ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN

MÔN HỌC: TÍNH TOÁN ĐA PHƯƠNG TIỆN

ĐỀ TÀI

THUẬT TOÁN HUFFMAN VÀ SHANNON-FANO

Giảng viên hướng dẫn: Đỗ Văn Tiến

Sinh viên thực hiện: Tô Thanh Hiền - 19521490
Trần Vĩ Hào - 19521482
Lê Đặng Đăng Huy - 19521612

Lớp: CS232.M22.KHCL

Thành phố Hồ Chí Minh, ngày 21 tháng 06 năm 2022

MỤC LỤC

I. TÓM TẮT

Ở đồ án lần này, chúng tôi tiến hành thực nghiệm hai thuật toán nén Huffman 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. Và với những gì mà chúng tôi nhận được thông qua thực nghiệm thì bước đầu có thể kết luận Huffman vượt trội hơn Shannon-Fano trên bộ dữ liệu mà chúng tôi đã chuẩn bị. Để biết thêm chi tiết, mời bạn đọc bài báo cáo sau.

II. GIỚI THIỆU CHUNG

1. Tại sao phải nén dữ liệu?

a. 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.

b. 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.

2. Phương pháp nén không mất thông tin*

a. 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.

b. 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.

c. 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.*

3. Mô tả bài toán

Ở đồ án lần này, chúng tôi quyết định đánh giá và so sánh hai thuật toán Huffman 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.

III. THUẬT TOÁN HUFFMAN

1. Tác giả

Thuật toán được đề xuất bởi David A. Huffman khi ông còn là sinh viên Ph.D. tại MIT, và công bố năm 1952 trong bài báo "A Method for the Construction of Minimum-Redundancy Codes". Sau này Huffman đã trở thành một giảng viên ở MIT và sau đó ở khoa Khoa học máy

tính của Đại học California, Santa Cruz, Trường Kỹ nghệ Baskin (Baskin School of Engineering).

2. Ý tưởng chính

- 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.

3. Mã tiền tố

Để mã hóa các kí hiệu (kí tự, chữ số, ...) ta thay chúng bằng các xâu nhị phân, được gọi là từ mã của kí hiệu đó. Chẳng hạn bộ mã ASCII, mã hóa cho 256 kí hiệu là biểu diễn nhị phân của các số từ 0 đến 255, mỗi từ mã gồm 8 bít. Trong ASCII từ mã của kí tự "a" là 1100001, của kí tự "A" là 1000001. Trong cách mã hóa này các từ mã của tất cả 256 kí hiệu có độ dài bằng nhau (mỗi từ mã 8 bít). Nó được gọi là mã hóa với độ dài không đổi.

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

**Mã tiền tố là bộ các từ mã của một tập hợp các kí hiệu sao cho từ mã của mỗi ký hiệu không là tiền tố (phần đầu) của từ mã một ký hiệu khác trong bộ mã ấy.*

Đương nhiên mã hóa với độ dài không đổi là mã tiền tố.

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

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

4. Nén tĩnh (Static Huffman)

a. Tính chất

- Là cây nhị phân, mỗi nút chứa ký tự và trọng số (tần suất của ký tự đó).
- Mỗi ký tự được biểu diễn bằng 1 nút lá (tính tiền tố).
- Nút cha có tổng ký tự, tổng trọng số của 2 nút con.
- Các nút có trọng số, ký tự tăng dần từ trái sang phải.
- Các nút có trọng số lớn nằm gần nút gốc và ngược lại.

b. Mã Huffman

- Là chuỗi nhị phân được sinh ra dựa trên cây Huffman.
- Mã Huffman của ký tự là đường dẫn từ nút gốc đến nút lá đó.
  + Sang trái ta được bit 0.

+ Sang phải ta được bit 1.

- Có độ dài biến đổi (tối ưu bảng mã).
  + Các ký tự có tần suất lớn có độ dài ngắn.
  + Các ký tự có tần suất nhỏ có độ dài dài hơn.

c. Thuật toán 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.

