Mục lục

Mở đầu

Chúng ta bước vào một thời kỳ phát triển mới, đó là sự kết nối tri thức toàn cầu. Từng phút, từng giây nhiều tỷ tỷ bit dữ liệu đang được luân chuyển trên mạng máy tính, và trong tương lai dung lượng thông tin trung chuyển còn tăng nhanh và lớn đến mức mà chúng ta khó lòng mà mường tượng nổi. Dòng tin lớn sẽ dẫn đến việc tắc nghẽn giao thông trên mạng, hơn thế thời gian cũng như chi phí chuyển tải, lưu trữ tin tăng cao làm cho hiệu quả kinh tế giảm sút. Đứng trước thực tế này, người ta có thể đề ra nhiều giải pháp để tháo gỡ khó khăn, ví dụ như việc nâng cấp hệ thống mạng thông tin, hay là việc quy hoạch toàn cầu... Bên cạnh các giải pháp này chúng ta luôn có một giải pháp, đó là nén dữ liệu lại. Về mặt khoa học, nén dữ liệu không chỉ đơn thuần vì lý do kinh tế mà còn để đảm bảo cho một hệ thống xã hội cho dù lớn đến mức nào đi chăng nữa thì thông tin vẫn thông chuyển được.

Mục tiêu của luận văn này nhằm hệ thống các kiến thức về nén văn bản thông qua minh họa cụ thể và lý thuyết xác suất, từ đó đưa ra giới hạn nén của một văn bản.

Nhiệm vụ của luận văn là:

- Phân loại văn bản, đưa ra mô hình biểu diễn văn bản, nghiên cứu giới hạn nén của văn bản và kiểm tra lại lý thuyết nén văn bản bằng chương trình.
- Nghiên cứu một số mã nén, giải thuật nén và giải nén văn bản.

Phạm vi nghiên cứu: Nghiên cứu nén văn bản dựa trên mô hình Markov hiện và nén bảo toàn văn bản.

Phương pháp nghiên cứu là :

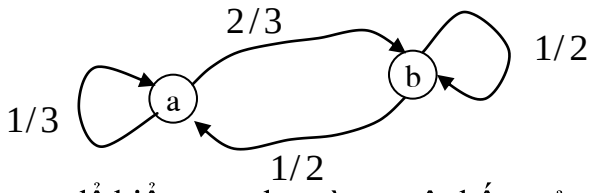
- Sử dụng lý thuyết xác suất nhằm đưa ra quy trình nén văn bản.
- Sử dụng phương pháp nghiên cứu thực nghiệm mô phỏng một file văn bản theo mô hình Markov và kiểm chứng tính đúng đắn của lý thuyết bằng chương trình. Cụ thể đưa ra một số trình ví dụ cho phép tạo ra các văn bản dựa theo mô hình Markov, và tính được tỷ lệ nén theo lý thuyết nén văn bản, có chạy trình winrar để kiểm tra tính đúng đắn của lý thuyết.
- Sử dụng công cụ lập trình triển khai các phương pháp nén văn bản dựa trên mô hình Markov.

Nội dung luận văn gồm 4 chương:

Chương 1. Văn bản và các định lý về nén văn bản

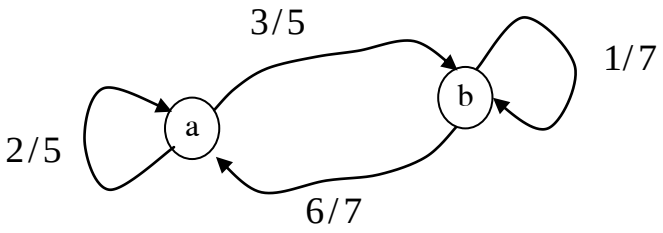
Chương này trình bày về khái niệm văn bản, bit trung bình, entropy, định lý về nén văn bản tổng quát, mô hình Markov để biểu diễn văn bản, phân bố ổn định, cách tính entropy của mô hình Markov, các nguồn cùng xác xuất nhưng khác Entropy, nguồn có entropy nhỏ nhất và định lý nén văn bản theo mô hình Markov, từ đó đưa ra giới hạn nén một văn bản. Cuối cùng là các trình ví dụ dùng để tạo ra văn bản theo mô hình Markov và tính tỷ lệ nén văn bản. Trong đó:

- Ví dụ 1.5. Trình tạo ra file văn bản một cách ngẫu nhiên từ các chữ cái a và b, với xác suất tương ứng p1 = 2/3, p2 = 1/3, có dung lượng 64000b. Theo lý thuyết ta có $E = 2/3 \log_2(3/2) + 1/3 \log_2(3) \approx 0.918$. Sau khi nén còn ≈ 11%. Dùng Winrar để kiểm tra cho cùng một kết quả. (trang 19)
- Ví dụ 1.6. Trình tạo ra file văn bản theo mô hình Markov, có dung lượng 64000b. File nén theo lý thuyết có dung lượng bằng 12%. (trang 20)



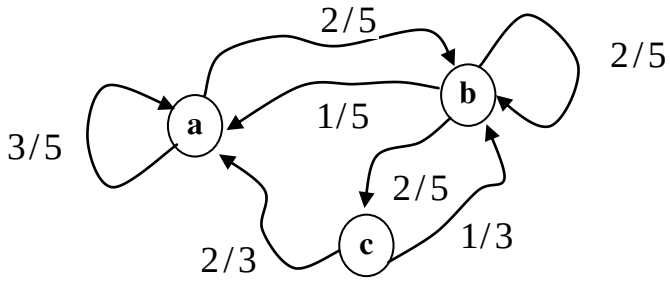
Dùng Winrar để kiểm tra cho cùng một kết quả.

- Ví dụ 1.7. Trình tạo ra file văn bản theo mô hình Markov, có dung lượng 64000b. File nén theo lý thuyết có dung lượng bằng 10%. (trang 22)



Dùng Winrar để kiểm tra cho cùng một kết quả.

- Ví dụ 1.8. Trình tạo ra file văn bản theo mô hình Markov, có dung lượng 640000b. File nén theo lý thuyết có dung lượng bằng 15%. (trang 25)



Dùng Winrar để kiểm tra cho cùng một kết quả.

Chương 2. Các mã nén và thuật toán nén văn bản cổ điển

Với các mã nén văn bản cổ điển, mỗi chữ cái của bảng chữ cái được biểu diễn bằng một xâu bit trong đó không có xâu nào là đoạn đầu của xâu kia và chữ cái nào có xác suất xuất hiện lớn hơn thì được biểu diễn bằng xâu bit có

độ dài ngắn hơn, chữ cái nào có xác suất xuất hiện nhỏ thì được biểu diễn bằng xâu bit có độ dài dài hơn.

Chương này trình bày về khái niệm mã tổng, mã phân tách, mã tối ưu và chỉ ra sự tồn tại của mã tối ưu, định lý về bit trung bình của mỗi chữ cái của hầu hết các văn bản và bit trung bình của mã, định lý về điều kiện đủ để giải mã được một dãy bit được tạo bởi một mã tổng từ một bảng mã bit "0/1" có độ dài thay đổi , định lý Kraft - Mc Milan về điều kiện cần và đủ để có mã tổng các chữ cái bằng xâu bit 0/1, đồng thời đưa ra các mã nén văn bản cổ điển và giải thuật nén tương ứng, cuối mỗi phần có trình minh họa cho cách nén theo mỗi giải thuật. Cụ thể gồm các mã nén Shanon, mã Fano, mã Huffman tĩnh, mã Huffman động.

Chương 3. Mã số học

Mã số học biểu diễn mỗi văn bản bằng một số thực nằm trong nửa đoạn [0,1) sao cho số thực ứng với mỗi văn bản có số chữ số có nghĩa là ít nhất. Văn bản càng lớn ứng với số thực càng nhỏ.

Chương này trình bày về biểu diễn nguồn nói chung và biểu diễn nguồn cho mô hình Markov, mã số học với số nguyên, thuật toán nén và giải nén văn bản bằng mã số học và trình minh họa cho mã số học.

Chương 4. Mã LZW

Đối với mã LZW, thay vì mã hóa từng ký tự của bảng chữ cái nó đi mã hóa từng móc xích và sử dụng kỹ thuật từ điển động. Trong đó, từ điển được thành lập trong quá trình mã và giải mã.

Chương này trình bày về nguyên lý mã theo từ điển (nguyên lý LZ), từ điển tĩnh, từ điển động, khái quát hóa về thuật toán LZ, các công đoạn thực hiện khi mã bằng LZ và cuối cùng là trình bày về mã LZW (loại mã hay dùng hiện nay), thuật toán nén bằng giải nén bằng mã LZW và trình minh họa.

Tôi xin trân trọng cảm ơn tất cả các thầy cô giáo trong khoa CNTT và bạn bè, đồng nghiệp đã giúp đỡ tôi hoàn thành luận văn này.

Hải Phòng, tháng 7 năm 2009

Chương 1. Văn bản và các định lý về nén văn bản

1.1. Văn bản và nén văn bản

- **Bảng chữ cái** là một tập hợp $\Omega=\{a_1,a_2,....,a_m\}$. Mỗi phần tử a_i của nó được gọi là chữ cái hay kí tự. Nếu bảng chữ chỉ có 2 chữ cái thì gọi các chữ cái là bit và kí hiệu là 0/1.
- **Văn bản** là một dãy nào đó gồm các chữ của một bảng chữ cái. Số lượng các chữ cái được gọi là độ dài của văn bản.
- Nếu có ánh xạ $f:A\rightarrow B$ tương ứng 1-1 giữa hai tập A và B các văn bản thì ta nói là tồn tại ánh xạ mã hoá văn bản A thành B. Nếu B là các văn bản được tạo ra từ các bit "0/1" thì ta gọi loại mã này là mã nhị phân và gọi tắt B là "bản mã", còn "văn bản" được ngầm hiểu là dùng để chỉ A.

Người ta thường ký mã thông qua các từ của một bảng chữ cái nào đó và lưu chúng lại trên các thiết bị vật lý. Trong số các cách mã thì cách nào ký mã ngắn hơn ta nói là nó nén tin tốt hơn (so với cách mã khác.)

Thường ngày ta hay dùng trình nén để nén các file, tức là các văn bản tạo ra từ 256 byte. Nén một file nhiều lần liên tiếp thì sớm hay muộn ta cũng sẽ thu được một file mà trình nén này không thể thu nhỏ lại được nữa, bởi nếu không ta sẽ nén được file ấy xuống thành 1 file không có bit nào cả.

Với mọi thuật toán mã các file văn bản luôn tồn tại một văn bản mà nó không thể nén được thành file có dung lượng nhỏ hơn.

Từ khẳng định trên suy ra không thể vạch định ra được một gianh giới rõ ràng giữa một bên là mã hoá văn bản và một bên là mã nén. Để đánh giá khả năng nén của một thuật toán ta đưa ra khái niệm về số bit trung bình cần thiết để ghi lại một chữ cái của văn bản.

- **Định nghĩa 1.1:** Tỷ số giữa độ dài của bản mã chia cho số các chữ cái của văn bản được gọi là bit trung bình cho một chữ cái của văn bản, hay gọi tắt là bit trung bình (hay bit trung bình cho từng chữ cái).

- **Định nghĩa 1.2** : Kí hiệu A_n là tập các văn bản có độ dài n tạo ra từ các chữ cái $a_1,a_2,...,a_m$. Giả sử ta có một mã nào đó mà văn bản $\zeta \in A_n$ có bản mã dài $L(\zeta)$ bit. Khi đấy ta gọi bít trung bình của mã là giá trị $\frac{\sum_{\zeta \in A_n} p(\zeta)L(\zeta)}{n}$.

Vấn đề đặt ra là làm thế nào để biết được $p(\zeta)$ - xác suất xuất hiện văn bản ζ. Về nguyên tắc thì xác suất này là phụ thuộc vào người sử dụng văn bản. Văn bản nào hay được dùng hơn thì có xác suất xuất hiện lớn hơn, văn bản nào ít được dùng hơn thì có xác suất xuất hiện nhỏ hơn. Như vậy định nghĩa bao hàm ý tưởng, để có thể nén được tốt hơn thì một văn bản cần phải được mã nén không phụ thuộc vào văn bản ấy dài hay ngắn mà là phụ thuộc theo xác suất mà người ta sử dụng nó. Tuy nhiên có một thực tế là phần lớn các văn bản lưu trữ trong kho rất ít khi được sử dụng. Như vậy ta khó lòng xác định được xác suất sử dụng của các văn bản một khi chúng chưa hề hoặc rất ít khi được sử dụng. Nhu cầu nén văn bản buộc ta phải suy nghĩ đến vấn đề này dưới góc độ khác hơn. Việc một văn bản được sử dụng như thế nào, nhiều hay ít phụ thuộc vào nội dung của văn bản. Như vậy ta cần tìm cách làm thế nào đánh giá được xác suất xuất hiện văn bản thông qua ngay chính nội dung của nó.

Một văn bản có thể do nhiều nguồn sinh ra. Căn cứ vào sự phụ thuộc tin, ta có thể phân văn bản thành hai loại, một loại là mô hình rời rạc (không phụ thuộc) tức là mô hình mà xác suất xuất hiện các chữ cái của văn bản được chọn một cách ngẫu nhiên trong một bảng chữ cái, một loại là mô hình phụ thuộc tức là mô hình mà xác suất xuất hiện một chữ cái chỉ phụ thuộc vào quá khứ và có thể mô tả thông qua mô hình Markov.

1.2. Định lý về nén văn bản tổng quát

Cho bảng chữ cái $\Omega=\{a_1,a_2,....,a_m\}$ với xác suất xuất hiện của các chữ cái tương ứng là $p_1=p(a_1)$, $p_2=p(a_2)$,..., $p_m=p(a_m)$.

Nếu văn bản $\zeta= \omega_1\omega_2...\omega_n$ được sinh ra từ việc chọn ngẫu nhiên các chữ cái thì sẽ có xác suất xuất hiện là $p(\zeta)= p(\omega_1)\ p(\omega_2)...\ p(\omega_n)$.

Nén văn bản không phải là việc các văn bản bị ghi nén lại. Bản chất của các thuật toán nén văn bản là ghi lại văn bản (mã lại văn bản) ở dạng khác. Xuất hiện hai câu hỏi. Câu hỏi thứ nhất có thể nén văn bản trên nhỏ đến bao nhiêu cũng được không hay là có một giới hạn nhất định nào đó mà ta không thể vượt qua được. Câu hỏi thứ hai có hay không một thuật toán nén tốt nhất.

Điều kiện đầu tiên để nén được văn bản là các văn bản khác nhau thì có các file nén khác nhau. Bởi nếu không thì ta không thể khôi phục lại văn bản nguồn. Mọi văn bản không thể nén lại thành một file chỉ có 1 bit vì số lượng các file có 1 bit là 2. Một qui trình nén như vậy thì chỉ có thể dùng để nén 2 văn bản mà thôi đến văn bản thứ 3 là nội dung của file nén sẽ bị trùng lặp. Vậy thì không thể nén một văn bản nhỏ tùy ý được. Giới hạn nén của một văn bản là bao nhiêu? Shannon là người đầu tiên chứng minh được sự tồn tại một giới hạn nén cho mỗi văn bản. Một văn bản thực ra chỉ có thể nén đến một giới hạn nhất định, giới hạn ấy gọi là lượng tin của văn bản. Lượng tin chỉ phụ thuộc vào bản thân văn bản chứ không phụ thuộc vào thuật toán nào. Mọi

thuật toán đều không thể nén một văn bản đến một file nhỏ hơn lượng tin mà văn bản có. Lượng tin còn được gọi là entropy.

Đối với văn bản được sinh ra từ mô hình rời rạc thì

$$\text{entropy} = \sum_{\mathbf{i}=1}^{\mathbf{m}} \mathbf{p_i} \log_2 \frac{1}{\mathbf{p_i}}$$

- **Định lý Shannon** Xét các văn bản được tạo ra theo cách chọn ngẫu nhiên các chữ cái của bảng chữ cái $\Omega=\{a_1, a_2, ..., a_m\}$ với xác suất xuất hiện tương ứng $p_1 \geq p_2 \geq ... \geq p_m > 0$.

1. Với mọi mã nhị phân

(a) Bit trung bình của mã thoả mãn $\frac{1}{n} \sum_{\zeta \in A_n} p(\zeta) L(\zeta) \geq \sum_{\mathbf{i}=1}^{\mathbf{m}} \mathbf{p_i} \log_2 \frac{1}{\mathbf{p_i}}$

(b) Với hầu hết các văn bản bit trung bình (cho một chữ cái) của văn bản không nhỏ hơn $\sum_{i=1}^{m} p_i \log_2 \frac{1}{p_i}$

2. Tồn tại mã nhị phân cho từng khối k chữ cái có tính phân tách sao cho bit trung bình (cho một chữ cái) của nó nằm giữa $\sum_{i=1}^{m} p_i \log_2 \frac{1}{p_i}$ và $\frac{1}{k} + \sum_{i=1}^{m} p_i \log_2 \frac{1}{p_i}$.

Như vậy, định lý khẳng định rằng 'entropy đúng là giới hạn nhỏ nhất có thể mà bit trung bình của một mã nén nhị phân có thể đạt được' cho dù mã được tạo ra theo bất cứ cách nào.

(định lý đã được chứng minh trong tài liệu lý thuyết mã nén của nhóm tác giả: Nguyễn Lê Anh, Trần Duy Lai, Phạm Thế Long, Nguyễn Văn Xuất).

Ví dụ 1.1. Văn bản

adbadacbdcbacbdbacbacdcdacbadacbdba
cbacbacdbadacbacbacbadacbacbacbadcd
bacbadbacdbdcbacdacbacbacbacdda

Có tất cả 30 chữ ‘a’, 26 chữ ‘b’, 26 chữ ‘c’ và 19 chữ ‘d’ được sinh ra một cách ngẫu nhiên.

