mỗi tin là chuỗi bao gồm các kí hiệu theo thứ tự lần lượt được gán cho các nhóm có chứa xác suất tương ứng của tin.

2.3.4.2. Ví dụ

Mã hóa nguồn S = {a1, a2, a3, a4, a5, a6} với các xác suất lần lượt là 0,3; 0,25; 0,2; 0,12; 0,08; 0,05.

Tin a_i	Xác suất	Phân nhóm lần				Từ mã w_i
		1	2	3	4	
a_1	0,3	0	0			00
a_2	0,25	0	1			01
a_3	0,2	1	0			10
a_4	0,12	1	1	0		110
a_5	0,08	1	1	1	0	1110
a_6	0,05	1	1	1	1	1111

*Chú ý: Trong nhiều trường hợp có nhiều hơn một cách chia thành các nhóm có tổng xác suất gần bằng nhau, ứng với mỗi cách chia có thể sẽ cho ra các bộ mã có chiều dài trung bình khác nhau.

2.3.5. Ưu điểm và hạn chế của Shannon-Fano

- **Ưu điểm:**
  - Đối với Shannon ta không cần xây dựng toàn bộ mã, mà chỉ cần lấy mã cho thẻ tương ứng với một trình tự nhất định.
  - Tốc độ thực thi nhanh.
  - Dễ thực hiện.
- **Hạn chế:**
  - Có thể có 2 mã khác nhau cho một kí tự tùy thuộc vào cách chúng ta xây dựng cây của mình.
  - Không đảm bảo mã tối ưu.
  - Mất 2 lần duyệt file khi nén.
  - Phải lưu trữ thông tin giải mã vào file nén.

III. So sánh tỉ lệ nén và tốc độ nén của các thuật toán Huffman, LZW, Shannon và Fano:

3.1. Mô tả bộ test

3.1.1. Text

- Gồm có 3 test, trong đó:

Test 1 là một chuỗi và lặp lại. Có kích thước là 5016 bytes

Nội dung file

HYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNH
YIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYI
RMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIR
MNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRM
NHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMN
HYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNH
YIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYI
RMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIRMNHYIR

Test 2 là một file văn bản chứa nội dung lớn 5.3 MB:

Link: https://www.kaggle.com/datasets/kewagbln/shakespeareonline

Nội dung file

This is the 100th Etext file presented by Project Gutenberg, and
is presented in cooperation with World Library, Inc., from their
Library of the Future and Shakespeare CDROMS. Project Gutenberg
often releases Etexts that are NOT placed in the Public Domain!!

Shakespeare

This Etext has certain copyright implications you should read!

Test 3 là một file lớn có dung lượng 6.3 MB :

Link https://www.gutenberg.org/files/1661/1661-0.txt

Nội dung file

The Project Gutenberg EBook of The Adventures of Sherlock Holmes
by Sir Arthur Conan Doyle
(#15 in our series by Sir Arthur Conan Doyle)

Copyright laws are changing all over the world. Be sure to check the
copyright laws for your country before downloading or redistributing
this or any other Project Gutenberg eBook.

3.1.2. Image

	Kích thước gốc (bytes)	Kích thước sau khi chuyển đổi thành mảng (bits)	Size
1.jpg	156,708	30 345 511	640 x 427
2.jpg	178,027	54 202 176	880 x 587
3.jpg	594,137	233 660 700	1920 x 1080

3.2. Phương pháp

3.2.1. Tỉ số nén

Tỉ số nén (compression ratio) là một đánh giá về hiệu quả của việc nén dữ liệu. Nó đo lường sự giảm kích thước của dữ liệu nén so với kích thước gốc của dữ liệu.

Tỉ số nén được tính bằng cách chia kích thước dữ liệu gốc cho kích thước dữ liệu sau khi nén. Công thức tỉ số nén được biểu diễn như sau:

Tỉ số nén = Kích thước dữ liệu gốc / Kích thước dữ liệu sau nén

3.2.2. Hiệu suất nén

Hiệu suất nén (compression performance) là một khái niệm để đánh giá tốc độ và hiệu quả của quá trình nén dữ liệu. Nó đo lường khả năng của thuật toán nén dữ liệu trong việc giảm kích thước của dữ liệu và tốn bao nhiêu thời gian và tài nguyên để thực hiện quá trình nén.
Ở đây ta tính dựa trên tỉ số nén

$$\text{Hiệu suất nén} = 1 - \frac{1}{\text{tỉ số nén}}$$

3.2.3. Tỉ lệ nén

Tỉ lệ nén (compression ratio) là một đánh giá về mức độ nén của dữ liệu sau khi áp dụng thuật toán nén. Nó đo lường sự giảm kích thước của dữ liệu sau khi nén so với kích thước gốc của dữ liệu.