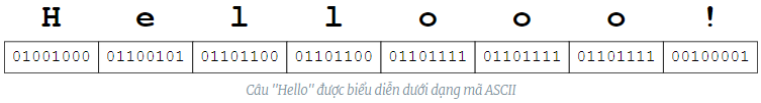
- 2.1: Tạo danh sách chứa các nút lá bao gồm phần tử đầu vào và trọng số nút là tần suất xuất hiện của nó. Sắp xếp các nút lá theo thứ tự giảm dần.
- 2.2: Từ danh sách này, lấy ra 2 phần tử có tần suất xuất hiện ít nhất. Sau đó gắn 2 nút vừa lấy ra vào một nút gốc mới với trọng số là tổng của 2 trọng số ở nút vừa lấy ra để tạo thành một cây.
- 2.3: Đẩy cây mới vào lại danh sách. Sau đó, sắp xếp lại thứ tự các nút lá.
- 2.4: Lặp lại bước 2 và 3 cho đến khi danh sách chỉ còn một nút gốc duy nhất của cây.
- 2.5: 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.

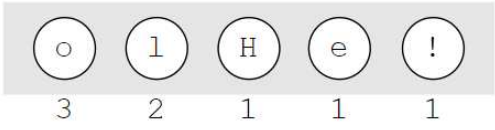
B5: Lưu lại cây Huffman (bảng mã) dùng cho việc giải nén. Xuất file nén.

d. Ví dụ



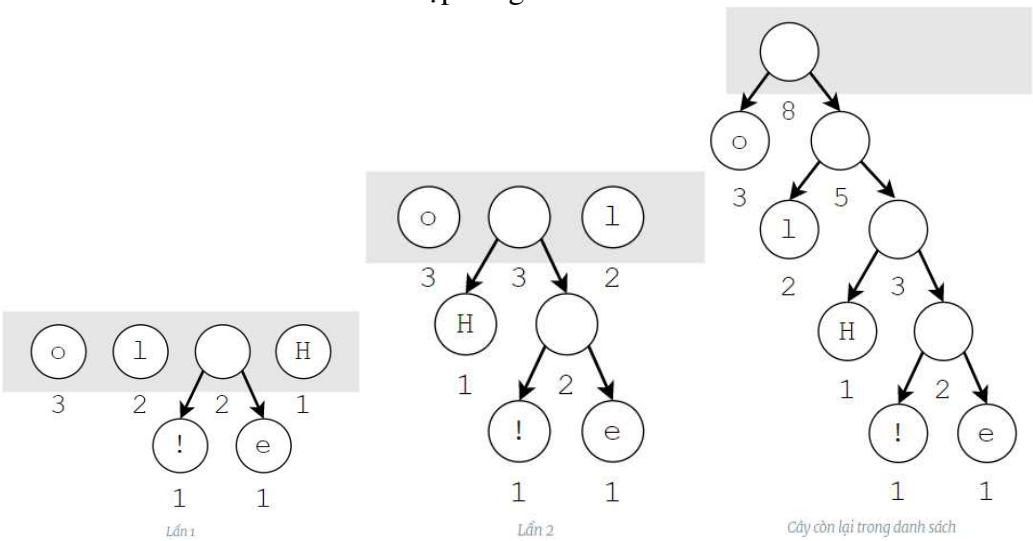
Câu "Hello" được biểu diễn dưới dạng mã ASCII

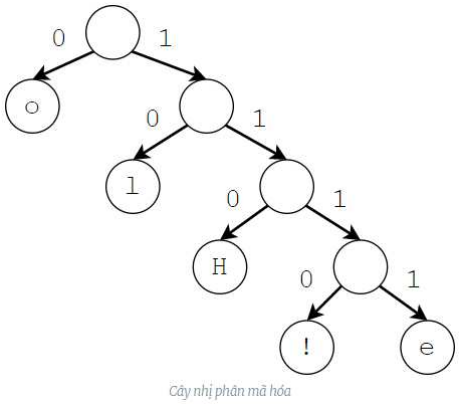
Chuỗi cần mã hóa



Danh sách ban đầu

Lập bảng tần suất





Cây nhị phân mã hóa

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

e. Thuật toán 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.

f. Ưu, nhược điểm của nén tĩnh

- **Ưu điểm:**
  + Hệ số nén tương đối cao.
  + Phương pháp thực hiện tương đối đơn giản.
  + Đòi hỏi ít bộ nhớ.
- **Nhược điểm:**
  + Mất 2 lần duyệt file khi nén.
  + Phải lưu trữ thông tin giải mã vào file nén.
  + Phải xây dựng lại cây Huffman khi giải nén.