Entropy=1.98

$$\text{entropy}=-(\frac{30}{101}\log_2\frac{30}{101}+\frac{26}{101}\log_2\frac{26}{101}+\frac{26}{101}\log_2\frac{26}{101}+\frac{19}{101}\log_2\frac{19}{101})=1.98$$

Tuy nhiên, văn bản do con người tạo ra không phải các chữ cái xuất hiện nột cách ngẫu nhiên, đương nhiên là phụ thuộc lẫn nhau tuân thủ theo các qui tắc tạo từ, tạo câu, ... Để nghiên cứu vấn đề này ta xét mô hình Markov là mô hình do A. A. Markov (1856-1922) đưa ra.

1.3. Mô hình Markov (trạng thái).

1.3.1. Định nghĩa mô hình Markov (trạng thái).

- **Định nghĩa đồ thị định hướng.** Đồ thị định hướng bao gồm một tập hợp hữu hạn các đỉnh - trạng thái, $S=\{S_1, S_2, ..., S_m\}$ và các cạnh định hướng $\Omega=\{a_1,a_2...a_l\}$.
- **Định nghĩa mô hình Markov (trạng thái).** Mô hình Markov là một đồ thị định hướng. Mỗi cạnh có xác xuất di chuyển theo cạnh. Tổng các xác suất chuyển trạng thái ra khỏi một đỉnh bất kỳ của đồ thị luôn bằng 1.
- **Một văn bản do một mô hình Markov sinh ra.** Mỗi một tiến trình được xác định duy nhất thông qua các đỉnh và các cạnh mà nó đi qua. Xác suất xuất hiện của một tiến trình là tích của các xác suất dọc theo các cạnh mà tiến trình đi qua. Số các đỉnh của một tiến trình tương ứng tỷ lệ với số các cạnh mà tiến trình đi qua. Văn bản của một tiến trình là dãy các chữ cái tên của đỉnh đầu tiên và các cạnh mà một tiến trình đi qua.

- Nếu có không quá 1 cạnh nối từ đỉnh này tới đỉnh kia thì mỗi tiến trình được xác định duy nhất bởi các đỉnh mà nó đi qua. Khi ấy văn bản của một tiến trình tương ứng duy nhất với dãy tên của các đỉnh mà tiến trình đi qua.

- Nếu chỉ quan tâm đến các đỉnh, ví dụ như tần suất viếng thăm các đỉnh chẳng hạn thì ta có thể gộp các cạnh cùng nối từ đỉnh này tới đỉnh kia lại để mô hình trở thành trường hợp mà từ đỉnh này tới đỉnh kia được nối bởi không quá 1 cạnh.

Gọi p_{ij} với i, j = 1.. m là xác suất di chuyển từ đỉnh A_i tới đỉnh A_j dọc theo tất cả các cạnh nối. Mỗi cạnh đi từ đỉnh A_i tới đỉnh A_j có một trọng số là xác suất chuyển động dọc theo cung đó. Giá trị p_{ij} được tính bằng tổng tất cả các trọng số của các cạnh đi từ đỉnh A_i tới đỉnh A_j. Ma trận F tạo ra từ các p_{ij} là ma trận vuông cấp m. Ma trận xác suất chuyển là một ma trận thống kê với các tính chất sau:

Các phần tử của nó không âm: $p_{ij} \geq 0$

Tổng các phần tử của mỗi cột bằng 1: $\sum_{j=1}^{m} p_{ij} = 1$. Do $\sum_{j=1}^{m} p_{ij}$ bằng tổng các trọng số đi ra từ đỉnh thứ i (theo tối đa là l cạnh) nên nó bằng 1.

Do tổng các xác suất thoát khỏi một đỉnh bất kỳ bằng 1 cho nên ma trận F có tính chất là tổng của các số của một cột bất kỳ luôn bằng 1. Ma trận như thế nhận λ=1 làm giá trị riêng.

Nếu tại thời điểm nào đó xác suất xuất hiện tại các đỉnh tương ứng là P thì tại thời điểm tiếp theo xác suất gặp các đỉnh đó là FP. Ta thấy rằng có thể áp dụng lý thuyết của xích Markov cho mô hình Markov. Ký hiệu $\xi = (\xi_k, P, F)$ là xích Markov thuần nhất (ma trận xác suất chuyển không phụ thuộc vào thời gian) có m trạng thái với phân bố xác suất ban đầu là vector

dòng $P=(p_i)$ và ma trận xác suất chuyển là $F=\|p_{ij}\|$. Nếu ta qui định đối với mô hình Markov luôn có đỉnh xuất phát thì P = (1,0,0,..,0).

Ta ký hiệu $p_{ij}^{(k)} = P\{\xi_k = j \mid \xi_0 = i\}$, đó là xác suất chuyển sau k bước từ trạng thái i sang trạng thái j, đó chính là các phần tử của ma trận F^k. Khi đó có phương trình Kolmogorov sau: $p_{ij}^{(k+l)} = \sum_{\alpha} p_{i\alpha}^{(k)} p_{\alpha j}^{(l)}$.

- **Định nghĩa Egordic.** Mô hình Markov có tính egordic nếu như sau một số bước đủ lớn, xuất phát từ một đỉnh ta có thể đến được tất cả các đỉnh khác với xác suất lớn hơn 0.

Trong ngôn ngữ của ma trận xác suất chuyển thì điều kiện ergodic chính là: tồn tại số n_0 sao cho $\varepsilon = \min_{i,j} p_{ij}^{n_0} > 0$.

Dưới quan điểm của lý thuyết đồ thị thì điều kiện ergodic chính là: có thể chuyển từ một đỉnh bất kỳ đến tất cả các đỉnh trong đồ thị theo các cạnh có định hướng. Đó chính là tính liên thông của đồ thị.

Một điều cần chú ý là đồ thị của mô hình Markov có m đỉnh. Nhưng các chữ cái đi kèm với một cạnh lại thuộc một bảng chữ cái có **n** chữ. Nối 2 đỉnh có thể có các cạnh bội ứng với các chữ cái khác nhau nên **n** có thể lớn hơn m. Khi ta nói chú châu chấu nhảy từ một đỉnh này sang một đỉnh khác thì có nghĩa là nó di chuyển theo một trong các cạnh nối 2 đỉnh ấy.

1.3.2. Phân bố ổn định

Xét mô hình Markov ergodic.

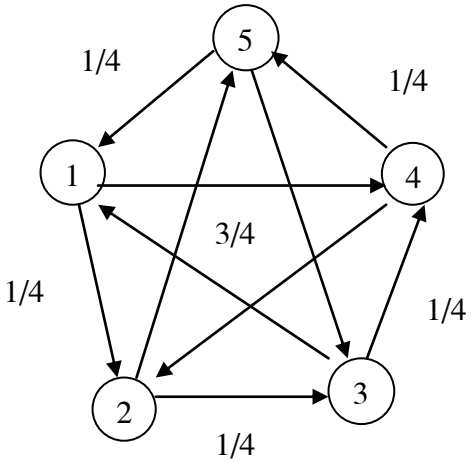
- Định lý 1.1**.** Đối với mô hình ergodic với mọi phân bố xác suất ban đầu $P=\{p_i\}$, thì dãy FP, F^2P, F^3P,... tiến đến một phân bố duy nhất - phân bố ổn

định $\Pi = \lim_{n\to\infty} F^n P$. Phân bố này là nghiệm của phương trình FΠ=Π với điều kiện $\sum_{i=1}^{m} \pi_i = 1$.

(định lý đã được chứng minh trong tài liệu lý thuyết mã nén của nhóm tác giả: Nguyễn Lê Anh, Trần Duy Lai, Phạm Thế Long, Nguyễn Văn Xuất trang 133).

Ví dụ 1.2. Giải phương trình tìm điểm bất động với điềukiện $\sum_{i=1}^{5} \pi_i = 1$.

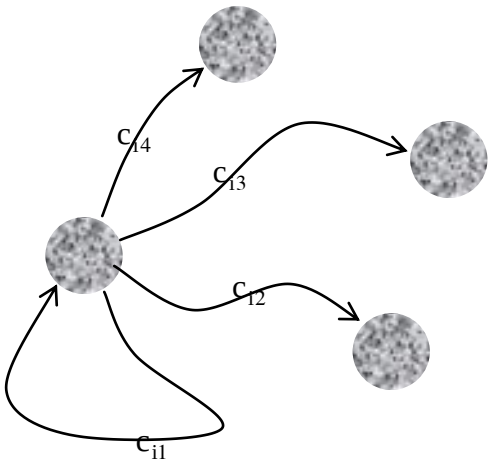
$$\begin{pmatrix} 0 & 0.25 & 0 & 0.75 & 0 \\ 0 & 0 & 0.25 & 0 & 0.75 \\ 0.75 & 0 & 0 & 0.25 & 0 \\ 0 & 0.75 & 0 & 0 & 0.25 \\ 0.25 & 0 & 0.75 & 0 & 0 \end{pmatrix} \begin{pmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \\ \pi_5 \end{pmatrix} = \begin{pmatrix} \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \\ \pi_5 \end{pmatrix}$$



Hình 1.1

tìm được nghiệm duy nhất $\pi_1=\pi_2=\pi_3=\pi_4=\pi_5=\frac{1}{5}$ là phân bố ổn định của mô hình.

1.3.3. Entropy.



Hình 1.2

Ký hiệu các đỉnh của mô hình là $\{A_1, A_2,...,A_m\}$, các cạnh đi ra từ đỉnh A_i là c_{ij}(trong đó j=1,2,.., m_i), phân bố ổn định là $\Pi=\{\pi_1, \pi_2,..., \pi_m\}$, trọng số các cạnh đi ra từ đỉnh A_i là w_{ij}(lưu ý j=1,2,..,m_i). Giá trị $E_i = \sum_{j=1}^{m_i} w_{ij} \log_2 \frac{1}{w_{ij}}$ được gọi là entropy của đỉnh A_i. Giá trị $H=\sum_{i=1}^{m} \pi_i E_i = \sum_{i=1}^{m} \pi_i \sum_{j=1}^{m_i} w_{ij} \log_2 \frac{1}{w_{ij}}$ được gọi là entropy của mô hình.

- **Định lý 1.2** Xét các văn bản được tạo ra từ mô hình Markov.

1. Với mọi mã nhị phân

(a) Với n đủ lớn, bit trung bình của mã không nhỏ hơn entropy.

$$\frac{\sum_{\zeta \in A_n} p(\zeta) L(\zeta)}{n} \geq \sum_{i=1}^{m} \pi_i \sum_{j=1}^{m_i} w_{ij} \log_2 \frac{1}{w_{ij}}.$$

(b) Bit trung bình (cho một chữ cái) của hầu hết các văn bản không nhỏ hơn entropy.

2. Với mọi giá trị ε>0 nhỏ tuỳ ý, luôn chỉ ra được mã nhị phân, mà khi văn bản đủ dài bit trung bình của mã và của hầu hết các văn bản, nằm trong khoảng entropy và entropy+ε

(định lý đã được chứng minh trong tài liệu lý thuyết mã nén của nhóm tác giả: Nguyễn Lê Anh, Trần Duy Lai, Phạm Thế Long, Nguyễn Văn Xuất trang 146).

Như vậy ta có

- **Định lý 1.3.** Với hầu hết các văn bản ξ thì $\lim_{n\to\infty} -\frac{\log_2 p(\xi)}{n} = \text{entropy}$.

1.3.4. Các nguồn cùng xác suất khác entropy.

Bài toán mô hình hoá một nguồn tin trên thực tế là một bài toán khó. Một luồng tin hữu hạn có thể do nhiều nguồn tin sinh ra.

Ví dụ 1.3. Văn bản

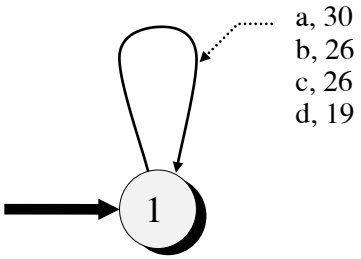
adbadacbdcbacbdbacbacdcdacbadacbdba
cbacbacdbadacbacbacbadacbacbacbadcd
bacbadbacdbdcbacdacbacbacbacdda

Có tất cả 30 chữ ‘a’, 26 chữ ‘b’, 26 chữ ‘c’ và 19 chữ ‘d’. Có thể coi như luồng tin được sinh ra từ các nguồn sau.

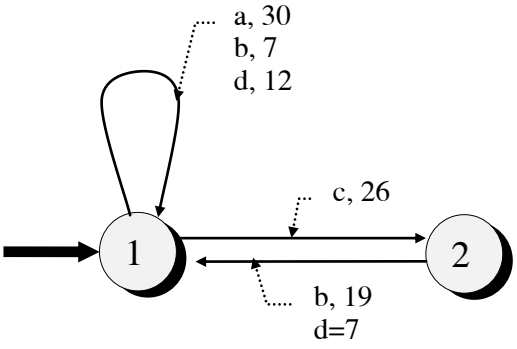
Nguồn 1. Entropy=1.98

$$\text{entropy}=-(\frac{30}{101}\log_2\frac{30}{101}+\frac{26}{101}\log_2\frac{26}{101}+\frac{26}{101}\log_2\frac{26}{101}+\frac{19}{101}\log_2\frac{19}{101})=1.98$$

Trong mô hình nguồn trên ta hầu như chỉ chú ý đến xác suất xuất hiện của các chữ “a, b, c, d”. Tuy nhiên ta có thể nhận thấy mô hình sau đây là thích hợp hơn với văn bản trên, để ý rằng sau khi xuất hiện chữ c thì không thể xuất hiện chữ a.



Nguồn 1



Nguồn 2. Entropy=1.55

Hình 1.3

Ta có thể tính entropy theo 2 cách

Cách thứ nhất: Tính theo công thức đã được định nghĩa.

- **Bước 1.** Xác định số trạng thái: bằng 2.

- **Bước 2.** Tìm ma trận xác suất chuyển trạng thái

$p_{11}=\frac{30+7+12}{(30+7+12)+26}=\frac{49}{75}$; $p_{12}=\frac{26}{(30+7+12)+26}=\frac{26}{75}$; $p_{21}=\frac{26}{26}=1$;

$a_{22}=0$

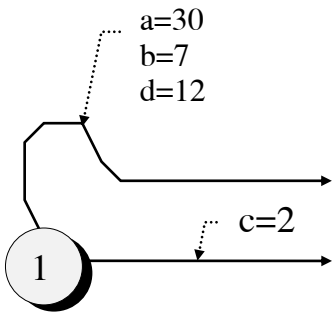
$$F=\begin{pmatrix} p_{11} & p_{21} \\ p_{12} & p_{22} \end{pmatrix}=\begin{pmatrix} \frac{49}{75} & 1 \\ \frac{26}{75} & 0 \end{pmatrix}$$

- **Bước 3.** Giải phương trình FΠ = Π

$$\begin{pmatrix} \frac{49}{75} & 1 \\ \frac{26}{75} & 0 \end{pmatrix}\begin{pmatrix} \pi_1 \\ \pi_2 \end{pmatrix}=\begin{pmatrix} \pi_1 \\ \pi_2 \end{pmatrix}$$, với điều kiện $\pi_1 + \pi_2 = 1$.

Ta thu được nghiệm $\pi_1=\frac{75}{101}$; $\pi_2=\frac{26}{101}$.

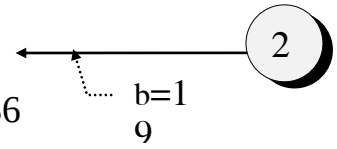
- **Bước 4.** Tính các entropy của từng trạng thái



Trạng thái 1

$$E_1 = \frac{30}{30+7+12+26}\log_2\frac{30+7+12+26}{30} + \frac{7}{30+7+12+26}\log_2\frac{30+7+12+26}{7} + \frac{12}{30+7+12+26}\log_2\frac{30+7+12+26}{12} + \frac{26}{30+7+12+26}\log_2\frac{30+7+12+26}{26} = 1.80096$$

Trạng thái 2



$$E_2 = \frac{19}{19+7}\log_2\frac{19+7}{19} + \frac{7}{19+7}\log_2\frac{19+7}{7} = 0.84036$$

- **Bước 5** Tìm entropy của nguồn bằng cách lấy tổng các tích xác suất xuất hiện của trạng thái với entropy riêng của nó.

$$E = \pi_1 E_1 + \pi_2 E_2 = \frac{75}{101}1.80096 + \frac{26}{101}0.84036 = 1.55368.$$

Kết luận entropy của nguồn là 1.55368

Cách thứ hai: Sử dụng khả năng tính nhanh của máy tính để mô phỏng sự hoạt động của nguồn nhằm mục đích tính các giá trị xác suất π_1 và π_2. Trong chương trình sau chúng được ký hiệu là Pa và Pb trong đó a là kí hiệu trạng thái 1 và b là kí hiệu trạng thái 2 của nguồn. Ea, Eb là các entropy riêng

tương ứng với các trạng thái a và b. Ta bắt đầu từ phân bố xác suất Pa=1 và Pb=0. Tức là bắt đầu tiến trình tại trạng thái 1. Ta dùng một tính chất của tiến trình egordic là trung bình theo thời gian bằng trung bình theo không gian. Kết quả được in ra E=1.55370 tuy không hoàn toàn chính xác nhưng nó gần đúng với giá trị thật của entropy.

```
var a, b, Ea, Eb, Pa, Pb: extended;
   i, s: longint;
begin
randomize;
a:=0; b:=0; s:=1;
for i:=1 to 10000000 do
  begi n
    if s=1 then
          if random < 26/(30+7+12+26) then
                        begin
                          s:=2;
                          b:=b+1;
                        end
                else a:= a+1;
    if s=2 then
          begin
            a:=a+1;
            s:=1;
          end;
end;
Pa:= a/(a+b);
Pb:=b/(a+b);
writeln(Pa:10:7,'  ',Pb:10:7);
```

```
Ea:=    -30/(30+7+12+26)*ln(30/(30+7+12+26))/ln(2)
        -7/(30+7+12+26)*ln(7/(30+7+12+26))/ln(2)
   -12/(30+7+12+26)*ln(12/(30+7+12+26))/ln(2)
   -26/(30+7+12+26)*ln(26/(30+7+12+26))/ln(2);
Eb:=    -19/(19+7)*ln(19/(19+7))/ln(2)
        -7/(19+7)*ln(7/(19+7))/ln(2);
writeln(Ea*Pa+Eb*Pb:10:5);
end.
```

Qua ví dụ trên nếu chỉ với mục đích ước lượng entropy thì ta có thể sử dụng phương pháp thứ 2 vì nó thật sự đơn giản hơn việc tìm vector riêng của một ma trận mà về nguyên tắc nó có thể có bậc rất lớn.