Tỉ lệ nén = Kích thước dữ liệu gốc / Kích thước dữ liệu sau khi nén

3.3. Kết quả:

File		Data ban đầu (bytes)	Data sau khi nén (bytes)	Hiệu suất nén	Tỉ số nén	Thời gian nén (s)	Thời gian giải nén
Test 1	H	5.016	1674	66.63%	2.99642	0.00742	0.01785
	LZW	**5.016**	**972**	**80.62%**	**5.16049**	**0.002**	**0.00349**
	S	5.016	1881	62.5%	2.66667	0.00719	0.02297
	F	5.016	1672	66.67%	3	0.00471	0.02009
Test 2	**H**	**5.582.653**	**3157249**	**43.45 %**	**1.7682**	**43.62**	**46.49**
	LZW	5.582.653	3302528	40.84 %	1.69042	2.48	1.65
	S	5.582.653	3548557	36.44 %	1.57322	42.85	13.08
	F	5.582.653	3224635	42.24 %	1.73125	29.28	12.65
Test 3	**H**	**6.616.755**	**3682826**	**44.34 %**	**1.79665**	**71.82**	**60.23**
	LZW	6.616.755	3971008	39.99 %	1.66627	3.259	1.35
	S	6.616.755	4119948	37.73 %	1.60603	52.06	10.34
	F	6.616.755	3755034	43.25 %	1.7621	40.64	8.66

Nén ảnh

File		Data ban đầu (bit)	Data sau khi nén (bit)	Hiệu suất nén	Tỉ số nén	Thời gian nén (s)	Thời gian giải nén
Test 1	H	30 345 511	15 185 294	49.96 %	1.99835	9.96352	13.33859
	LZW	**30 345 511**	**11 597 376**	**61.78 %**	**2.61658**	**2.58432**	**1.57215**
	S	30 345 511	17 060 800	43.78 %	1.77867	10.86339	7.80134
	F	30 345 511	16 451 238	45.79 %	1.84457	13.95744	44.40579
Test 2	H	54 202 176	27 097 969	50.01 %	2.00023	27.72809	35.3615
	LZW	**54 202 176**	**20 029 079**	**63.05 %**	**2.70617**	**4.84693**	**3.5622**
	S	54 202 176	30 437 678	43.84 %	1.78076	31.28466	14.37404
	F	54 202 176	29 245 750	46.04 %	1.85334	25.3036	78.29486
Test 3	H	233 660 700	117 637 336	49.65 %	1.98628	490.84862	476.38434
	LZW	**233 660 700**	**66 068 058**	**71.72 %**	**3.53667**	**21.33664**	**41.10317**
	S	233 660 700	131 803 767	43.59 %	1.77279	581.03776	100.06862
	F	233 660 700	120 495 742	48.43 %	1.93916	109.38912	364.66621

3.4. Nhận xét:

Với test 1 là một file có kích thước nhỏ có nhiều chuỗi lặp lại

- LZW có hiệu suất nén cao nhất và thời gian xử lý nhanh nhất
- Những thuật toán còn lại cũng đem lại hiệu suất nén khá ổn trên 60%.
- Do file này có kích thước khá nhỏ nên thời gian giải nén và nén rất nhanh.

Với test 2 là một file ebook có kích thước lớn (5.3MB)

- Xét về hiệu suất nén huffman>fano>lzw>shannon
- Xét về thời gian nén và giải nén lzw>fano>shannon>huffman
- Hiệu suất nén của các thuật toán đều không cao < 45%
- LZW có thời gian nén và giải nén nhanh hơn rất nhiều so với các thuật toán còn lại

Với test 3 là tổng hợp các bài thơ của Shakespeare có kích thước lớn(6.3 MB)

- Ta có đánh giá tương tự như test 2

IV. Kết luận và hướng phát triển:

4.1. Kết luận:

Cả bốn thuật toán đều hoạt động nhanh và hiệu quả, nhưng vẫn có sự chênh lệch về tỉ lệ nén và thời gian nén khi thực thi trên những bộ test khác nhau. Và bốn test là chưa đủ để thấy rõ sự chênh lệch và ưu nhược điểm của từng thuật toán.