5. Nén động (Adaptive Huffman)

a. Tính chất

- Tính chất anh em: Trọng số của nút bên trái nhỏ hơn nút bên phải và nhỏ hơn nút cha.
- Nút NYT (not yet transmitted) có trọng số luôn = 0, dùng để nhận biết ký tự đã xuất hiện trong cây hay chưa.
- Trọng số nút cha bằng tổng trọng số 2 nút con.

b. Thuật toán nén

B1: Duyệt tuần tự từng ký tự có trong file nhập.
- TH1: Nếu ký tự chưa tồn tại:
  + Chuỗi bit: đường dẫn đến NYT + Mã bit của ký tự.
  + Chèn nút mới (Ký tự | trọng số = 1) vào NYT. Đánh lại số thứ tự.
- TH2: Nếu ký tự đã tồn tại:
  + Chuỗi bit: đường dẫn đến ký tự đó.

+ Tăng trọng số của ký tự đó (+1).

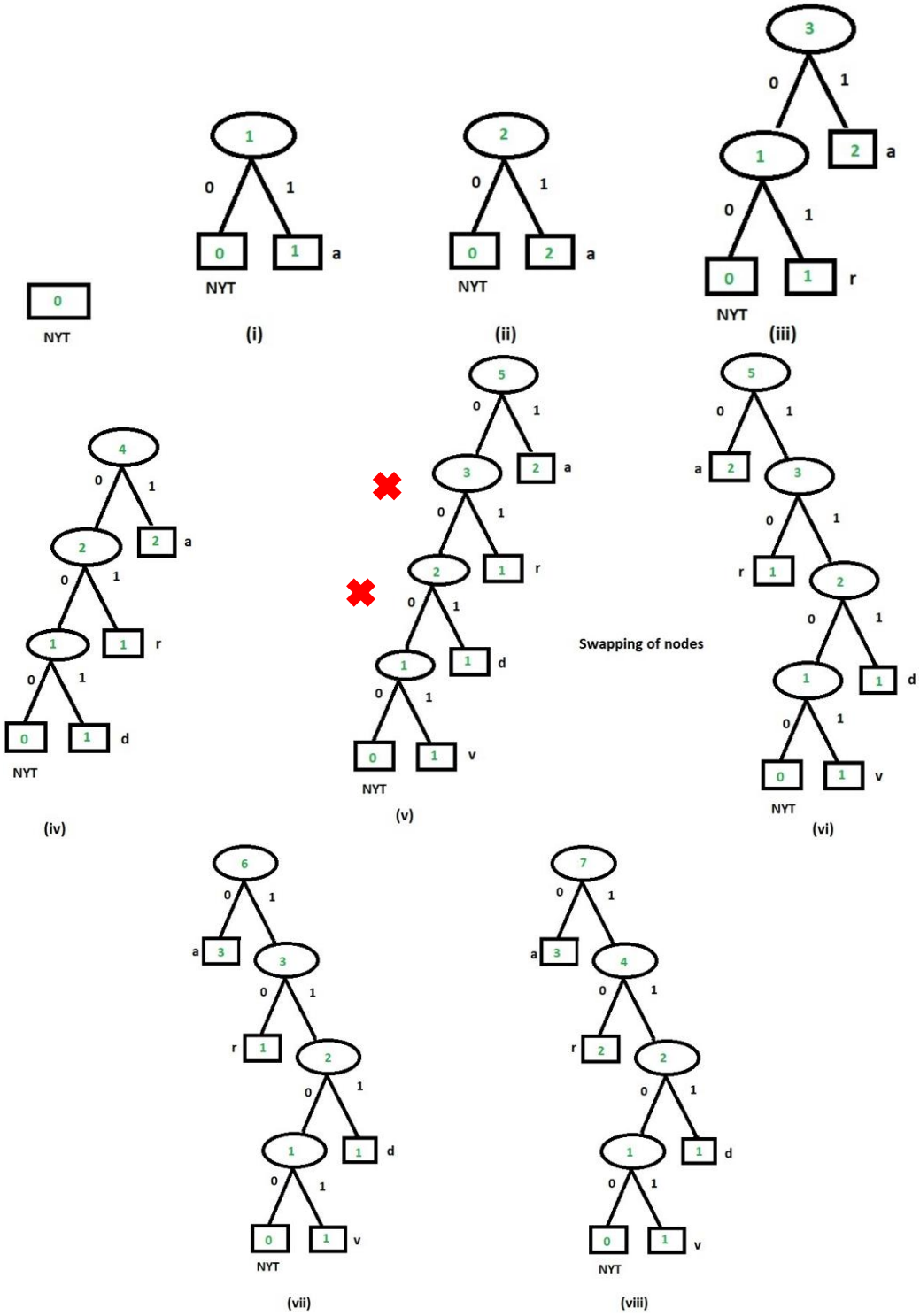
B2: Tăng trọng số của các nút cha (+1). Nếu vi phạm tính anh em thì điều chỉnh cho đến khi hết vi phạm.