1.3.5. Nguồn có entropy nhỏ nhất.

Một văn bản có thể được sinh ra từ nhiều nguồn trạng thái khác nhau. Trong số chúng nguồn nào có entropy nhỏ nhất thì văn bản do chúng sinh ra sẽ nén lại được nhiều nhất. Bài toán đặt ra là dựa vào một văn bản làm sao có thể tìm được mô hình nguồn sinh ra văn bản ấy mà lại có entropy nhỏ nhất. Khi tăng số đỉnh của mô hình nguồn lên thì ta có cơ hội tìm thấy được các mô hình nguồn có entropy nhỏ hơn. Tuy nhiên số đỉnh của mô hình mà quá lớn thì nó cản trở cho việc thể nghiệm thuật toán, bởi vì bộ nhớ của máy chỉ có hạn. Trong số các mô hình nguồn có cùng số đỉnh thì ta mô hình nguồn với entropy nhỏ nhất được gọi là mô hình tối ưu. Như vậy bài toán nén dữ liệu dựa vào mô hình nguồn là làm sao tìm được mô hình nguồn tối ưu.

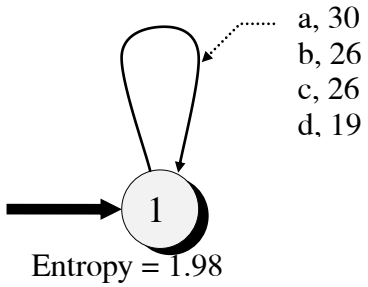
Ví dụ 1.4. Xét văn bản

adbadacbdcbacbdbacbacdcdacbadacbdba

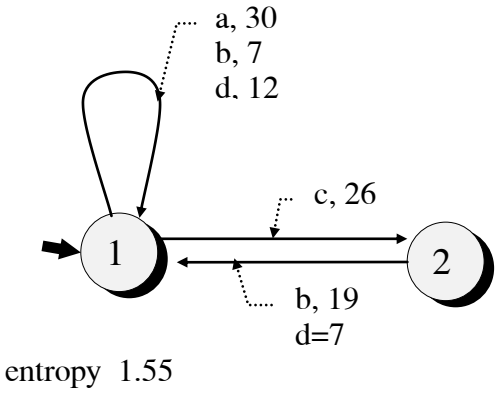
cbacbacdbadacbacbacbadacbacbacbadcd

bacbadbacdbdcbacdacbacbacbacdda

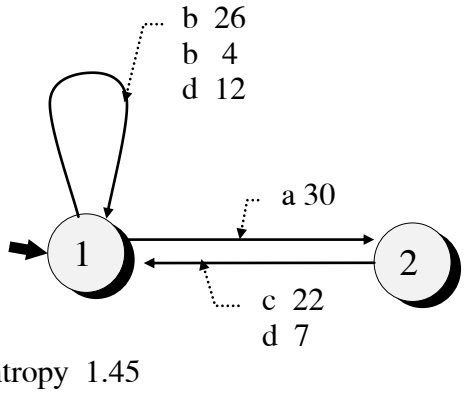
Các mô hình nguồn sau là các mô hình có thể và tối ưu có cùng số đỉnh. Phía bên phải là mô hình tối ưu, còn phía bên trái là mô hình có cùng số đỉnh nhưng không phải là mô hình tối ưu. Tất cả các nguồn sau đều sinh ra được văn bản nói trên. Sự khác biệt chỉ là entropy của chúng.



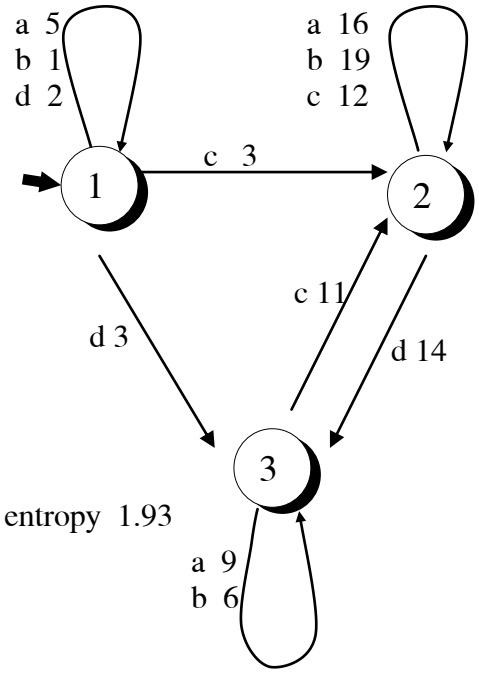
Hình 1.4a



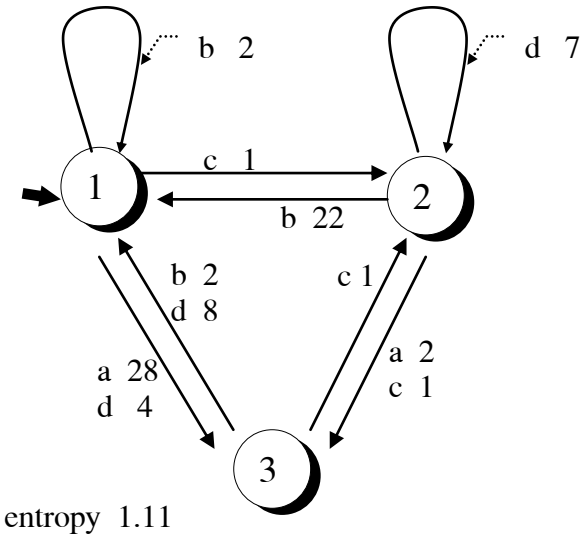
Hình 1.4b



Hình 1.4c



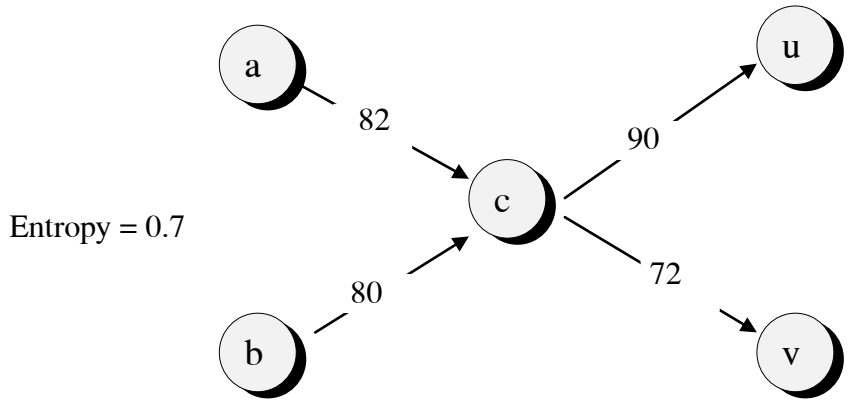
Hình 1.4d



Hình 1.4e

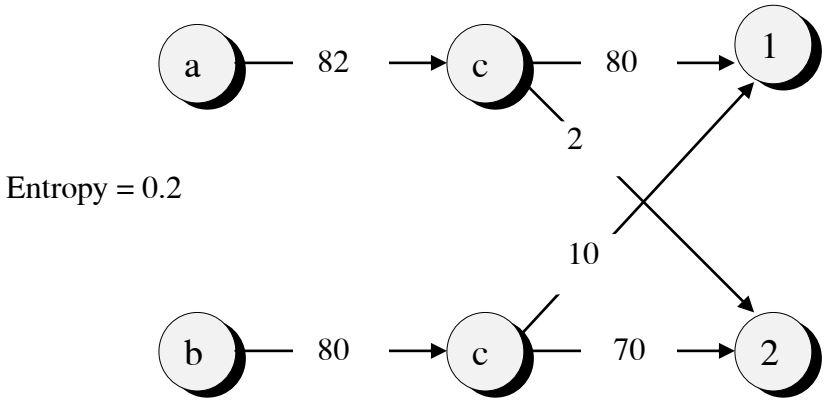
Tạo ra thuật toán để tìm nguồn có entropy nhỏ nhất có thể là một bài toán khó. Có một giải pháp gọi là phương pháp tự phân chia (clone) để tìm ra được một mô hình có entropy nhỏ hơn, nhưng không chắc đã là mô hình tối ưu. Phương pháp này là cơ sở cho thuật toán nén DMC (Dynanic Markov Coding.)

Giả sử ta có mô hình mà tại điểm ‘c’ có một số đỉnh đi tới và đi ra với các trọng số như sau.



Hình 1.5

Trong số các đỉnh đi vào ‘c’ giả sử như đi từ đỉnh ‘b’ lại là thường vào ‘v’, trong khi đi từ ‘a’ lại thường vào u. Để cho dễ mường tượng ta coi ‘c’ như một nút giao thông, mà ở đó người ta đi từ ‘a’ và ‘b’ tới ‘u’ và ‘v’. Nếu là đường đi bộ thì bằng quan sát, ta cũng thấy lối đi sẽ tách dần ra làm 2. Như vậy chỉ cần biết một người đi từ đâu tới là ta có thể đoán biết được anh ta sẽ đi đâu. Do đó mô hình sau sẽ phản ánh đúng thực chất của sự phụ thuộc hơn. Tức là nó có entropy nhỏ hơn mô hình cũ.



Hình 1.6

Ngược lại với tự phân chia là nhập 2 đỉnh lại thành 1 đỉnh - kiêm nhiệm, nếu như việc nhập này không làm thay đổi entropy quá nhiều mà lại tiết kiệm được bộ nhớ do số đỉnh ít đi.

1.4. Các trình ví dụ

- Ví dụ 1.5. Trình tạo ra file văn bản một cách ngẫu nhiên từ các chữ cái a và b, với xác suất tương ứng p1 = 2/3, p2 = 1/3.

Theo lý thuyết ta có $E = 2/3 \log_2(3/2) + 1/3 \log_2(3) \approx 0.918$

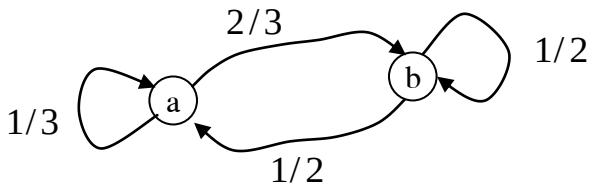
Sau khi nén còn ≈ 11%

```
uses crt;
var f:file of byte;
   a,b,c,d : byte;
   i, da, db : longint;
   E:real;
begin
assign(f,'c:\kpt1.txt');rewrite(f);
 a:=ord('a');b:=ord('b');
da:=0;
db:=0;
 for i:=1 to 640000 do
     begin
      if random<=2/3 then
            begin
               write(f,a);
               da:=da+1;
            end
            else
            begin
               write(f,b);
               db:=db+1;
```

```
            end
        end;
        close(f);
        clrscr;
        E:=(da/640000*ln(640000/da)+ db/640000*ln(640000/db))/ln(2);
        writeln('   ty le nen con = ',round(E/8*100), '%');
        Readln;
        end.
```

- Ví dụ 1.6.

Trình sau tạo ra file văn bản theo mô hình Markov. File nén có dung lượng bằng 12%,



Hình 1.7

Ma trận trạng thái là

$$\begin{pmatrix} 1/3 & 1/2 \\ 2/3 & 1/2 \end{pmatrix}$$

Phân bố ổn định là nghiệm của phương trình

$$\begin{pmatrix} 1/3 & 1/2 \\ 2/3 & 1/2 \end{pmatrix}\begin{pmatrix} p_a \\ p_b \end{pmatrix}=\begin{pmatrix} p_a \\ p_b \end{pmatrix}$$

Với điều kiện $p_a + p_b = 1$

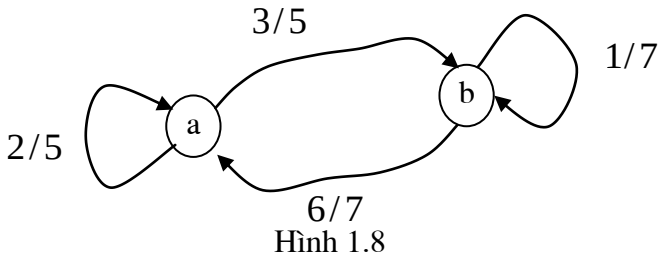
Lời giải là $p_a = 3/7$ và $p_b = 4/7$

```
var f:file of byte;
   a,b:byte;
   M:char;
   i:word;
   Ea,Eb,pa,pb,E:real;
begin
assign(f,'c:\CPT1.txt');rewrite(f);
 a:=ord('a');b:=ord('b');
 M:='a';
da:=0; db:=0;
 for i:=1 to 64000 do
 case M of
     'a':begin if random<1/3 then
                 begin write(f,a);M:='a'; end
               else
                 begin write(f,b);M:='b'; end;
         end;
     'b':begin if random<1/2 then
                 begin write(f,a);M:='a'; end
               else
                 begin write(f,b);M:='b'; end;
         end;
 end;
```

```
close(f);
Ea:=(1/3*ln(1/(1/3))+ 2/3*ln(1/(2/3)))/ln(2);
Eb:=(1/2*ln(1/(1/2))+ 1/2*ln(1/(1/2)))/ln(2);
pa:=3/7; pb:=4/7;
E:=pa*Ea+pb*Eb;
writeln('   ty le nen = ',round(E/8*100),' %');
Readln;
end.
```

- Ví dụ 1.7

Trình sau tạo ra file văn bản theo mô hình Markov. File nén có dung lượng bằng 10%.



Hình 1.8

Ma trận trạng thái là

$$\begin{pmatrix} 2/5 & 6/7 \\ 3/5 & 1/7 \end{pmatrix}$$

Phân bố ổn định là nghiệm của phương trình

$$\begin{pmatrix} 2/5 & 6/7 \\ 3/5 & 1/7 \end{pmatrix}\begin{pmatrix} p_a \\ p_b \end{pmatrix}=\begin{pmatrix} p_a \\ p_b \end{pmatrix}$$

Với điều kiện $p_a + p_b = 1$

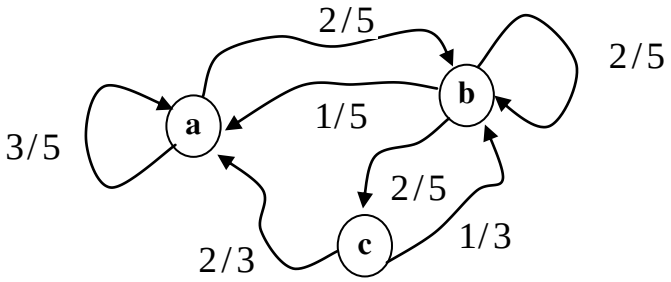
Lời giải là $p_a = 30/51$ và $p_b = 21/51$

```
var f:file of byte;
    a,b:byte;
    M:char;
    i,j:word;
    Ea,Eb,pa,pb,E:real;
    Na,Nb,N:real;
begin
assign(f,'c:\CPT2.txt');rewrite(f);
 a:=ord('a');b:=ord('b');
 M:='a';
 Na:=0;Nb:=0;N:=10*64000;
 for j:=1 to 10 do
 for i:=1 to 64000 do
 case M of
     'a':begin if random<2/5 then
                 begin write(f,a);M:='a';Na:=Na+1;  end
```

```
            else
              begin write(f,b);M:='b';Nb:=Nb+1; end;
        end;
    'b':begin if random<6/7 then
              begin write(f,a);M:='a';Na:=Na+1; end
            else
              begin write(f,b);M:='b';Nb:=Nb+1; end;
        end;
 end;
close(f);
Ea:=(2/5*ln(1/(2/5))+ 3/5*ln(1/(3/5)))/ln(2);
Eb:=(1/7*ln(1/(1/7))+ 6/7*ln(1/(6/7)))/ln(2);
writeln('   pa=',Na/N,' pb= ',Nb/N);
pa:=Na/N; pb:=Nb/N;  {pa=30/51  & pb=21/51}
E:=pa*Ea+pb*Eb;
write('   ty le nen = ',round(E/8*100));
end.
```

- Ví dụ 1.8

Trình sau tạo ra file văn bản theo mô hình Markov. File nén có dung lượng bằng 15%.