4.2. Hướng phát triển:

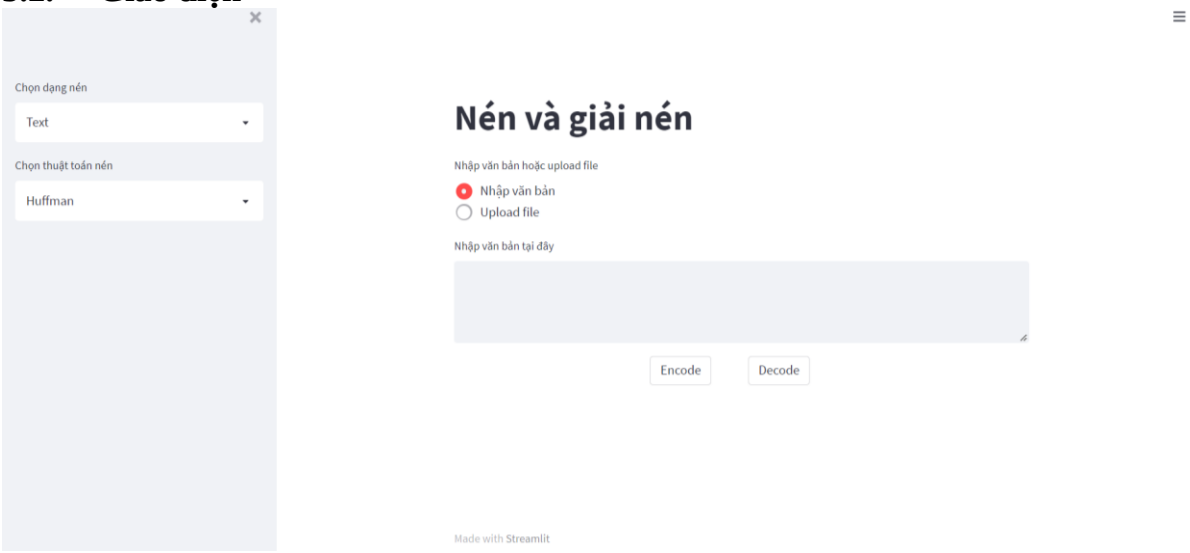
Chúng em sẽ tìm thêm nhiều thuật toán nén khác và làm đa dạng thêm bộ test để có thể đưa ra nhận xét chính xác hơn cho từng thuật toán cũng như hiểu thêm về chúng.

V. Chương trình ứng dụng:

5.1. Thư viện sử dụng

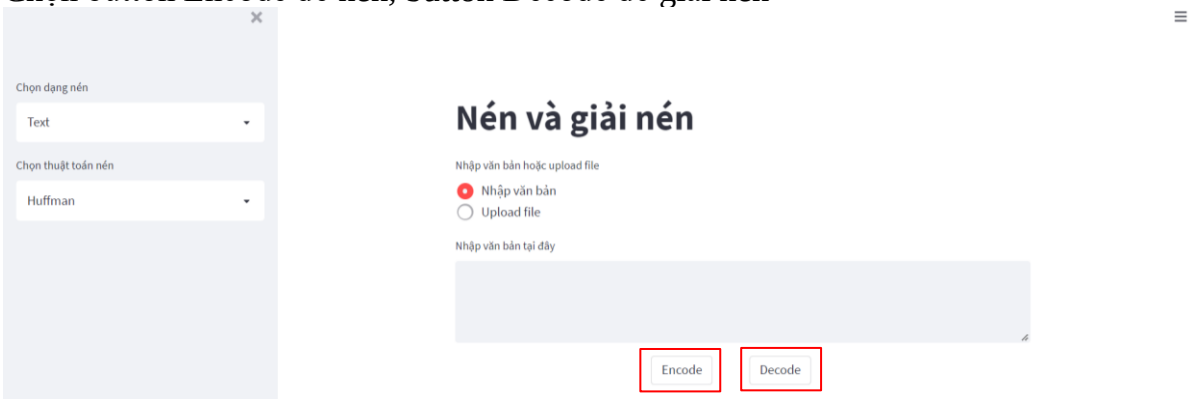
Streamlit là một thư viện mã nguồn mở và miễn phí được sử dụng để xây dựng các ứng dụng web tương tác nhanh chóng cho việc trình bày dữ liệu và mô hình trong Python. Với Streamlit, bạn có thể tạo các ứng dụng web chất lượng cao mà không cần phải viết mã HTML, CSS hoặc JavaScript.