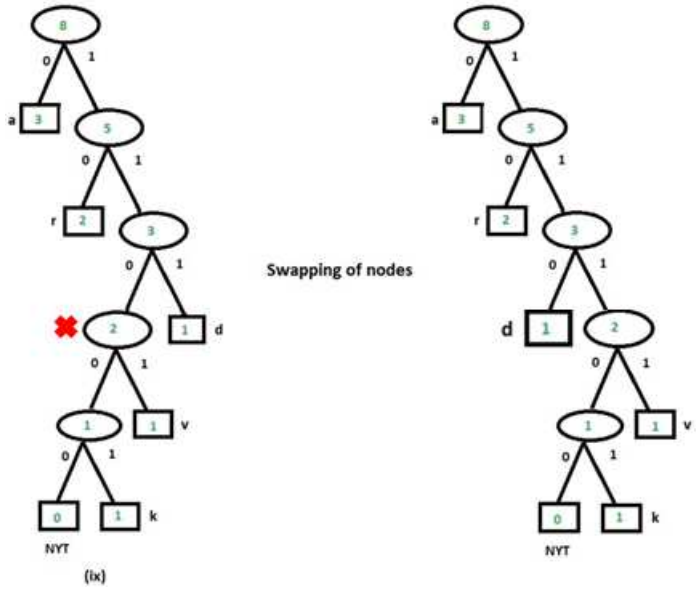
- Nếu trọng số nút hiện hành > nút lân cận từ phải sang trái, từ dưới lên trên thì vi phạm.
- Tìm nút xa nhất có trọng số cao nhất < trọng số nút vi phạm để hoán đổi vị trí.

B3: Lưu chuỗi bit vào file xuất. Lặp lại B1, B2 đến khi duyệt hết file.

c. Ví dụ

Chuỗi cần mã hóa: aardvark





Cây Huffman động hoàn chỉnh

d. Thuật toán giải nén

B1: Duyệt file, đọc lần lượt từng bit trong file nén và duyệt cây.

B2: Xuất ký tự tương ứng khi duyệt hết nút lá.

B3: Nếu gặp nút NYT, đọc 8bit tiếp theo. Xuất ký tự tương ứng. Cập nhật ký tự vừa xuất vào cây.

B4: Thực hiện B1, B2, B3 cho đến khi duyệt hết file.

B5: Xuất file đã giải nén.

e. Ưu điểm so với nén tĩnh

- Không cần phải lập bảng tần suất.
- Chỉ duyệt file 1 lần.
- Không cần lưu lại thông tin cho việc giải nén.
- Đầu đọc vừa duyệt, vừa cập nhật cây Huffman, vừa xuất kết quả ra file nén theo thời gian thực.

6. Ưu điểm và hạn chế của Huffman

- **Ưu điểm:**
  + Xử lý khá tốt độ dư thừa phân bố kí tự.
  + Quá trình mã hóa và giải mã tương đối đơn giản.
  + Cho mã có độ dài tối ưu.
- **Hạn chế:**
  + Giải quyết kém hiệu quả đối với loại độ dư thừa vị trí.
  + Tốn nhiều thời gian xây dựng cây mã.
  + Cấu trúc của cây mã hoặc bộ từ mã đã dùng để mã hóa phải được gởi đi cùng với số liệu đã được mã hóa. Điều này làm giảm hiệu suất nén.

IV. THUẬT TOÁN SHANNON-FANO

1. Tác giả

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. 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 $l_i=[-log_2\ p_i]$. 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.