Hình 1.9

Ma trận trạng thái là

$$\begin{pmatrix} 3/5 & 1/5 & 2/3 \\ 2/5 & 2/5 & 1/3 \\ 0 & 2/5 & 0 \end{pmatrix}$$

Phân bố ổn định là nghiệm của phương trình

$$\begin{pmatrix} 3/5 & 1/5 & 2/3 \\ 2/5 & 2/5 & 1/3 \\ 0 & 2/5 & 0 \end{pmatrix}\begin{pmatrix} p_a \\ p_b \\ p_c \end{pmatrix}=\begin{pmatrix} p_a \\ p_b \\ p_c \end{pmatrix}$$

Với điều kiện $p_a + p_b + p_c = 1$

Lời giải là $p_a = 35/77,\ p_b = 30/77,\ p_c = 12/77$

```
var
   f:file of byte;
   a,b,c:byte;
   M:char;
   i,j:word;
   Ea,Eb,Ec,pa,pb,pc,E:real;
   Na,Nb,Nc,N,R:real;
begin
assign(f,'c:\CPT3.txt');rewrite(f);
 a:=ord('a');b:=ord('b');c:=ord('c');
 M:='a';
 Na:=0;Nb:=0;Nc:=0;N:=10*64000;
 for j:=1 to 10 do
```

```
for i:=1 to 64000 do
Begin
R:=random;
case M of
    'a':begin if R<3/5 then
              begin write(f,a);M:='a';Na:=Na+1;end
            else
              begin write(f,b);M:='b';Nb:=Nb+1;end;
        end;
    'b':begin if R<1/5 then
              begin write(f,a);M:='a';Na:=Na+1;end
            else if R<3/5 then begin write(f,b);M:='b';Nb:=Nb+1;end
            else begin write(f,c);M:='c';Nc:=Nc+1;end
        end;
    'c':begin if R<2/3 then
              begin write(f,a);M:='a';Na:=Na+1;end
            else
              begin write(f,b);M:='b';Nb:=Nb+1;end;
        end;
end;
end;
close(f);
writeln(' .....', Na+Nb+Nc, '=' ,N);
Ea:=(3/5*ln(1/(3/5))+ 2/5*ln(1/(2/5)))/ln(2);
Eb:=(1/5*ln(1/(1/5))+ 2/5*ln(1/(2/5))+ 2/5*ln(1/(2/5)))/ln(2);
Ec:=(2/3*ln(1/(2/3))+ 1/3*ln(1/(1/3)))/ln(2);
pa:=Na/N; pb:=Nb/N; pc:=Nc/N; {pa:=35/77; pb:= 30/77; pc:= 12/77;}
writeln('  pa=',pa:3:7,' pb=',pb:3:7,' pc=',pc:3:7);
E:=pa*Ea+pb*Eb+pc*Ec;
write('   ty le nen = ',round(E/8*100));
  end.
```

Chương 2. Các mã và thuật toán nén văn bản cổ điển

2.1. Mã tổng và mã phân tách

- **Định nghĩa 2.1** Cho A và B là hai văn bản. Tổng của A+B là một văn bản mới thu được từ A viết tiếp B vào bên phải của A. Như vậy độ dài của tổng các văn bản là tổng của các độ dài của chúng.
- **Định nghĩa 2.2** Một mã được gọi là mã tổng nếu như bản mã của tổng các văn bản là tổng của các bản mã.

Trong định nghĩa cho mã tổng ta đã sử dụng khái niệm "tổng của các văn bản". Nếu mã của "a" là f(a), của b là f(b) thì mã của "ab" là "f(a)f(b)", mã của "ba" là "f(b)f(a)". Xét mã tổng trên bảng chữ cái $\Omega=\{a_1, a_2, ..., a_m\}$. Mỗi chữ cái $a_1, a_2, ..., a_m$ có mã của nó, mà ta gọi là từ mã. Từ mã của các chữ cái xác định ánh xạ f: $\Omega \rightarrow M$, từ tập các chữ cái vào tập các xâu bit "0/1". Như vậy với mọi $x \in \Omega$, xâu bit f(x) là từ mã của x, độ dài xâu bit f(x) được ký hiệu là $\vartheta(x)$.

Theo định nghĩa mã tổng thì xâu các chữ cái $\zeta= \omega_1\omega_2...\omega_n$ tương ứng duy nhất với xâu bit có dạng $f(\zeta)=f(\omega_1)+f(\omega_2)+...+f(\omega_n)$. Bản mã $f(\zeta)$ có độ dài $L(\zeta)=\sum_{i=1}^{n} \vartheta(\omega_i)$ bit.

- **Định lý 2.1.** Nếu f: $\Omega \rightarrow M$ là mã tổng xác định trên bảng chữ cái $\Omega=\{a_1,a_2,....,a_m\}$, mà mỗi chữ cái $a_1, a_2, ..., a_m$ có xác suất xuất hiện tương ứng là $p_1, p_2, ..., p_m$ thì

Bit trung bình cho một chữ cái của hầu hết các văn bản có n chữ $\zeta= \omega_1\omega_2...\omega_n$

thoả mãn $\lim_{n\to\infty} \frac{\sum_{i=1}^{n} \vartheta(\omega_i)}{n} = \sum_{j=1}^{m} p_j \vartheta(a_j)$, ở đây $\vartheta(\omega)$ là độ dài từ mã của chữ cái $\omega \in \Omega$.

Bit trung bình của mã $\dfrac{\sum_{\zeta \in A_n} p(\zeta)L(\zeta)}{n} = \sum_{j=1}^{m} p_j \vartheta(a_j)$

Trong đó $p(\zeta)=p(\omega_1)p(\omega_2)...p(\omega_n)$ là xác suất xuất hiện văn bản ζ, và $L(\zeta)$ là độ dài bản mã của nó.

(định lý đã được chứng minh trong tài liệu lý thuyết mã nén của nhóm tác giả: Nguyễn Lê Anh, Trần Duy Lai, Phạm Thế Long, Nguyễn Văn Xuất).

Từ đây, ta chỉ đề cập đến các mã tổng nhị phân. Nếu các từ mã có độ dài cố định thì ta luôn giải mã được. Nhưng nếu độ dài của từ mã thay đổi thì không phải với ánh xạ mã nào cũng có thể giải mã được.

Ví dụ 2.1. Xét ánh xạ mã

a -> 100

b -> 1000

c -> 0

Mã của "ac" và "b" đều là dãy bit "1000". Như vậy khi nhận được chuỗi bit 1000 ta không thể biết được rằng văn bản ban đầu là "b" hay là "ac". Cho nên ánh xạ tạo thành bảng mã cho các chữ cái cần phải có tính chất là giải mã được. Tính phân tách được đưa ra dưới đây sẽ đảm bảo cho tính giải được của mã.

- **Định nghĩa 2.3:** Cho A và B là hai đoạn tạo ra từ các bit 0/1. Ta nói A là đầu của B nếu như có một đoạn C sao cho B = A + C.

- **Định nghĩa 2.4:** Một tập hợp M tạo ra từ các đoạn bit 0/1 được gọi là phân tách nếu không có đoạn nào là đầu của đoạn kia.

Như vậy, mã có độ dài từ mã cố định là mã phân tách.

- **Định lý 2.2.** Điều kiện đủ để giải mã được một dãy bit được tạo bởi một mã tổng từ một bảng mã bit "0/1" có độ dài thay đổi là mỗi chữ cái ứng với một xâu bit không có xâu nào là bắt đầu của xâu khác.

- **Định lý 2.3. (Kraft-McMilan)**

Điều kiện cần và đủ để có mã tổng mã các chữ cái $\Omega=\{a_1, a_2, ..,a_m\}$ bằng xâu bit 0/1 với độ dài tương ứng $\vartheta_i=\vartheta(a_i)$ là $\sum_{i=1}^{m}\frac{1}{2^{\vartheta_i}}\le 1$.

- Điều kiện cần (Định lý McMillan)

- Điều kiện đủ (Định lý Kraft)

(định lý đã được chứng minh trong tài liệu lý thuyết mã nén của nhóm tác giả: Nguyễn Lê Anh, Trần Duy Lai, Phạm Thế Long, Nguyễn Văn Xuất).

Hệ quả Mọi mã tổng đều có thể thay thế bằng mã phân tách có cùng độ dài các từ mã.

2.2. Mã Shannon

Xét bảng các chữ cái $\Omega=\{a_1,a_2,.....a_m\}$ với xác suất tương ứng $p_1\ge p_2\ge ... \ge p_m>0$. Ta xây dựng một mã phân tách cho bảng chữ cái Ω như sau.

Với mỗi p_i có thể chỉ ra số nguyên r_i sao cho $\frac{1}{2^{r_i}}\le p_i<\frac{1}{2^{r_i-1}}$. Rõ ràng, nếu i<j thì do $p_j\le p_i<\frac{1}{2^{r_i-1}}$ nên $r_i\le r_j$.

Sử dụng kí hiệu

$Q_1 = 0$

$Q_2 = p_1$

$Q_3 = p_1+ p_2$

$Q_4 = p_1+ p_2+ p_3$

.....

$Q_m = p_1+ p_2+.....+ p_{m-1}$

Khi đó do $p_1, p_2, ..., p_m>0$ nên $Q_1< Q_2<.....<Q_m<1$

Một số x<1 bất kỳ có thể biểu diễn duy nhất ở dạng $x=\frac{\alpha_1}{2}+\frac{\alpha_2}{2^2}+\frac{\alpha_3}{2^3}+...+\frac{\alpha_k}{2^k}+....$

Trong đó α_i là các số hoặc bằng 0 hoặc bằng 1. Ta sử dụng ký hiệu $0,\alpha_i\alpha_2\alpha_3...\alpha_k...$ để ghi lại tổng trên và gọi nó là biểu diễn theo cơ số 2 của số x. Như vậy, số có dạng 0,11 có nghĩa là $\frac{1}{2}+\frac{1}{2^2}=\frac{3}{4}$. Trong phần chứng minh sau vì không có khả năng gây ra nhầm lẫn nên để cho đơn giản ta bỏ "0," ra khỏi biểu diễn trên, và vẫn gọi $\alpha_i\alpha_2\alpha_3....\alpha_k...$ là biểu diễn cơ số 2 của x.

Xét biểu diễn các số $Q_1< Q_2<.....<Q_m$ dưới dạng cơ số 2 như trên. Cứ với mỗi một trong m dãy cơ số 2 nói trên ta giữ lại, tương ứng với từng Q_i dãy $\wp_i$ tạo ra từ r_i số đầu tiên. Như vậy, ta có m dãy $\wp_i$ với i=1..m là các dãy tạo ra từ các bit "0,1". Với mỗi i=1..m ta sử dụng $\wp_i$ để mã hoá trạng thái a_i thì thu được một phương pháp mã nhị phân trong đó mỗi trạng thái a_i được ứng với một dãy có r_i bit. Loại mã này gọi là mã **Shannon**.

- Thuật toán tìm mã Shannon.

  Input. nhập n và các giá trị xác suất $P_1\geq P_2\geq..... P_n$

  Out put. tính code[i]

```
Q:=0;                                   for j:=1 to r do
for i:=1 to n do                        begin
begin                                   S:=S*2;
  r:=1;w:=1/2;                          if S>1 then
  while not (w<= P_i) do                    begin
      begin                             S:=S-1;
      w:=w/2;r:=r+1;                    code[i]:=code[i]+'1'
      end;                              end
  code[i]:='';                          else code[i]:=code[i]+'0'
  S:=Q;Q:=Q+ P_i;                           end;
                                          end;
```

Chương trình minh hoạ tạo mã Shannon.

```
const n=20;                 {Số ký tự của bảng chữ cái}
var   P:array[1..n] of real;        {Xác suất từng ký tự}
     code:array[1..n] of string;  {Mã Shannon cho từng ký tự}
Procedure coding;
Var     S,Q,w: real;
i,j,r:integer;
Begin
Q:=0;
for i: =1 to n do
begin
  r:=1;w:=1/2;
while not (w<= P[i]) do begin w:=w/2;r:=r+1;end;
  code[i]:='';
  S:= Q;Q:=Q+ P[i];
  for j:=1 to r do
    begin
S:=S*2;
    if S>1 then begin S:=S-1;code[i]:=code[i]+'1';end
```

```
else code[i]:=code[i]+'0'
    end;
end;
End;
{Phần chính của trình.}
const U:array[1..n] of integer=
(371,332,313,257,252,249,205,202,178,173,151,132,123,107,73,59,48,4,2,1);
Var i:integer;
   s:real;
   f:text;
Begin
{Nhập dữ liệu}
s:=0;for i:=1 to n do s:=s+U[i];
for i:=1 to n do
     begin
        p[i]:=U[i]/s;code[i]:='';
     end;
{Tạo mã}
Codi ng;
{Ghi kết quả ra file}
assign(f,'c:\kq.txt');rewrite(f);
for i:=1 to n do writeln(f,code[i]:15,U[i]:5);
close(f);
End.
```

- **Định lý 2.4.**

Cho bảng chữ cái $\Omega=\{a_1,a_2,.....a_m\}$ với xác suất tương ứng $p_1 \geq p_2 \geq ... \geq p_m>0$.

Mã Shannon là mã phân tách. Nó mã mỗi chữ cái a_i với xác suất p_i bằng một từ mã nhị phân có độ dài r_i thoả mãn $\log_2 \frac{1}{p_i} \le r_i < 1 + \log_2 \frac{1}{p_i}$

Bit trung bình của mã Shannon $\bar{r} = \sum_{i=1}^{m} p_i \cdot r_i$ thoả mãn hệ thức

$$\sum_{i=1}^{m} p_i \cdot \log_2 \frac{1}{p_i} \le \bar{r} < \sum_{i=1}^{m} p_i \cdot \log_2 \frac{1}{p_i} + 1 .$$

Hay Entropy(Ω)≤ $\bar{r}$ < Entropy(Ω)+1.

(định lý đã được chứng minh trong tài liệu lý thuyết mã nén của nhóm tác giả: Nguyễn Lê Anh, Trần Duy Lai, Phạm Thế Long, Nguyễn Văn Xuất).

2.3. Mã tối ưu và sự tồn tại của mã tối ưu

2.3.1. Định nghĩa mã tối ưu.

Cho bảng chữ cái $\Omega=\{a_1,a_2,.....,a_m\}$ với xác suất tương ứng $p_1 \ge p_2 \ge ..... \ge p_m > 0$.

Xét mã tổng ξ trên Ω với các từ mã tương ứng là $e_1=\xi(a_1)$, $e_2=\xi(a_2)$,..., $e_m=\xi(a_m)$. Các từ mã $e_1, e_2,...., e_m$ có độ dài tương ứng là $\vartheta_1, \vartheta_2,...., \vartheta_m$.

Một mã tổng ξ được gọi là tối ưu nếu bit trung bình của mã $\lambda_\xi = \sum_{i=1}^{m} p_i \vartheta_i$ là nhỏ nhất có thể. Lưu ý rằng mọi mã tổng có thể thay thế bởi một mã phân tách cho nên mã tối ưu là mã phân tách.

Ta đi chứng minh có tồn tại mã tối ưu.

2.3.2. Sự tồn tại của mã tối ưu

Khẳng định:

- Mã tối ưu đã tồn tại
- Trong số các mã tối ưu thì tìm được một mã tối ưu mà
  - Chữ cái có xác suất lớn hơn sẽ có độ dài từ mã bé hơn.
  - Từ mã của hai chữ cái có xác suất nhỏ nhất có cùng độ dài và chỉ khác nhau bit cuối cùng.

(Khẳng định đã được chứng minh trong tài liệu lý thuyết mã nén của nhóm tác giả: Nguyễn Lê Anh, Trần Duy Lai, Phạm Thế Long, Nguyễn Văn Xuất).

2.4. Mã Huffman

- **Định nghĩa 2.5**

Nếu bảng chữ cái chỉ có 2 chữ cái thì ta đánh mã chúng là "0" và "1".

Ta định nghĩa mã Huffman cho bảng có m chữ cái bằng đệ qui như sau:

Xếp bảng chữ cái theo thứ tự xác suất xuất hiện của nó giảm dần ($p_1 \geq p_2 \geq ... \geq p_m > 0$). Như vậy chữ cái ở cuối bảng là chữ cái có xác suất xuất hiện nhỏ nhất.

Ghép 2 chữ cái với xác suất nhỏ nhất lại thành một chữ cái kép với xác suất xuất hiện là tổng của hai xác suất ấy. Như vậy trong bảng chữ cái mới 2 chữ cái này bị loại nhưng chữ cái kép được thêm vào.

Tạo mã Huffman cho bảng chữ cái mới này (có m - 1 chữ).

Tạo 2 từ mã mới bằng cách thêm "0" và thêm "1" vào mã của chữ cái kép. Gán 2 mã này cho 2 chữ cái bị ghép lại.

- **Thuật toán tạo mã Huffman.**

Bước 1. Liệt kê tất cả chữ cái cùng với xác suất của nó theo thứ tự giảm dần.

Bước 2. Ghép 2 chữ cái có xác suất nhỏ nhất (2 chữ cuối bảng) thành một chữ cái kép. Giả sử như 2 chữ ấy là "a","b". Ta dùng kí hiệu {a,b} để ký hiệu chữ cái kép ấy. Xác suất của chữ cái kép bằng tổng của 2 xác suất của 2 chữ cái tạo ra chữ kép ấy.

Bước 3. Nếu đã tìm được mã cho bảng cái "kép" thì mã của chữ "a" sẽ gồm mã của chữ kép thêm 0, và mã chữ "b" thêm 1.

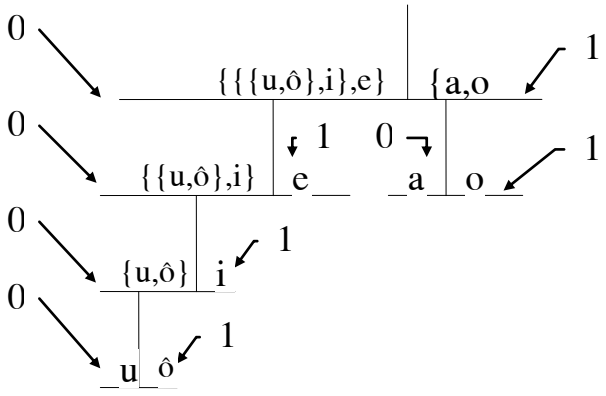
Bước 4. Quay lại bước 1 cho đến khi chỉ còn 1 chữ kép có xác suất bằng 1.

Ví dụ 2.2. Với không gian xác suất các sự kiện {e, a, i, o, u, ô} các xác suất tương ứng là (e,0. 3) (a,0.2) (o,0.2) (i,0.1) (u,0.1) (ô,0.1) thì ta cần ghép 5 lần như sau:

e → 0.3 a → 0.2 o → 0.2 i → 0.1 u → 0.1 ô → 0.1	e →0.3 a →0.2 o →0.2 {u,ô} →0.2 i →0.1	e →0.3 {{u,ô},i}→0.3 a →0.2 o →0.2	{a,o}→0.4 e →0.3 {{u,ô},i}→0.3	{{{u,ô},i},e}→0.6 {a,o} →0.4	{{{{u,ô},i},e},{a,o}}→1.0

Bảng 2.1

Việc gán mã đ- ợc thực hiện nh- sau:



Bảng mã của các chữ cái.

u→0000
ô→0001
i→001
e→01
a→10
o→11

- **Trình minh hoạ tạo mã Huffman**

Dưới đây là trình lập mã Huffman bằng Pascal theo thuật toán đã mô tả ở trên. Sử dụng phương pháp đệ qui thì có ưu điểm là dễ hiểu nhưng cũng có nhược điểm là đòi hỏi bộ nhớ lớn.