5.2. Giao diện

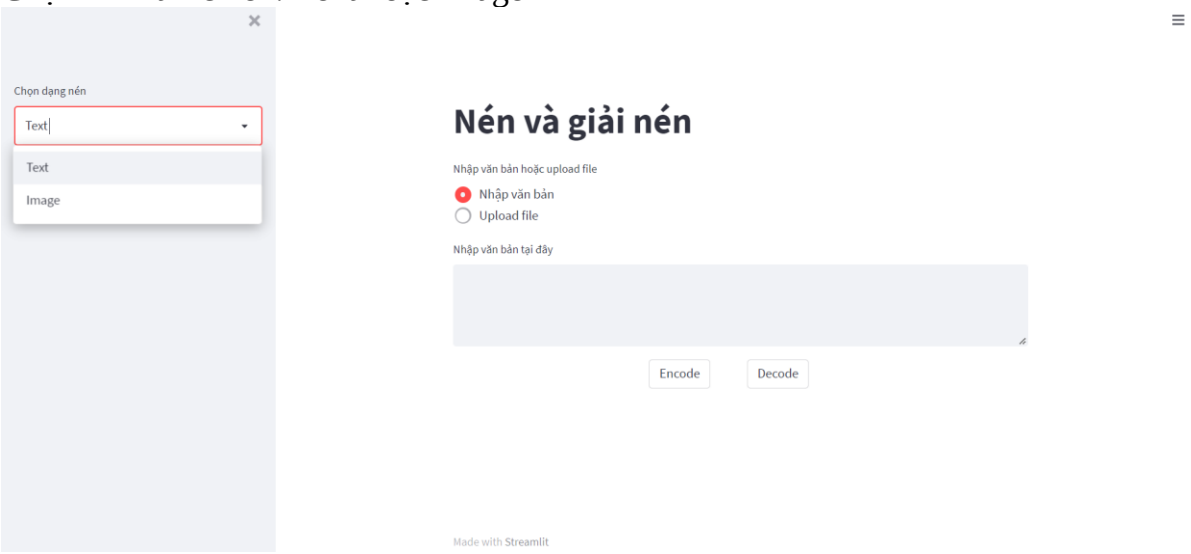


5.3. Các chức năng chính

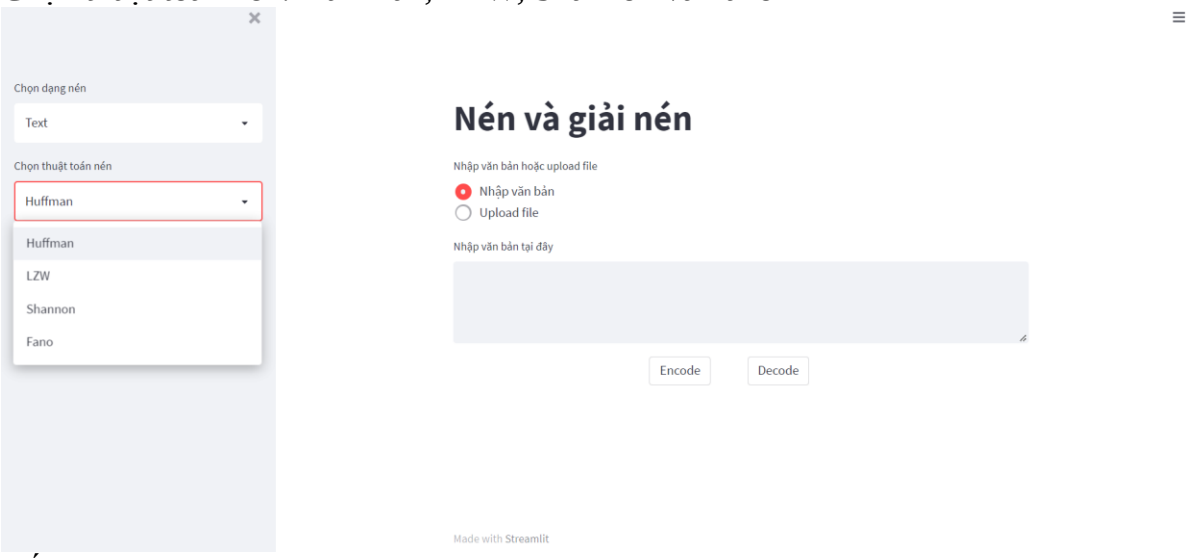
- Chọn button Encode để nén, button Decode để giải nén