3. Shannon

a. Thuật toán nén

B1: Sắp xếp các xác suất theo thứ tự **giảm dần**. Không mất tổng quát giả sử $p_1 \geq ... \geq p_K$.

B2: Định nghĩa $\boldsymbol{q_1} = 0$, $\boldsymbol{q_i} = \sum_{j=1}^{i-1} p_j \cdot 0$, $\forall\ i = 1, 2, ..., K$.

B3: Đổi $\boldsymbol{q_i}$ sang cơ số **2**, (biểu diễn q_i trong cơ số 2) sẽ được một chuỗi nhị phân.

B4: Từ mã được gán cho $\boldsymbol{a_i}$ là $\boldsymbol{l_i}$ kí hiệu lấy từ vị trí sau dấu phẩy của chuỗi nhị phân tương ứng với $\boldsymbol{q_i}$, trong đó $\boldsymbol{l_i} = [-\boldsymbol{log_2}\ \boldsymbol{p_i}]$.

b. Ví dụ

Mã hóa nguồn $S = \{a_1, a_2, a_3, a_4, a_5, a_6\}$ 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

4. Fano

a. Thuật toán nén

B1: Sắp xếp các xác suất theo thứ tự **giảm dần**. Không mất tổng quát giả sử $p_1 \geq ... \geq p_K$.

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.

b. Ví dụ

Mã hóa nguồn $S = \{a_1, a_2, a_3, a_4, a_5, a_6\}$ 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.*

5. Ưu 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.

V. THỰC NGHIỆM

Như đã nói ở phần mô tả bài toán, chúng tôi sẽ tiến hành đánh giá thuật toán Huffman và Shannon-Fano trên hai tác vụ là nén tệp văn bản và ảnh.

1. Dữ liệu

a. Nén tệp văn bản

Bộ dữ liệu gồm có **5** file ở định dạng *.txt có kích thước và chủ đề như bảng sau:

File	Kích thước (bytes)	Chủ đề
1.txt	715.332	Đoạn văn tiếng Anh.
2.txt	1.115.033	Dataset bán hàng trên Kaggle.
3.txt	251.705	Những bài văn tiếng Việt.
4.txt	78.674	Dữ liệu crawl được từ bài tập Lab1 và Lab2 của môn học này.
5.txt	2.167.737	Văn bản tiếng Anh.

b. Nén ảnh

Bộ dữ liệu gồm **5** ảnh màu ở định dạng *.jpg có kích thước và size như bảng sau:

File	Kích thước gốc (bytes)	Kích thước sau khi chuyển đổi thành mảng (bits)	Size
1.jpg	480.273	68.836.691	1200x600
2.jpg	114.644	59.081.540	549x960
3.jpg	2.137.052	213.530.513	1920x1080
4.jpg	431.489	227.492.125	1920x1080
5.jpg	120.999	59.284.232	691x777

2. Kết quả

a. Nén tệp văn bản

**H: Huffman, S-F: Shannon-Fano.*

File		Kích thước sau khi nén (bytes)	Tỷ số nén	Thời gian nén (giây)	Thời gian giải nén (giây)
1.txt	*H*	394.017	1,81549	0,38568	0,85217
	S-F	446.830 – 400.490	1,6009 – 1,78614	0,38883 – 0,39092	1,35034 – 1,22663
2.txt	*H*	558.512	1,99644	0,55016	1,19519
	S-F	610.344 – 571.243	1,82689 – 1,95194	0,56058 – 0,55436	1,81574 – 1,75459
3.txt	*H*	118.259	2,12842	0,12282	0,2659
	S-F	132.112 – 119.654	1,90524 – 2,10361	0,13127 – 0,12323	0,43577 – 0,37388
4.txt	*H*	52.576	1,49639	0,04489	0,10977
	S-F	56.792 – 53.839	1,3853 – 1,46128	0,05273 – 0,05597	0,19472 – 0,1689
5.txt	*H*	1.145.521	1,89236	1,15352	2,38875
	S-F	1.232.335 – 1.180.838	1,75905 – 1,83576	1,12055 – 1,1209	3,69857 – 3,62304