Const n=20;

```
Type nod=record
        code:string;
        prob:integer;
  end;
var
    a:array[1..n] of nod;
    x:nod;
    Sx:string;
    i,k:integer;
    f:text;
Procedure coding(m:integer);
var   k:integer;
        y:integer;
begin
        Case m of
          1     :exit;
          2..n :begin
              {Điều kiện thoát}
              if m=2 then
                begin
                    a[m-1].code:='0';a[m].code:='1';exit;
                 end;
              {Tạo chữ cái kép}
              y:=a[m-1].prob;inc(a[m-1].prob,a[m].prob);
              {Xếp lại}
              k:=m-1;
```

```
            while (k>1)and (a[k].prob>a[k-1].prob) do
              begin
                  x:=a[k-1]; a[k-1]:=a[k]; a[k]:=x; k:=k-1;
              end;
            {Giả sử đã có mã cho bảng chữ cái "kép"}
            coding(m-1);
            {Khi đó mã của các chữ cái là}
            Sx: =a[k].code;
            for i:=k to m-2 do a[i]:=a[i+1];
            a[m-1].code:=Sx+'0';a[m-1].prob:=y;
            a[m].code:=Sx+'1';
            end;
        end;
end;
{Phần chính của trình.}
const U:array[1..n] of integer =
(371,332,313,257,252,249,205,202,178,173,151,132,123,107,73,59,48,4,2,1);
begin
      {Nhập dữ liệu}
      for i:=1 to n do
       begin
            a[i].prob:=U[i];
            a[i].code:='';
       end;
      {Tạo mã} coding(n);
      {In kết quả}
```

```
        assign(f,'c:\KQ.txt');rewrite(f);
        for i:=1 to n do writeln(f,a[i].prob:4,' ',a[i].code);
        close(f);
    end.
```

- **Định lý 2.5.** Mã Huffman là mã tối ưu.
- **Định lý 2.6**. Đối với mã tối ưu thì $\sum_{i=1}^{m} p_i \log_2 \frac{1}{p_i} \le \sum_{i=1}^{m} p_i \vartheta_i \le 1+\sum_{i=1}^{m} p_i \log_2 \frac{1}{p_i}$.

(Các định lý đã được chứng minh trong tài liệu lý thuyết mã nén của nhóm tác giả: Nguyễn Lê Anh, Trần Duy Lai, Phạm Thế Long, Nguyễn Văn Xuất).

2.5. Mã Fano.

- **Thuật toán tạo mã Fano:**

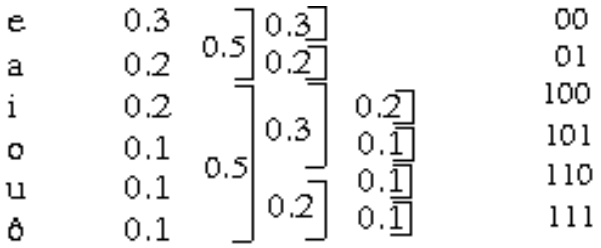
Giả sử ai với i=1..n là các chữ của một alphabet nào đó và ai xuất hiện với tần suất tương ứng là pi. Lưu ý rằng p1+p2+... + pn=1

Bước 1. Bằng cách xếp và kí hiệu lại ta có thể coi các chữ cái a_1, a_2, ..., a_n, có tần suất là $p_1 \ge p_2 \ge ... \ge p_n$ (theo thứ tự giảm dần).

Bước 2. Chia các chữ cái ra làm 2 nửa, nửa trên và nửa dưới, sao cho chúng có tổng gần bằng nhau nhất. Nửa trên nhận mã là 0, nửa dưới là 1.

Bước 3. Lặp lại công việc cho từng nửa và cứ tiếp tục với các nửa mới sinh ra cho tới khi trong mỗi nửa mới chỉ có 1 chữ cái. Dãy các số 0,1 được tạo ra là mã của các chữ cái.

Ví dụ 2.3. Không gian xác suất các sự kiện {e, a, i, o, u, ô} với các xác suất tương ứng là (e,0.3) (a,0.2) (i,0.2) (o,0.1) (u,0.1) (ô,0.1).



- **Trình minh hoạ tạo mã Fano**

```
Const n=20;                {số ký tự của bảng mã}
Type
    nod = record
        code:string;       {mã Huffman}
        prob:real;         {tần xuất}
    end;
var
    a:array[1..n] of nod;
    f:text;
    i:integer;
Procedure coding(bottom,top:integer);
var
  s, r:real;
  h:integer;
Begin
  {Điều kiện dừng}
  if bottom = top then exit;
  {Chia bảng mã ra làm 2 phần}
  s:=0;
  for i:=bottom to top do s:=s+a[i].prob;
  h:=bottom; r:=a[h].prob;
```

```
while r<s-r do
begin
    h:=h+1; r:=r+a[h].prob;
end;
if h=top then h:=h-1;
{Nửa dưới nhận mã 1}
for i:=bottom to h do a[i].code:=a[i].code+'1';
{Nửa trên nhận mã 0}
for i:=h+1 to top do a[i].code:=a[i].code+'0';
{làm tương tự như vậy cho mỗi nửa thu được}
coding(bottom,h);coding(h+1,top);
end;
const U:array[1..n] of integer=
(371,332,313,257,252,249,205,202,178,173,151,132,123,107,73,59,48,4,2,1);
Begin
    {Nhập dữ liệu}
    for i:=1 to n do
    begin
        a[i].prob:=U[n-i];a[i].code:= '';
    end;
    {Tìm mã}
    coding(1,n);
    {Ghi kết quả}
    assign(f, 'c:\KQ.txt ');rewrite(f);
    for i:=n downto 1 do writeln(f,a[i].prob:4:0, ' ',a[i].code);
    close(f);
```

End.

Kết quả chạy các chương trình trên được trình bày trong bảng tổng hợp ở phía sau.

Kết quả tính.

Bảng tổng hợp.

Mã Shannon	Mã Fano	Mã Huffman	Tần xuất
0000	000	100	371
0001	001	110	332
0011	010	111	313
0101	0110	0001	257
0110	0111	0010	252
0111	100	0011	249
1000	1010	0101	205
1001	10110	0110	202
10101	10111	1010	178
10111	1100	1011	173
11001	1101	00000	151
11010	11100	00001	132
11011	11101	01000	123
11101	11110	01110	107
111100	111110	01111	73
111101	1111110	010010	59
1111101	11111110	0100110	48
1111111101	111111110	01001110	4
11111111110	1111111110	010011110	2
111111111110	1111111111	010011111	1

Bảng 2.2

Bít trung bình cho từng loại:

Shannon			Fano			Huffman			xác suất
Mã	độ dài	xs* độ dài	Mã	độ dài	xs* độ dài	Mã	độ dài	xs* độ dài	
0000	4	0.459	000	3	0.344	100	3	0.344	0.115
0001	4	0.411	001	3	0.308	110	3	0.308	0.103
0011	4	0.387	010	3	0.291	111	3	0.291	0.097
0101	4	0.318	0110	4	0.318	0001	4	0.318	0.080
0110	4	0.312	0111	4	0.312	0010	4	0.312	0.078
0111	4	0.308	100	3	0.231	0011	4	0.308	0.077
1000	4	0.254	1010	4	0.254	0101	4	0.254	0.063
1001	4	0.250	10110	5	0.313	0110	4	0.250	0.063
10101	5	0.275	10111	5	0.275	1010	4	0.220	0.055
10111	5	0.268	1100	4	0.214	1011	4	0.214	0.054
11001	5	0.234	1101	4	0.187	00000	5	0.234	0.047
11010	5	0.204	11100	5	0.204	00001	5	0.204	0.041
11011	5	0.190	11101	5	0.190	01000	5	0.190	0.038
11101	5	0.166	11110	5	0.166	01110	5	0.166	0.033
111100	6	0.136	111110	6	0.136	01111	5	0.113	0.023
111101	6	0.110	1111110	7	0.128	010010	6	0.110	0.018
1111101	7	0.104	11111110	8	0.119	0100110	7	0.104	0.015
1111111101	10	0.012	111111110	9	0.011	01001110	8	0.010	0.001
11111111110	11	0.007	1111111110	10	0.006	010011110	9	0.006	0.001
111111111110	12	0.004	1111111111	10	0.003	010011111	9	0.003	0.000
bit trung bình		4.408			4.009			3.958	

Bảng 2.3

Theo như kết quả trên thì mã Huffman có bít trung bình nhỏ nhất, vì thế hệ số nén cao nhất.

Khẳng định.

Với nguồn có n sự kiện thì qui trình mã/giải nén mã Huffman và Shannon được thực hiện với 0(log2n) phép toán.

Chứng minh.

Quá trình mã là việc tra từ điển tìm mã 0/1 của nó. Quá trình này được thực hiện nhờ thuật toán tìm kiếm nhanh hết 0(log2(n)) phép toán. Quá trình giải nén thực hiện tìm kiếm nhanh nhờ cây nhị phân hết 0(log2(n)) phép toán. Như vậy tổng số thời gian cần để mã và giải nén hết 0(log2(n)) phép toán.

2.6. Mã Huffman động.

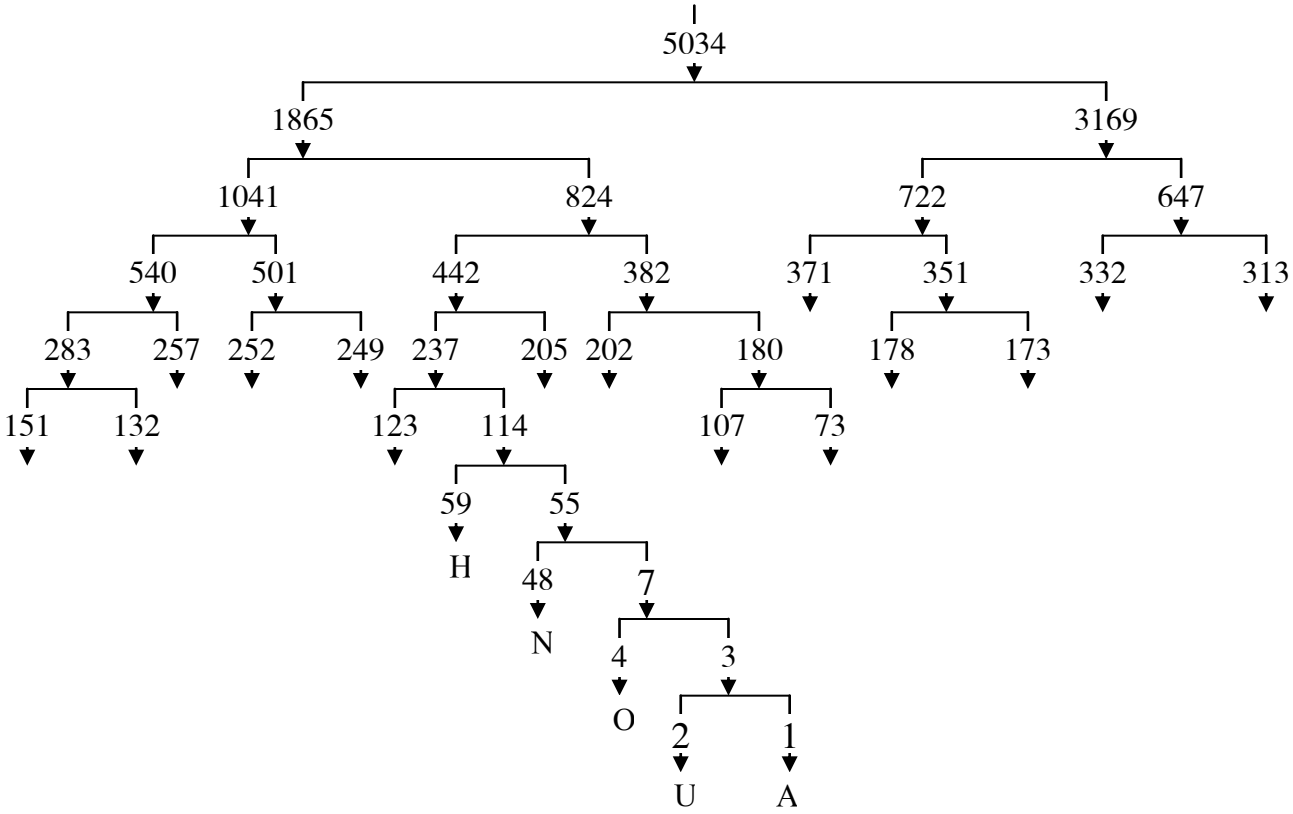
- Cây nhị phân cho mã Huffman động.

Nguyên lý tạo mã động là dựa vào việc tạo lại mã với bảng tần xuất mới. Tuy nhiên việc tạo lại bảng mã mất thời gian tính, làm giảm hiệu quả mã và giải mã. Phần này ta làm quen với thuật toán tạo nhanh bảng mã Huffman song song với quá trình mã và giải mã.

Nguyên tắc tạo mã Huffman là dựa vào việc thay hai chữ cái có tần xuất thấp nhất thành một chữ cái kép có tần xuất bằng tổng của chúng. Thực hiện quá trình nhóm cho tới khi ta chỉ có hai chữ cái. Quá trình sinh mã Huffman ngược với quá trình nhóm. Kết quả là ta thu được một cây nhị phân, mà lá của nó là các chữ cái. Tại mỗi lá có ghi tần xuất xuất hiện của chữ cái ấy và tại mỗi nhánh ghi tổng các tần xuất có ở các lá của nhánh. Các chỉ số này được gọi là "trọng số nhánh". Trọng số của nhánh bên trái luôn không nhỏ hơn trọng số của nhánh bên phải.

- **Quá trình giải mã.** Ta bắt đầu đi từ đỉnh cây và nếu gặp bit '1' thì rẽ sang nhánh bên phải, gặp bit '0' thì rẽ sang nhánh trái. Khi nào tới lá thì dừng lại và in chữ cái đó ra.
- **Quá trình mã.** Nhập chữ cái vào và kiểm tra xem có lá nào chứa chữ cái này không. Nếu có thì in ra con đường đi từ lá ấy tới gốc của cây, sao cho nếu rẽ sang trái thì in ra bit ‘1’ rẽ sang phải thì in ra bit ‘0’.

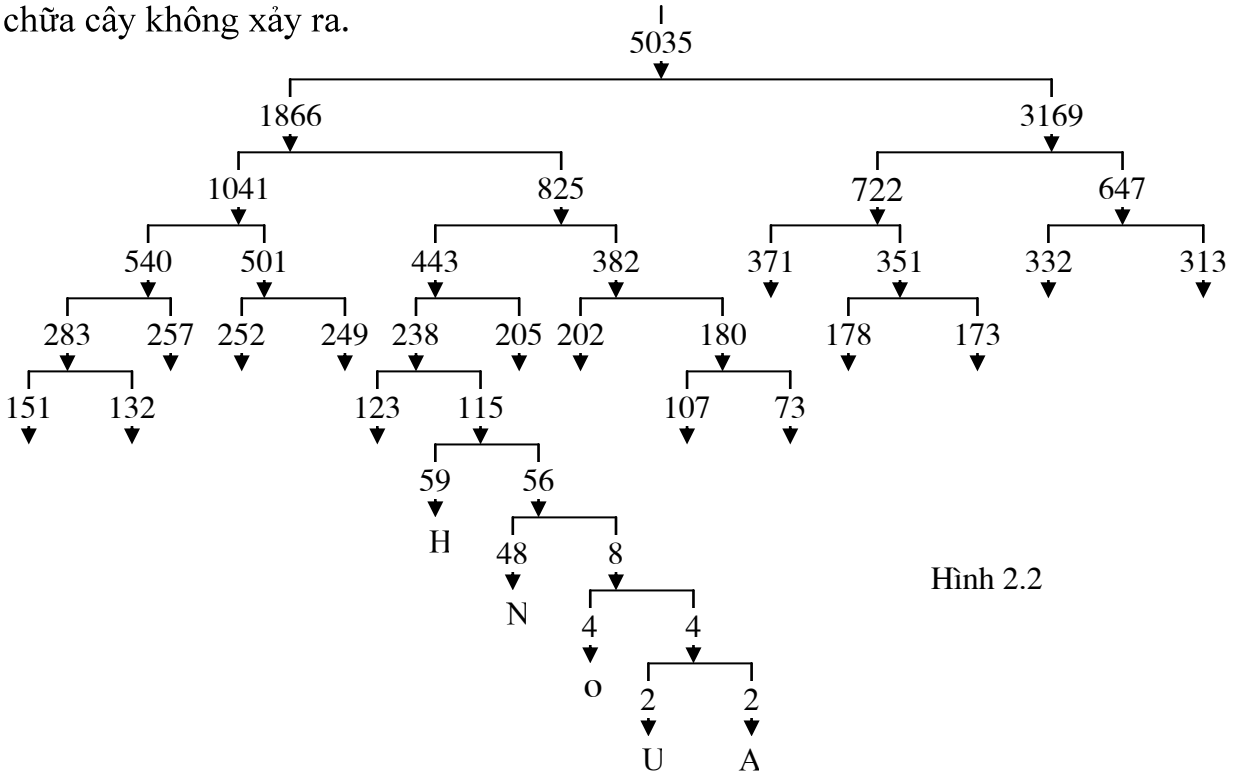
Cứ mỗi khi mã, hay giải mã được 1 chữ thì số lượng chữ cái mỗi loại thay đổi theo, vì thế cây nhị phân Huffman cần phải được sửa lại cho hợp với các số liệu thống kê mới.