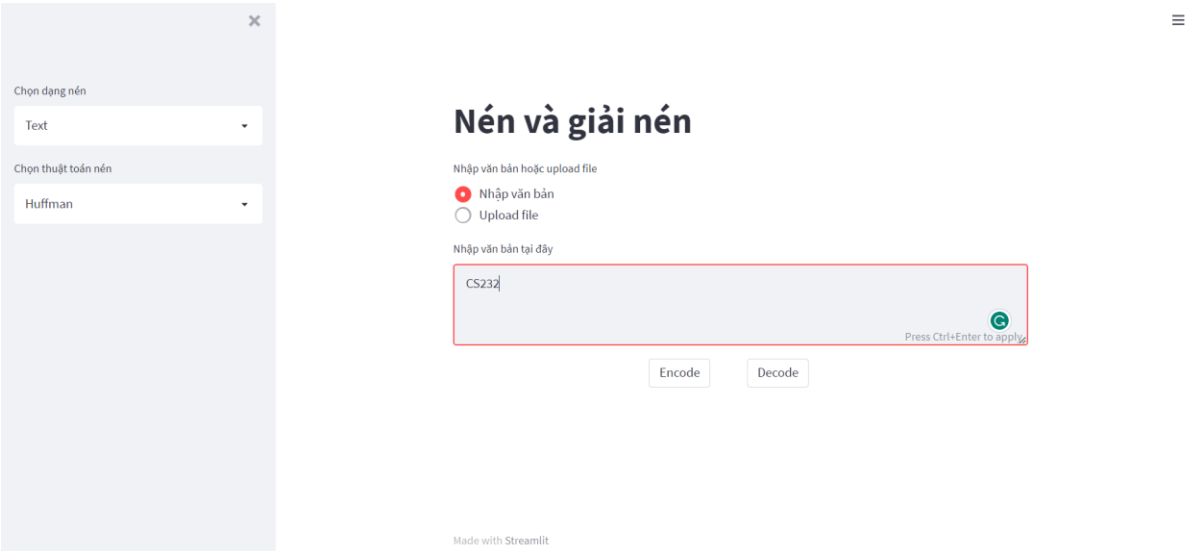
- Chọn hình thức nén: Text hoặc image



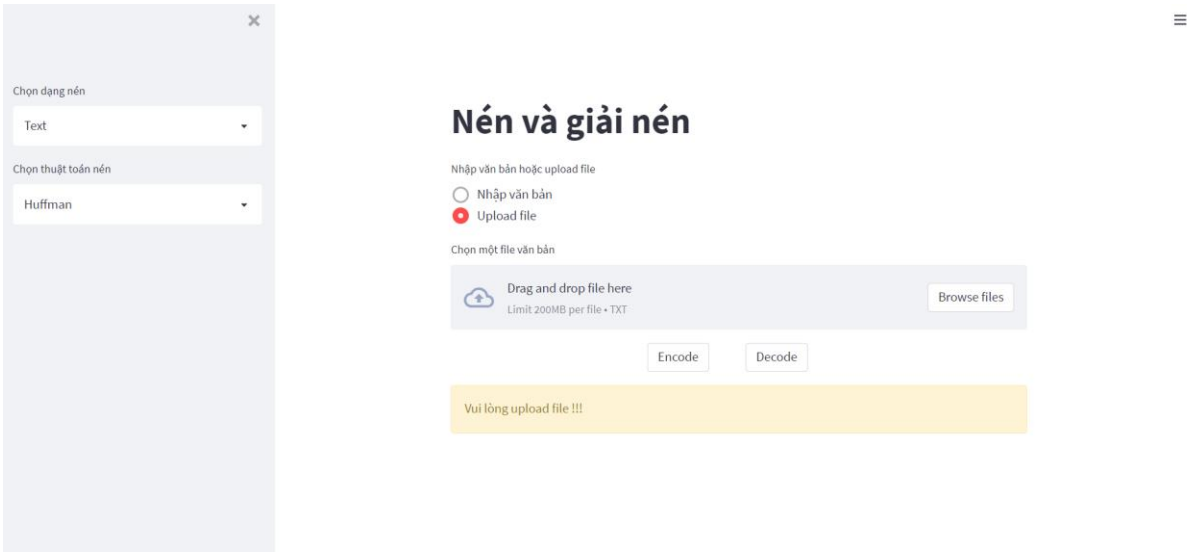
- Chọn thuật toán nén: Huffman, LZW, Shannon và Fano