**Source code: [Link].*

b. Nén ảnh

File		Kích thước sau khi nén (bits)	Tỷ số nén	Thời gian nén (giây)	Thời gian giải nén (giây)
1.jpg	*H*	32.946.601	2,08934	21,11621	87,65538
	S-F	43.480.367 – 35.589.154	1,58317 – 1,9342	22,05799 – 23,11391	121,25229 – 94,00442
2.jpg	*H*	29.634.697	1,99366	18,30662	80,19416
	S-F	41.422.532 – 31.131.113	1,42631 – 1,89783	20,11017 – 19,55467	107,30088 – 82,86994
3.jpg	*H*	106.141.666	2,01175	74,42264	301,52266
	S-F	148.499.611 – 109.822.600	1,43792 – 1,94432	76,21462 – 78,42185	388,5325 – 290,18915
4.jpg	*H*	113.470.111	2,00486	79,82168	322,04064
	S-F	158.240.326 – 117.531.798	1,43764 – 1,93558	75,68447 – 79,59545	395,44557 – 383,99727
5.jpg	*H*	29.722.538	1,99459	18,04945	82,94047
	S-F	42.015.277 – 30.688.090	1,41102 – 1,93183	21,25767 – 19,73568	104,35448 – 77,21478

**Source code: [Link].*

3. Kết luận

- Từ kết quả nhận được thông qua quá trình thực nghiệm trên cả hai tác vụ nén tệp văn bản và nén ảnh, ta có thể nhận thấy Huffman tỏ ra vượt trội so với Shannon-Fano trên bộ dữ liệu mà chúng tôi đã chuẩn bị.
- Huffman không chỉ có tỷ số nén cao hơn mà còn có thời gian nén lẫn thời gian giải nén ngắn hơn Shannon-Fano.
- Ở tác vụ nén tệp văn bản, ta có thể thấy sự khác biệt nhưng vẫn chưa đủ rõ ràng. Cho đến khi tác vụ nén ảnh được đưa vào thực nghiệm thì Shannon thực sự thua kém Huffman một cách rõ rệt và chỉ còn Fano là đủ sức cạnh tranh với Huffman.
- Ngoài ra, trong quá trình thực nghiệm, phương pháp Fano cho kết quả tốt hơn phương pháp Shannon. Trên cùng một tệp văn bản hay ảnh, qua mỗi lần nén thì thuật toán Shannon-Fano lại cho ra một tỷ số nén khác nhau (trong khi Huffman thì không gặp hiện tượng này), điều đó nói lên việc Shannon-Fano chưa thực sự cho ra mã tối ưu và cố định.

4. So sánh

- Huffman:
  + Hiệu quả và tối ưu.
  + Độ dài từ dự kiến $H(X) + 1 - p_{min}$.
  + Xây dựng cây nhị phân Bottom up.

- Shannon-Fano:
  + Kết quả tạo ra không tối ưu.
  + Độ dài từ dự kiến $H(X) + 1$.
  + Xây dựng cây nhị phân Top down.

**H(X) là entropy thông tin ngẫu nhiên rời rạc, được tính theo công thức:*

$$H(X) = -\sum p_i \log_2 p_i, \text{ với } p_i \text{ là xác suất của } i.$$

VI. TÀI LIỆU THAM KHẢO

[1] sudhamshu091. (2020) github.com. [Online]. https://github.com/sudhamshu091/Huffman-Encoding-Decoding-For-Image-Compression

[2] sudhamshu091. (2020) github.com. [Online]. https://github.com/sudhamshu091/Shannon-Fano-Encoding-and-Decoding-for-Image-Compression

[3] Mèo lười. 123docz.net. [Online]. https://123docz.net//document/326278-nen-va-xu-ly-du-lieu.htm

[4] Wikipedia. [Online]. https://vi.wikipedia.org/wiki/Mã_hóa_Huffman

[5] Wikipedia. [Online]. https://en.wikipedia.org/wiki/Shannon–Fano_coding

[6] Wikipedia. [Online]. https://vi.wikipedia.org/wiki/Entropy_thông_tin#:~:text=Entropy%20thông%20tin%20là%20một,một%20sự%20kiện%20ngẫu%20nhiên.

[7] "Mã hóa Fano-Shannon".

[8] "Bài thảo luận môn Lý thuyết thông tin".

[9] Khoa công nghệ thông tin, "Bài 7 Mã hóa tối ưu nguồn rời rạc không nhớ".

[10] "Đề cương bài giảng".