Hình 2.1

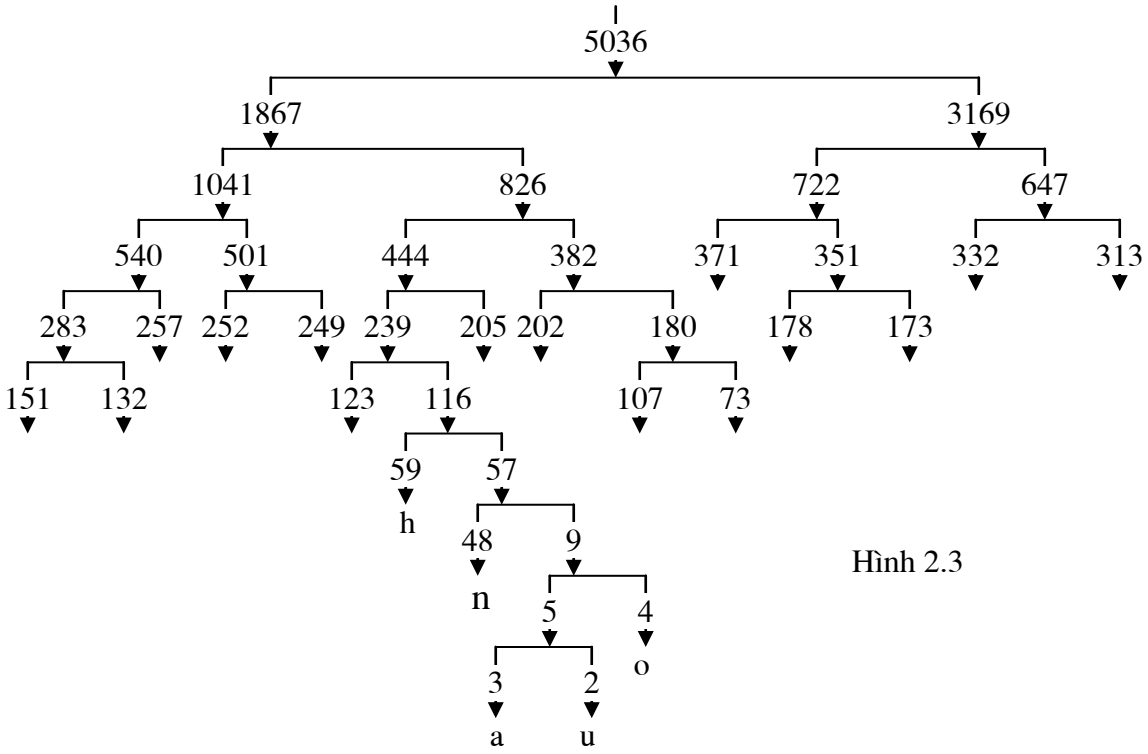
Giả sử tại một thời điểm nào đó có cây nhị phân mã Huffman sau:

Nếu chữ cái tiếp theo là "a" thì các trọng số sẽ thay đổi nhưng việc sửa chữa cây không xảy ra.



Hình 2.2

Nếu chữ tiếp theo là "a" nữa thì cây nhị phân sẽ đổi như sau:



Hình 2.3

- Trình tạo mã Huffman động

Thủ tục coding() được gọi đệ qui. Sau khi tìm được vị trí đúng cho đỉnh ghép thì 2 đỉnh cuối được tạo ra bằng các lệnh:

```
Sx:=a[k];
a[m-1]:=y;
a[m-1].code:=Sx.code+'0';
a[m].code:=Sx.code+'1';
```

trong đó y là đỉnh m-1 được lưu lại từ trước.

Do đỉnh ghép chèn vào giữa, nên các đỉnh phía sau phải dịch xuống:

```
for i:=k to m-2 do a[i]:=a[i+1];
```

Trình chính gọi lại coding(n) mỗi khi đọc thêm 1 chữ của văn bản và tính lại tần số.

```
Const n=8;
```

```
Type nod=record
        w:byte;
        c ode:string;
        prob:integer;
        end;
var  a   : array[1..n] of nod;
Sx, x    : nod;
k,i      : integer;
f   : text;
Procedure coding(m:integer);
var      k:integer;
   y:nod;
begin
Case m of
 1: exit;
 2..n: begin
     if m=2 then begin a[m-1].code:='0';a[m].code:='1';exit;end;
     y:=a[m-1];inc(a[m-1].prob,a[m].prob);
k:=m-1;
     while (k>1) and (a[k].prob>a[k-1].prob) do
     begin
        x:=a[k-1];a[k-1]:=a[k];a[k]:=x;k:=k-1;
     end;
coding(m-1);
     Sx:=a[k];for i:=k to m-2 do a[i]:=a[i+1];
     a[m-1]:=y;a[m-1].code:=Sx.code+'0';
```

```
        a[m].code:=Sx.code+'1';
        end;
    end;
    end;
    {Phần chính của trình.}
    const
    U:array[1..n] of integer=(1,1,1,1,1,1,1,1);
    S:string='aaaaaabcdefghaahhaaaaagabghabaecdcaaadaeccccccccccghaacbgbc
haecbdhabdehahcghghaebcd';
    Var h:word;
    begin
    for i:=1 to n do begin a[i].prob:=U[i];a[i].code:='';end;
    a[1].w:=ord('a');a[2].w:=ord('b');a[3].w:=ord('c');a[4].w:=ord('d');
    a[5].w:=ord('e');a[6].w:=ord('f');a[7].w:=ord('g');a[8].w:=ord('h');
    assign(f,'c:\KQ.txt');rewrite(f);
    h:=0;
    while true do
    begin
    coding(n);                                  {tạo mã}
    for i:=1 to n do writeln(f,char(a[i].w),' ',a[i].prob:4,' ',a[i].code);writeln(f);
    h:=h+1;if h>length(s) then begin close(f);exit;end;
    for i:=1 to n do if a[i].w=ord(s[h]) then inc(a[i].prob);      {thống kê lại tần
số}
    for i:=1 to n do a[i].code:='';
    for i:=n downto 2 do                        {xếp lại}
    if a[i].prob>a[i-1].prob then begin x:=a[i];a[i]:=a[i-1];a[i-1]:=x;end;
```

```
end;
end.
```

Đưa văn bản

aaaaaabcdefghaahhaaaaagabghabaecdcaaadaecccccccccghaacbgbchaecbdhabdehahcghghaebcd

Vào cho trình chương trình trên chạy, ta sẽ thu được bảng mã Huffman động. Để hình dung được sự thay đổi từ mã, ta in kết quả của 8 bước chạy trình, mỗi lần chạy đọc 1 chữ cái của văn bản.

a 010	a 01	a 00	a 1	a 1	a 1	a 0	a 1
b 011	b 001	b 011	b 011	b 011	b 011	b 111	b 010
c 000	c 0000	c 0100	c 0100	c 0100	c 0100	c 1100	c 0010
d 001	d 0001	d 0101	d 0101	d 0101	d 0101	d 1101	d 0011
e 110	e 110	e 110	e 0010	e 0010	e 0010	e 1010	e 0000
f 111	f 111	f 111	f 0011	f 0011	f 0011	f 1011	f 0001
g 100	g 100	g 100	g 0000	g 0000	g 0000	g 1000	g 0110
h 101	h 101	h 101	h 0001	h 0001	h 0001	h 1001	h 0111

Bảng 2.4

Theo dõi kết quả in ra ta nhận thấy có sự thay đổi liên tục của bảng mã, và cũng có lúc bảng mã không thay đổi. Sử dụng các mã do trình trên tạo ra, ta có dược mã của văn bản trên là 235 bit.

```
01001001111110010000101100011011001010010001011111110111100010000000110001101000101000100011111011010110000000000100100010000000000001111100000101000111010010011000100010111010000110011011011100110001110010001101000110001000110100
```

Khi thực hiện nén và giải nén bằng mã Huffman động.

Thông thường khi nén và giải nén các file, người ta sử dụng bảng chữ cái có 256 byte. Mặc dù điều này là không cần thiết. Mỗi khi gặp chữ cái mới, thì cây sẽ sinh thêm 1 nhánh cho lá ấy. Như vậy khi bắt đầu nén và giải nén cây

có thể chỉ gồm 1 gốc và 1 lá. Ngoài ra nội dung của lá mới này được ghi ngay vào bản mã nén, để phục vụ cho việc giải nén.

Chương 3. Mã số học

3.1. Biểu diễn nguồn.

Mỗi văn bản được ứng duy nhất với một khoảng[1] có độ dài bằng xác suất xuất hiện của văn bản. Văn bản dài thêm ra thì ứng với khoảng nhỏ dần.

- **ý tưởng chung**

Cách biểu diễn nguồn được trình bày ở đây đúng cho mọi mô hình nguồn mà theo đó tại mọi thời điểm ta biết được chữ nào sẽ xuất hiện với xác suất[2] bao nhiêu và xác suất ấy chỉ phụ thuộc vào các chữ đã xuất hiện trước đó. Chữ tiếp theo là một trong số các chữ cái của bảng chữ cái. Chữ cái đầu tiên của luồng tin S là empty. Biểu diễn empty là khoảng [0,1). Chia khoảng [0,1) ra thành các khoảng theo thứ tự tương ứng với các chữ cái của bảng chữ cái. Độ dài của các khoảng chia tương ứng với xác suất mà chữ cái ấy xuất hiện sau empty. Như vậy chữ xuất hiện tiếp theo empty sẽ là một trong số các khoảng [L(a),H(a)). Ta có thể coi khoảng [L(a),H(a)) như là khoảng [0,1) và xét kí tự xuất hiện tiếp theo. Lặp lại thao tác trên ta thu được một luồng tin S tương ứng duy nhất với khoảng [L(S),H(S)) nằm trong khoảng [0,1). Độ dài của khoảng này H(S)-L(S) bằng xác suất xuất hiện luồng tin S. Biểu diễn văn bản S thông qua khoảng [L(S),H(S)) được gọi là biểu diễn nguồn. Bất kỳ một số thực nào nằm trong khoảng [L(S),H(S)) là đủ để xác định văn bản S. Một số bất kỳ nằm trong khoảng [L(S),H(S)) được gọi là **mã số học** của S. Người ta thường biểu diễn số ở dạng nhị phân và mã số học của S được chọn ở dạng số nhị phân hữu hạn có độ dài nhỏ nhất có thể.

- Biểu diễn nguồn cho mô hình Markov.

[1] Để cho đơn giản ta gọi tắt nửa đoạn dạng [x,y) là khoảng.

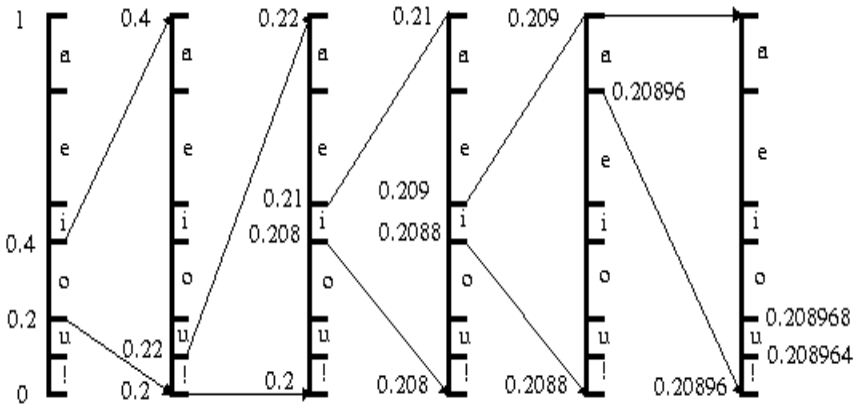
[2] Nh- vậy điều quan trọng là làm thế nào để luôn có thể xác định đ- ợc xác suất xuất hiện của chữ tiếp theo?

Ta xét mô hình Markov Ω có m trạng thái $\{u_1, u_2, .., u_m\}$ với xác suất p_1, p_2, p_3, ..., p_m tương ứng và sắp xếp thứ tự cho các cạnh đi ra từ từng trạng thái kèm theo xác xuất của nó. Giả sử đi ra từ trạng thái u_i là các trọng số w_{ij} (j=1,2,.., m_i). Xét văn bản S = a_1 a_2 a_3 ...., a_n ... trong đó $a_1 = u_{i_1}$, $a_2 = u_{i_2}$,.... $a_m = u_{i_m}$, ... Ta có thể biểu diễn hình học dãy S như sau. Chia khoảng [0,1) ra làm m phần theo thứ tự $\Delta_1, \Delta_2, ..., \Delta_m$ không giao nhau có dạng [x,y) có độ dài ứng với xác suất $p_1, p_2, p_3, ..., p_m$ của các phần tử Ω. Như thế để biểu diễn trạng thái thứ nhất của dãy S thì ta chỉ việc chỉ ra đoạn con ứng với trạng thái a_1, ký hiệu đó là Δ^1. Để biểu diễn trạng thái tiếp theo a_2, ta coi khoảng Δ^1 như là khoảng [0,1), sau đó tiến hành chia và chọn tương tự như với a_1. Cụ thể là ta chia Δ^1 ra làm một số khoảng tỷ lệ với các trọng số có thể chuyển đi từ a_1 theo thứ tự của các cạnh đã được định ra trước. Chọn khoảng $\Delta^2 \subset \Delta^1$ ứng với a_2 trong số các khoảng con vừa chia ra được từ Δ^1. Như thế khoảng Δ^2 có độ dài là tích của xác suất chọn a_1 là p_{i_1} và xác suất chọn a_2 khi đã chọn a_1 là $p_{i_1 i_2}$. Tức là độ dài của Δ^2 là xác suất xuất hiện của phép chọn kép a_1, a_2. Nếu ta cứ tiếp tục kéo dài biểu diễn các phần tử của dãy S thì ta thu được khoảng Δ^n biểu diễn dãy $a_1a_2a_3....a_n$ sao cho độ dài của Δ^n bằng xác suất xuất hiện của văn bản a_1 a_2 a_3... a_n. Phép tương ứng mỗi dãy các trạng thái ngẫu nhiên liên tiếp của nguồn trạng thái Ω bằng một khoảng như thế được gọi là biểu diễn số của nguồn. Trong biểu diễn số, entropy của văn bản $a_1a_2a_3....a_n$ bằng $\log_2 \frac{1}{\Delta^n}$. Nhận xét rằng để xác định một khoảng như trên ta cần chọn một số nào đó nằm trong khoảng ấy. Nếu ta sử dụng hệ thập phân thì nên chọn số đó là số thập phân hữu hạn có số chữ số ít nhất có thể được mà vẫn bảo đảm là nó nằm trong nửa khoảng nhỏ ấy. Như thế ta chỉ ra một mã dùng để mã các dãy của nguồn hay là một văn bản nào đó của nguồn.

Ví dụ 3.1. Để minh hoạ cho biểu diễn số, ta xét:

chữ cái và xác suất

a	0.2	[0.8, 1)
e	0.3	[0.5, 0.8)
i	0.1	[0.4, 0.5)
o	0.2	[0.2, 0.4)
u	0.1	[0.1, 0.2)
!	0.1	[0, 0.1)

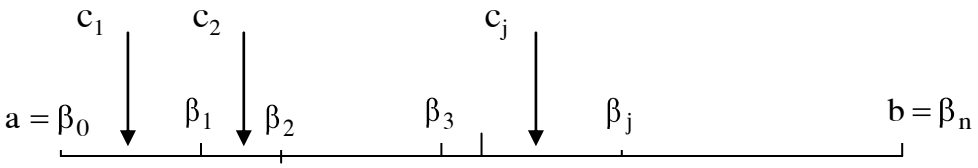


Hình 3.1

Như vậy dãy o!iiau được biểu diễn như là khoảng [0.208964, 0.208968). Bất kỳ một số nào nằm trong khoảng ấy đều có thể đại diện cho nó. Ta có thể mã o!iiau là 208964 cơ số 10. Nếu ta lấy dãy aaaaaa thì khoảng xác định nó là [0.999936,1) và có thể mã nó như là 99994. Có thể biểu diễn các số thực ở cơ số khác, ví dụ như cơ số 2, khi đó ta có mã bit 01 cho các dãy sinh ra từ nguồn này.

3.2. Mã số học với số nguyên

Bản chất của mã số học là mỗi một chữ cái được ứng với "khoảng xác định" có độ dài tỉ lệ tương ứng với tần suất xuất hiện của nó. Các chữ cái khác nhau nhất thiết phải được ứng với các khoảng không giao nhau. Giả sử bảng chữ cái có n chữ $c_1, c_2, ..., c_n$ với tần suất xuất hiện tương ứng là $p_1, p_2, ..., p_n$, ở đây $\sum_{j=1}^{n} p_j = p$ và với mọi j = 1..n thì $p_j > 0$. Đặt $\omega_j = \sum_{i=1}^{n-j} p_i$. Xét khoảng [a,b) $\subseteq$ [0,1). Ta xác định các mốc $\delta_j = a + (b-a)\frac{\omega_j}{p}$. Chữ cái đầu tiên của văn bản được ứng với một khoảng nhất định $[\delta_{j-1}, \delta_j)$ nào đó. Để tiến hành mã chữ cái



Hình 3.2

tiếp theo ta coi khoảng xác định là [a,b) và tiến hành mã tiếp tục như vậy cho đến hết văn bản.

Văn bản được ứng với dãy các khoảng lồng nhau $\Delta^0 \supset \Delta^1 \supset \Delta^2 \supset ... \supset \Delta^m$ bắt đầu là [0,1) với n là độ dài của văn bản. Để tìm lại văn bản ta chỉ cần một lượng thông tin đủ để xác định lại các "khoảng xác định" ấy. Vì các khoảng này lồng nhau cho nên để xác định lại văn bản ta chỉ cần biết 1 điểm chung của chúng, cùng với độ dài của văn bản.

3.3. Thuật toán mã số học

Như vậy, mã nén số học là biến một văn bản thành một số trong nửa đoạn [0, 1), sao cho số ứng với mỗi văn bản có số chữ số có nghĩa ít nhất.

Phép nén số học là một đơn ánh từ tập văn bản T vào [0, 1).