- Đối với nén văn bản sẽ có 2 lựa chọn: Nhập đoạn văn bản hoặc Upload tệp văn bản

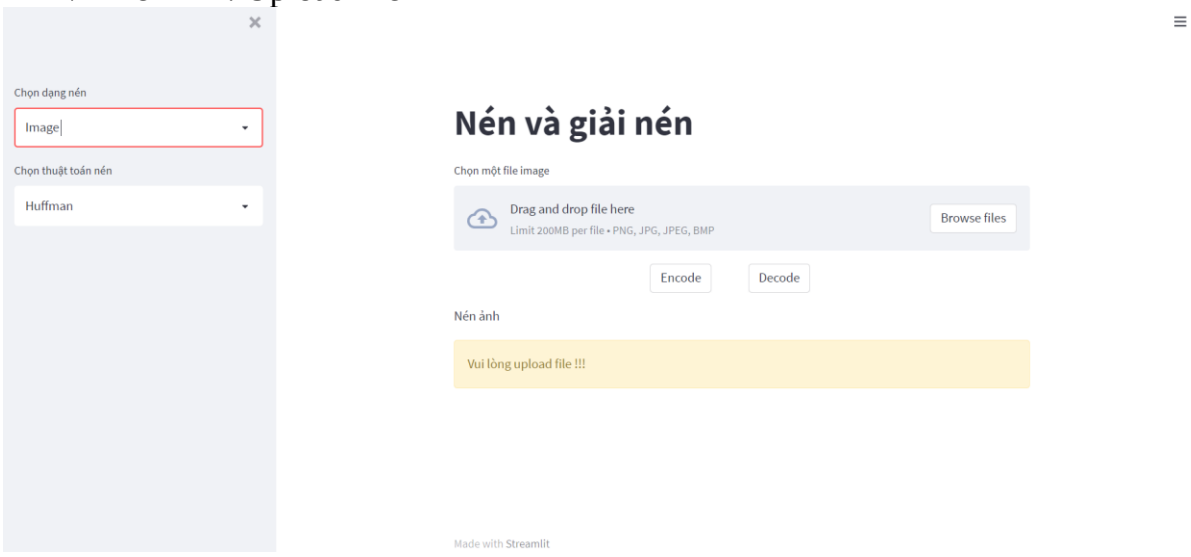


Nhập văn bản



Upload tệp văn bản

- Đối với nén ảnh: Upload file ảnh



TÀI LIỆU THAM KHẢO

[1] "LZW." In *Wikipedia tiếng Việt*, December 8, 2021. https://vi.wikipedia.org/w/index.php?title=LZW&oldid=67479845.

[2] "LZW (Lempel–Ziv–Welch) Compression Technique - GeeksforGeeks." Accessed July 14, 2023. https://www.geeksforgeeks.org/lzw-lempel-ziv-welch-compression-technique/.

[3] "LZW - Thuật Toán Nén Dữ Liệu - VOER." Accessed July 14, 2023. https://voer.edu.vn/m/lzw-thuat-toan-nen-du-lieu/8016a37b.

[4] Sharma, Gajendra. "Analysis of Huffman Coding and Lempel–Ziv–Welch (LZW) Coding as Data Compression Techniques," 2020.
"3.Pdf." Accessed July 14, 2023. http://web.mit.edu/6.02/www/f2011/handouts/3.pdf.

[5] GeeksforGeeks. "Huffman Coding | Greedy Algo-3," November 3, 2012. https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/.

[6] Kumar, Vikas. "Compression Techniques Vs Huffman Coding." *International Journal of Informatics and Communication Technology (IJ-ICT)* 4 (April 1, 2015): 29. https://doi.org/10.11591/ijict.v4i1.pp29-37.

[7] "Shannon–Fano Coding." In *Wikipedia*, June 28, 2023. https://en.wikipedia.org/w/index.php?title=Shannon%E2%80%93Fano_coding&oldid=1162295434.

[8] N, Sudhamshu B. "Shannon-Fano-Encoding-and-Decoding-for-Image-Compression." Python, May 8, 2023. https://github.com/sudhamshu091/Shannon-Fano-Encoding-and-Decoding-for-Image-Compression.

[9] "Sudhamshu091/Huffman-Encoding-Decoding-For-Image-Compression: Huffman Encoding and Decoding for Image Compression Using Python." Accessed July 14, 2023. https://github.com/sudhamshu091/Huffman-Encoding-Decoding-For-Image-Compression.