C : T -> [0, 1)

Cho trước:

- Văn bản T được xây dựng từ tập chữ cái A = {a_1, a_2, ..., a_k} có k phần tử.
- Bảng tỷ lệ p = p_1 : p_2 : ... : p_k ứng với bảng chữ cái A = a_1, a_2, ..., a_k.

- **Phép mã**

Mã văn bản t = t_1, t_2, ..., t_n, lần lợt đợc xác định từ các chữ cái t_i, bằng cách chọn các khoảng lồng nhau tương ứng với từng chữ cái t_i. Cụ thể nh sau:

Với chữ cái đầu tiên t_1.

- Bước 1: Chia [0, 1) thành k phần theo tỷ lệ p cho trớc [$\alpha_0 = 0$, α_1), [α_1, α_2), ..., [αn-1, 1 = αn). Mỗi nửa đoạn [αi, αi+1) đặt tương ứng với một chữ cái ai.

- Bước 2: Nếu $t_1 = a_{i1}$, chọn đoạn [α_{i1}, α_{i1+1}).

Với chữ cái thứ 2 quay lại bớc 1 nhưng với đoạn [α_{i1}, α_{i1+1}).

Cứ tiếp tục nh vậy đến hết văn bản. Cuối cùng bản mã là số α_{in}.

Như vậy ta được các khoảng lồng nhau: [α_{i1}, α_{i1+1}) $\supset$ [α_{i2}, α_{i2+1}) $\supset$ ... $\supset$ [α_{in}, α_{in+1}).

Ví dụ 3.2:

Có bảng chữ cái gồm 10 phần tử: a, b, c, d, e, f, g, h, i, j.

Bảng tỷ lệ: p_a: p_b: ...: p_j = 1:1:1:1:1:1:1:1:1:1.

Cần mã văn bản: “bacga”.

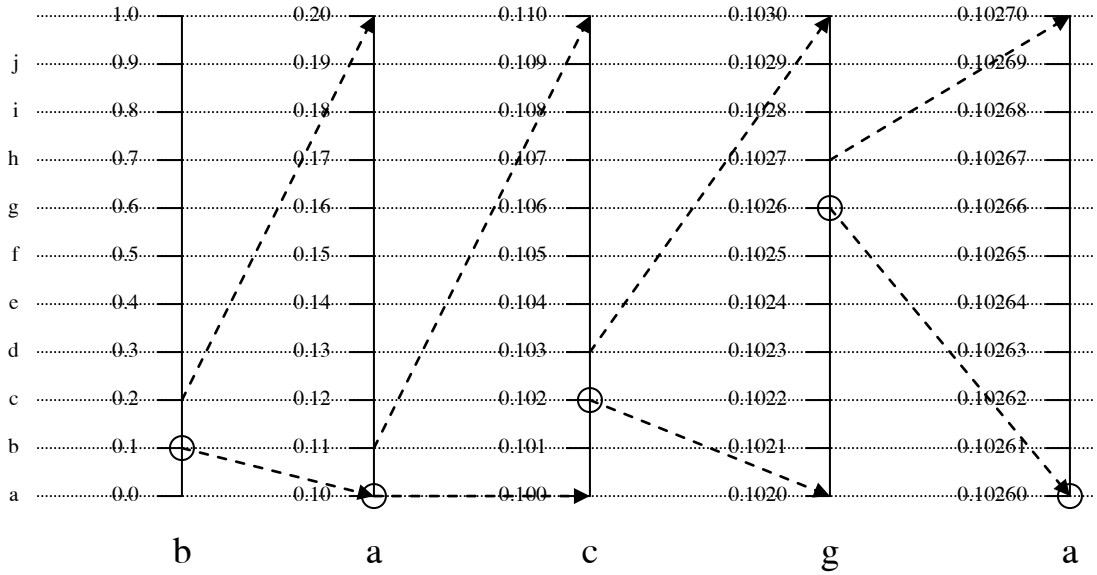
- Chia đoạn [0, 1) thành 10 đoạn theo tỷ lệ trên, chữ “b” nằm trong [0.1, 0.2) nên chọn đoạn [0.1, 0.2) để mã tiếp. (bản mã văn bản một chữ “b” là số 0.1).

- Chia đoạn [0.1, 0.2) thành 10 phần nh trên, chữ “a” nằm trong [0.10, 0.11), nên lấy đoạn [0.10, 0.11) để mã tiếp văn bản. (bản mã văn bản “ba” là số 0.10).

- Chia đoạn [0.10, 0.11) thành 10 phần nh trên, chữ “c” nằm trong [0.102, 0.103), nên lấy đoạn [0.102, 0.103) để mã tiếp văn bản. (bản mã văn bản “bac” là số 0.102).

- .....

- Cuối cùng bản mã văn bản "bacga" là số 0.1026.



Hình 3.3

- **Giải mã**

Giải mã văn bản T từ bản mã là số s thực chất là xác định dãy $[\alpha_i, \alpha_{i+1})$ lồng nhau. Việc thực hiện cụ thể nh sau:

- Bước 1. Chia đoạn [0, 1) thành k phần theo tỷ lệ p đợc các mốc $0 = \alpha_0 < \alpha_1 < ... < \alpha_n = 1$.

- Bước 2. $s \in [0, 1)$ nên tồn tại i: $s \in [\alpha i1, \alpha i1+1)$, từ đó xác định đợc chữ cái đầu tiên là ai.

- Lặp lại từ bước 1 nhưng với đoạn $[\alpha i1, \alpha i1+1)$ ta được các chữ cái tiếp theo.

Chú ý việc lặp này sẽ thực hiện vô hạn nếu như không biết độ dài văn bản.

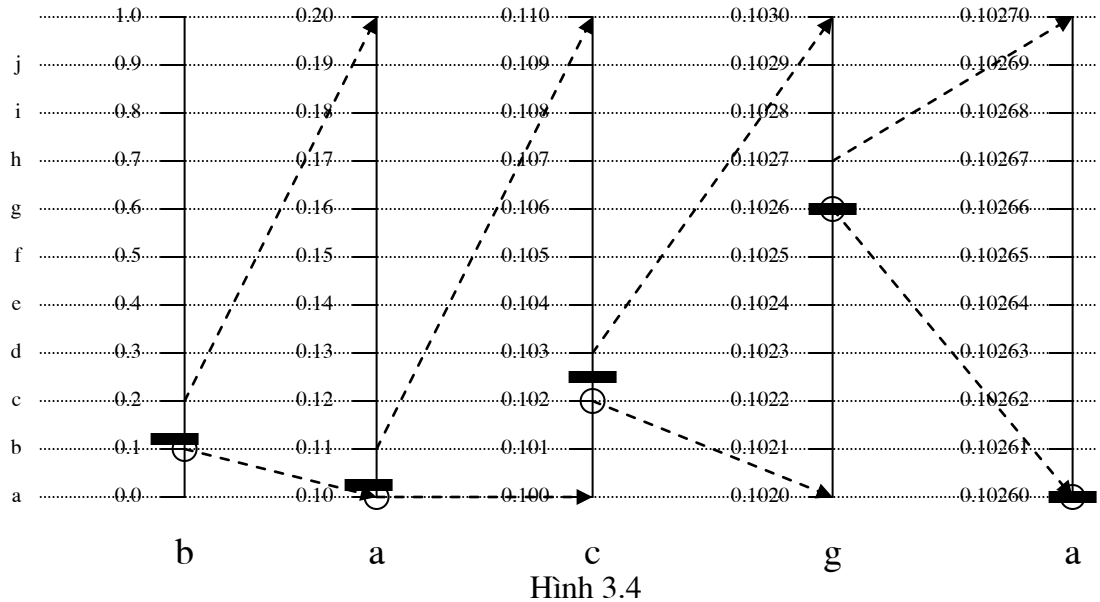
Ví dụ 3.3. Giải mã văn bản từ bản mã là số s = 0.1026, với bảng chữ cái và tỷ lệ trên.

- Chia đoạn [0, 1) thành 10 đoạn bằng nhau 0.0, 0.1, 0.2, ...,0.9, 1.0. Vì 0.1 <= s = 0.1026 < 0.2 nên xác định được chữ cái đầu tiên là chữ "b". Và khoảng tiếp theo là [0.1, 0.2).

- Chia đoạn [0.1, 0.2) thành 10 đoạn bằng nhau: 0.10, 0.11, 0.12, ..., 0.19, 0.20. Vì $0.10 \leq s = 0.1026 < 0.11$ nên xác định được chữ cái tiếp theo là chữ "a" – tương ứng với đoạn [0.10, 0.11) trong đoạn [0.1, 0.2) và xác định được đoạn tiếp theo là [0.10, 0.11).

- ....

- Tương tự như vậy ta xác định được dãy chữ cái "bacgaaaaaa...". Nếu lấy với độ dài 5 ta được văn bản "bacga".



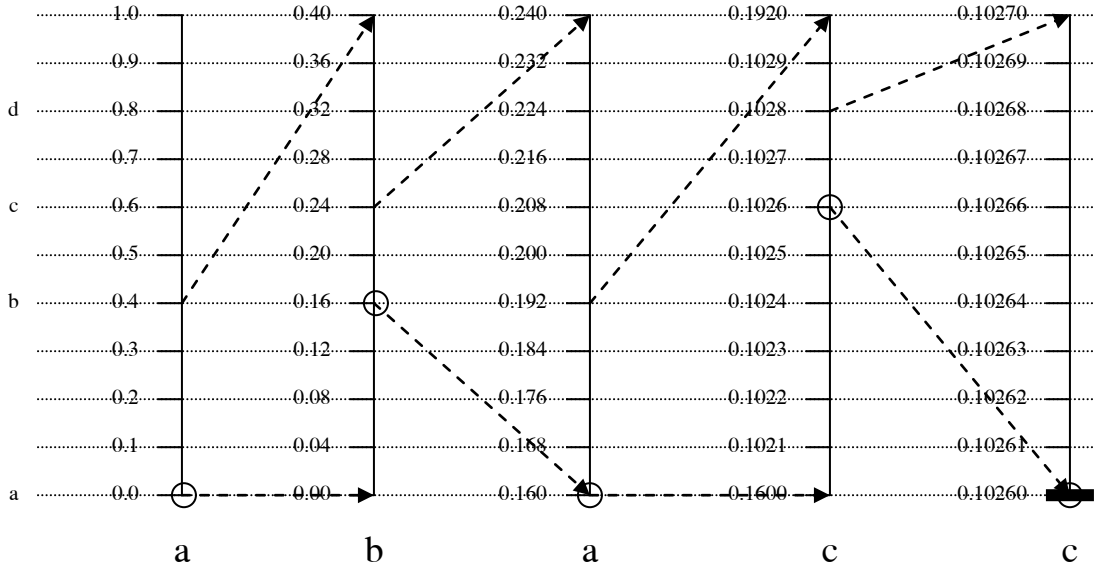
Hình 3.4

Ta thấy rằng để biểu diễn một đoạn càng nhỏ cần các số có số chữ số có nghĩa càng lớn. Vì vậy việc chia khoảng theo tỷ lệ nào đó sao cho đoạn lồng nhau giảm ít nhất. Để đạt được điều đó ta chia các đoạn $[\alpha_i, \alpha_{i+1})$ theo tỷ lệ tần suất xuất hiện các chữ cái. Những chữ cái xuất hiện nhiều được chia một khoảng lớn hơn, những chữ cái xuất hiện ít được đặt tương ứng với một khoảng nhỏ hơn.

Ví dụ 3.4.

- Cho bảng chữ cái a, b, c, d, e, f, g, h, i, j.
- Bảng tần suất $p_a = 40\%$, $p_b = 20\%$, $p_c = 20\%$, ...

Văn bản: abacab được mã theo sơ đồ sau:



Hình 3.5

- Chương trình.

Ví dụ 3.5.

Trong ví dụ này không gian xác suất có 10 phần tử có kí hiệu là 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 có giá trị xác suất tương ứng như sau 0.492, 0.058, 0.060, 0.077, 0.036, 0.049, 0.018, 0.121, 0.036, 0.052. Các văn bản được tạo ra bằng cách chọn ngẫu nhiên các chữ cái.

var

```
    P : array[1..11] of extended;
    H, L, S, rank : extended;
    j : longint;
Begin
  P[1]:=0.492; P[2]:=0.058; P[3]:=0.060; P[4]:=0.077; P[5]:=0.036;
  P[6]:=0.049; P[7]:=0.018; P[8]:=0.121; P[9]:=0.036; P[10]:=0.052;
  P[11]:=1.0; for j:=10 downto 1 do P[j]:=P[j+1]- P[j];
  {Tìm khoảng biểu diễn}
  L:=0;rank:=1;writeln;randomize;
  repeat
     S:=random;
     j:=10;while P[j]>S do j:=j-1;
     L:=L+P[j]*rank;
     rank:=rank*(P[j+1]-P[j]);                {Lưu ý H=L+ rank }
     write(j-1);
  until rank<1.0E-17;
  writeln;
  write(L:22:19,' ',L+rank:22:19);writeln;
  {Tìm lại văn bản}
  rank:=1;S:=0;
  repeat
     j:=10;while (j>1)and(S+rank*P[j]>L) do j:=j-1;
     S:=S+rank*P[j]; rank:=rank*(P[j+1]-P[j]);
     write(j-1);
  until rank<1.0E-17;
```

end.

Kết quả tính.

Cột bên trái là dãy các trạng thái, còn bên phải là khoảng tương ứng.

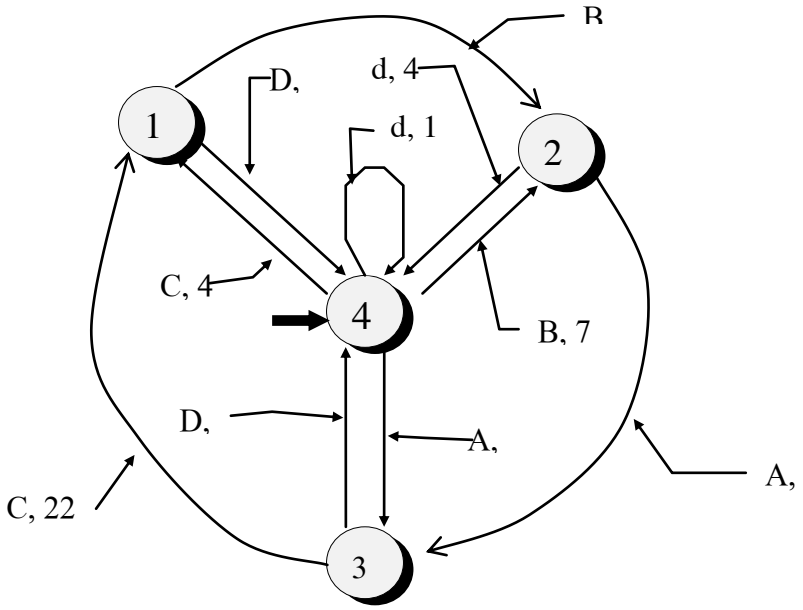
81010007302000998351	[0.929703631878883887, .929703631878883892)
02804002053170007550000	[0.297766190433928610, 0.297766190433928617)
75008751300077000938000	[0.879259863178538067, 0.879259863178538072)

Có thể chọn một điểm nằm trong khoảng tương ứng làm mã

văn bản	bản mã
81010007302000998351	0.92970363187888389
02804002053170007550000	0.297766190433928611
75008751300077000938000	0.87925986317853807

Để mã văn bản dài hơn ta cần thực hiện các phép tính với số thực có độ chính xác cao hơn.

Ví dụ 3.6**.** Biểu diễn văn bản sinh ra do nguồn 4 trạng thái và có entropy bằng 0.95



Hình 3.6

Trình ví dụ biểu diễn nguồn tin. Dịch và chạy trình bằng Pascal. T1 luồng tin đi từ nguồn. [V1,V2) khoảng biểu diễn T1. VL ∈ [V1,V2) một giá trị bất kỳ trong khoảng biểu diễn dùng để giải nén. T2 kết quả giải nén từ VL dùng để so sánh với T1. In 3 kết quả vào file "c:\KQ.TXT".}

```
var
    L,S,Rank: extended;
    V,V1,V2,VL,VH,T1,T2: string;
    state,m,k:integer;
    f:text;
begin
  assign(f,'C:\KQ.TXT');rewrite(f);randomize;
  for k:=1 to 3 do   { thực hiện 3 lần }
    begin
     L:=0;rank:=1;state:=1;T1:='';
     repeat {tạo luồng tin và biểu diễn nó bởi số thực L}
       S:=random;
       if state=1 then
         if S<7/26 then
          begin
            T1:=T1+'d';state:=4;Rank:=Rank*7/26;
         end
        else
          begin
            T1:=T1+'b';state:=2;L:=L+7/26*Rank;Rank:=Rank*19/26;
          end;
        if state=2 then
```

```
        if S<4/26 then
          begin
             T1:=T1+'d';State:=4;Rank:=Rank*4/26;
          end
          else
          begin
             T1:=T1+'a';
             State:=3;L:=L+4/26*Rank;
             Rank:=Rank*22/26;
          end;
    if State=3 then
      if S<7/29 then
        begin
         T1:=T1+'d';State:=4;Rank:=Rank*7/29;
        end
      else
        begin
         T1:=T1+'c';State:=1;L:=L+7/29*Rank;Rank:=Rank*22/29;
        end;
    if State=4 then
      if S<4/20  then
        begin
           T1:=T1+'c';State:=1;Rank:=Rank*4/20;
        end
        else
          if S<5/20  then
```

```
            begin
          T1:=T1+'d';
          State:=4;L:=L+4/20*Rank;Rank:=Rank*1/20;
            end
          else if S<12/20 then
            begin
              T1:=T1+'b'; State:=2;L:=L+5/20*Rank;Rank:=Rank*7/20;
            end
           else
            begin
              T1:=T1+'a';State:=3;L:=L+12/20*Rank;Rank:=Rank*8/20;
            end;
until rank <1.0E-17; {phụ thuộc vào độ chính xác của số thực trong tính toán}
Str(L:22:19,V1);Str(L+Rank:22:19,V2);V:='['+V1+V2+')';
Str(L+Rank/2:22:19,VL);Val(VL,L,m);
{Tạo lại luồng từ giá trị L}
State:=1;S:=0;Rank:=1;T2:='';
repeat
  if State=1 then
   if (L-S)/Rank<7/26 then
    begin
      T2:=T2+'d';State:=4;Rank:=Rank*7/26;
    end
     else
       begin
         T2:=T2+'b';State:=2;S:=S+7/26*Rank;Rank:=Rank*19/26;
```

```
        end;
    if State=2 then
     if (L-S)/Rank<4/26 then
       begin
         T2:=T2+'d';State:=4;Rank:=Rank*4/26;
       end
         else
          begin
            T2:=T2+'a';State:=3;S:=S+4/26*Rank;Rank:=Rank*22/26;
          end;
    if State=3 then
        if (L-S)/Rank<7/29 then
          begin
            T2:=T2+'d';State:=4;Rank:=Rank*7/29;
          end
          else
            begin
              T2:=T2+'c';State:=1;S:=S+7/29*Rank;Rank:=Rank*22/29;
            end;
    if State=4 then
        if (L-S)/Rank<4/20  then
          begin
             T2:=T2+'c';State:=1;Rank:=Rank*4/20;
          end
            else
              if (L-S)/Rank<5/20  then
```

```
                begin
                  T2:=T2+'d';State:=4;S:=S+4/20*Rank;Rank:=Rank*1/20;
                end
                else
                 if (L-S)/Rank<12/20 then
                  begin
                     T2:=T2+'b';State:=2;S:=S+5/20*Rank;Rank:=Rank*7/20;
                  end
                  else
                   begin
                     T2:=T2+'a';
                     State:=3;S:=S+12/20*Rank;
                     Rank:=Rank*8/20;
                   end;
        until rank<1.0E-17;
        writeln(f,T1);writeln(f,T2);writeln(f,VL);writeln(f,V);
        end;
        close(f);
end.
```

Kết quả tính.

ddbacdcdcbacbacbacbacbacdcdcbacdcbacddbacbacddbacbacbac

[0.058907264212235128, 0.058907264212235135)

bacdcbacbacbacddbdcdcbacbacdbdcdcbacbacdcbacbacbacbacbacdcbac

[0.553712632942476848, 0.553712632942476853)

dcbacbacbacdcbacbacbacbacbacbacbacdcdcddbdcdcbacbacbacbacbacbacdcbac

[0.048585947986781746, 0.048585947986781755)

bacbacbacdcbacbacbacbacbacbacdcbacbacdcbacbacbacbacbacbacbacbacbacba
cbacbacbacbacbacdbdcdcbacbacbac
[0.9022815439583768 85, 0.9022815439583768 93)

Chương 4. Mã LZW

4.1 Ngyên lý mã theo từ điển (Nguyên lý LZ)

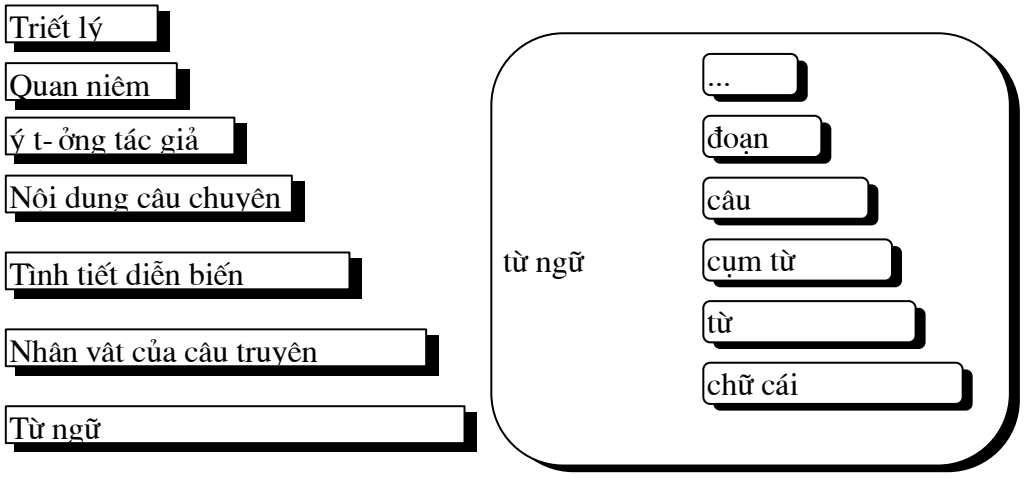
Bản chất của nén tin là làm thế nào để có thể dự đoán càng đúng càng tốt kí tự sẽ xuất hiện tiếp theo là kí tự nào. Nếu đoán được chính xác hoàn toàn chữ sẽ xuất hiện thì lẽ dĩ nhiên ta không cần phải tốn "giấy mực" để ghi nó ra nữa. Đoán đúng được chút nào thì vẫn có lợi chút ấy. Đứng đằng sau khả năng đoán nhận là tính qui luật hay nói cách khác là sự phụ thuộc, mà entropy là số đo của nó. Entropy càng nhỏ thì khả năng nén một văn bản càng lớn.

Có hai bài toán đặt ra:

Tính entropy như thế nào?

Lập mã và giải nén thế nào?

Tính entropy, tức là tìm ra lượng tin. Đối với một quyển truyện thì lượng tin phân bố theo các tầng kiến tạo tin sau đây.



Hình 4.1

Phân tích chi tiết hơn tầng từ ngữ bao gồm các lớp phụ thuộc tin như các chữ cái, qui tắc tạo từ, qui tắc tạo cụm từ, qui tắc tạo câu, qui tắc tạo đoạn văn, qui tắc tạo chương, ... Về mặt nguyên lý để tính được lượng tin có trong một văn bản một cách tốt nhất ta cần phải có thuật toán phân tích tìm ra mô hình phụ thuộc phân tầng nói trên. Việc làm này rất khó, chính vì vậy mà người ta

thường tạo ra thuật toán nén dựa trên sự lý giải có lý. Sự thật thì khi dự đoán ta luôn sử dụng những điều đã xảy ra trong quá khứ. Vậy thì có lẽ ta nên nghĩ rằng sự phụ thuộc thông tin trong hiện tại có lẽ cũng sẽ na ná như lúc nào đó trong quá khứ. Phương pháp thay một đoạn kí tự bởi toạ độ của một đoạn giống hệt nó trong quá khứ - đoạn copy, được gọi là nguyên lý LZ, do Jacob Ziv và Abraham Lempel phát triển năm 1977. Việc tính toán để dự báo cái gì sắp xảy ra được chuyển về sự tương tự với cái đã xảy ra trong quá khứ, vì thế việc tính toán để tìm các thông số phục vụ cho một thuật toán nén là không còn cần thiết nữa. Như vậy, tư duy thì thông qua các khái niệm, và khái niệm lại được thể hiện thông qua từ ngữ. Khái niệm là các yếu tố tương đối ổn định dẫn tới ngôn ngữ thường dùng có rất nhiều từ và cụm từ lặp đi lặp lại. Vì lý do đó mà thay vì phải tìm cách ghi nhận khái niệm, người ta nghĩ tới việc mô tả một từ hay cụm từ theo vị trí của nó đã gặp ở đâu đó. Tư duy như thế cũng đáp ứng được phần nào mô hình phụ thuộc phân tầng.

Như vậy, nguyên lý LZ là thay một đoạn copy bởi toạ độ của nó. Sử dụng nguyên lý này ta thu được một văn bản gồm các toạ độ thay vì chính văn bản ấy. Khả năng nén được của loại thuật toán này là ở chỗ, lưu trữ toạ độ thì tiết kiệm hơn là lưu trữ bản thân đoạn copy.

4.1.1. Từ điển.

Dựa vào từ điển một văn bản có thể được phân ra thành nhiều cụm từ và văn bản được ký mã lại bằng toạ độ của các cụm từ cùng với độ dài của cụm từ. Điều phải quan tâm nhất là làm sao chọn được **quyển từ điển tốt** và tìm được **cụm từ dài nhất** nếu có thể. Từ điển mà các thuật toán LZ sử dụng có đặc điểm

Từ điển có thể tĩnh hoặc động.

Từ điển được xây dựng tuỳ thuộc vào từng văn bản cụ thể.

- **Mã với từ điển tĩnh.**

Mã với từ điển tĩnh không khác phương pháp kí hiệu là mấy. Có điều ở đây tiến hành kí hiệu một cách có hệ thống tất cả các đoạn copy.

- **Mã với từ điển động.**

Mã với từ điển tĩnh thì độ dài đoạn copy càng lớn thì càng có khả năng nén tốt hơn. Bởi vậy cần sử dụng từ điển lớn. Tuy nhiên để giải mã văn bản ta cần không chỉ bản mã mà còn cần cả từ điển. Việc lưu giữ một từ điển lớn cùng với bản mã nén là không kinh tế. Vì tổng cộng dung lượng của từ điển phải lưu để giải nén và dung lượng của bản mã nén có khi lớn hơn cả dung lượng của văn bản ban đầu. Chính vì thế mà tư tưởng chủ đạo của mã nén LZ động là sử dụng ngay văn bản gốc làm từ điển. Đặt giả sử như ta đã mã đến ký tự thứ m và chuẩn bị mã cho các kí tự tiếp theo. Ta có thể dùng toàn bộ quá khứ của văn bản làm từ điển. Khi đó cả lúc mã lẫn lúc giải mã ta đều có ngay từ điển mà không cần phải lưu giữ chúng. Như vậy từ điển lớn dần và đoạn copy có độ dài trung bình là $\frac{\log_2 m}{h}$ cũng lớn dần, việc này dẫn đến khả năng nén càng ngày càng hiệu quả hơn.

4.1.2. Khái quát hoá thuật toán LZ.

Văn bản $\overline{z_1.....z_k z_{k+1}...}$.

- **Từ điển tĩnh** là một xâu có dạng $\overline{s_1 s_2 s_3.....s_n}$ gồm n ký tự lấy từ cùng một bảng chữ cái tạo nên văn bản $\overline{z_1.....z_k z_{k+1}...}$.

n←1

Tìm L là số lớn nhất có thể sao cho tồn tại m mà $\overline{z_{n+1} z_{n+2}...z_{n+L}} = \overline{s_{m+1} s_{m+2}.....s_{m+L}}$

Đoạn văn bản $\overline{z_n z_{n+1}...z_{n+L}}$ tương đương với (L,m).

Thay n←n+L

Lặp lại quá trình trên đến hết văn bản.

Kết quả $\overline{s_1 s_2 s_3 .....s_n}$ và (m_1, L_1) $(m_2, L_2)(m_3, L_3)$..... (m_r, L_r) được gọi là bản mã nén của $\overline{z_1 .....z_k z_{k+1} ...}$

- **Từ điển động**

  Văn bản $\overline{z_1 .....z_k z_{k+1} ...}$.

  $n \leftarrow n_0$,

  Tìm L là số lớn nhất có thể sao cho tồn tại m < n mà

  $$\overline{z_{n+1} z_{n+2} ...z_{n+L}} = \overline{z_{m+1} z_{m+2} .....z_{m+L}} .$$

  Khi đó văn bản $\overline{z_1 .....z_n z_{n+1} ...z_{n+L}}$ tương đương với $\overline{z_1 .....z_n}$ và (L,m).

  Thay $n \leftarrow n+L$

  Lặp lại quá trình cho đến hết văn bản.

  Dãy $\overline{z_1 z_2 .....z_{n_0}}$ và (m_1,L_1) $(m_2,L_2)(m_3,L_3)$..... (m_r,L_r) được coi là bản mã nén của văn bản.

Sau khi sử dụng các thuật toán trên người ta tiến hành mã các cặp (m,L) bằng các đoạn bit 0/1 phân tách và nếu có thể thì thực hiện thống kê tần số để tìm ra các mã bit 0/1 đó theo một trong số các cách như ta đã xét (Huffman, Fano,...).

4.1.3. Các công đoạn thực hiện khi mã bằng LZ.

Mã thuộc họ LZ được tiến hành thông qua 3 giai đoạn. Giai đoạn 1 là cắt khúc văn bản theo một nguyên lý nào đó. Giai đoạn 2 ký mã các khúc. Giai đoạn 3 là sử dụng một tập phân tách để mã. Không thể có một cơ sở lý luận chắc chắn nào cho phép tìm thuật toán cắt khúc tốt nhất. Ta cho rằng nếu xuất phát từ việc tìm lại sự tương tự như trong quá khứ thì nên tìm sự tương tự nào gần giống nhất. Chính vì thế mà các khúc khi cắt ra từ văn bản nguồn nên được tính toán sao cho chúng là các phần dài nhất có thể tìm thấy trong quá khứ.

Thuật toán cắt khúc (parsing algorithm) đóng một vai trò quan trọng trong các thuật toán LZ.

1. Cắt khúc và toạ độ hoá theo từ điển.

2. Mã hoá dãy toạ độ thu được bằng một mã nén đó.

3. Đóng gói thành các byte.

Ví dụ 4.1. Cắt khúc văn bản theo từ điển.

- **Với từ điển tĩnh**: Ta hãy cắt khúc một dãy

"0100111101011001000010011000010011110101100"

Theo từ điển "10000100111101011 00". Ta sẽ thu được 3 khúc như sau: <5,15> <1,10> <2,18>. ở đây 5 là vị trí bắt đầu của xâu con, 15 là số bit tiếp theo kể cả bit đầu tiên. Cụ thể các khúc đó bao gồm các đoạn bit sau

<5,15> là dãy con 010011110101100 trong từ điển 1000**010011110101100**

<1,10> là dãy con 1000010011 trong từ điển **1000010011**110101100

<2,18> là dãy con 000010011110101100 trong từ điển 1**000010011110101100**

- **Với từ điển động**: xâu sau đây là văn bản ban đầu

"1000011000000011001100000001100001100000001100000001100110"

Giả sử thoạt đầu sử dụng **từ điển có 2 bit 1/0,** và từ điển được kéo dài dần ra bằng cách thêm đoạn văn bản đã được xử lý vào. Kết quả ta thu được cách cắt khúc sau:

<2,1><2,2><1,1><1,4><2,8><6,12><3,13><7,15>

Ví dụ 4.2. Mô tả thuật toán.

Công đoạn 1. Toạ độ hoá theo từ điển (giống như ta vừa mô tả ở trên). Tức là ta chuyển văn bản

"1000011000000011001100000001100001100000001100000001100110"

thành dãy các toạ độ:

<2,1><2,2><1,1><1,4><2,8><6,12><3,13><7,15>

Công đoạn 2. Mã hoá dãy toạ độ thu được bằng một mã nào đó. Trong ví dụ trên, ta chuyển mỗi số thành 4 bit và thu được

"0100 1000 0100 0100 1000 1000 1000 0010 0100 0001 0110 0011 1100 1011 1110 1111"

Công đoạn 3. Đóng gói thành các byte.

4.2. Thuật toán nén LZW

Các thuật toán LZ khác nhau có được là do cách xây dựng từ điển và hàm mã (cách biểu diễn) khác nhau. Ta xuất phát điểm là một từ điển nhỏ, trong quá trình mã dần dần văn bản được mô tả thông qua từ điển mà bây giờ từ điểm bao gồm phần từ điển ban đầu cộng với phần văn bản đã được mã. Từ điển cứ lớn dần lên nhưng không thể lớn mãi được. Trong phần này ta chỉ nghiên cứu giải thuật nén LZW mà tiền thân của nó là LZ78,

4.2.1. LZ78

Thay vì thông báo vị trí đoạn văn lặp lại trong quá khứ mã LZ78 đánh số tất cả các đoạn văn sao cho mỗi đoạn ghi nhận số hiệu đoạn văn lặp lại trong quá khứ cộng với một kí tự mới ngay sau nó. Một dãy các ký tự của một đoạn như vậy được gọi là một đoạn copy. Như vậy đoạn copy tại một thời điểm nào đó là một khúc các kí tự liền nhau khi dịch khúc văn bản này vào quá khứ thì sẽ có một thời điểm mà trừ kí tự cuối cùng ra nó trùng với một đoạn copy nào đó của văn bản. Không nhất thiết mọi kí tự của đoạn copy nằm trong quá khứ. Ta kí hiệu một đoạn copy mới dưới dạng một tổng bao gồm đoạn copy cũ và kí tự mới.

Ví dụ 4.3. Mẩu chữ "aaabbabaabaaabab" phân thành các đoạn copy sau.

Input	a	aa	b	ba	baa	baaa	bab
đoạn copy number	1	2	3	4	5	6	7

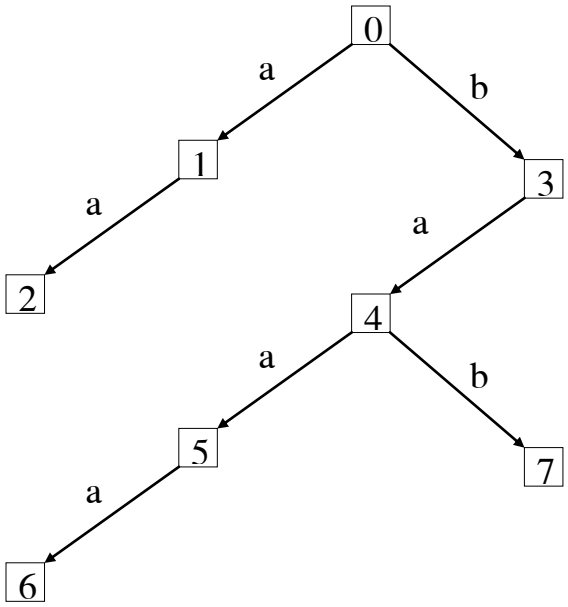
Output	0+a	1+a	0+b	3+a	4+a	5+a	4+b

Bảng 4.1

Bản mã nén gồm (0,a)(1,a)(0,b)(3,a)(4,a)(5,a)(4,b)

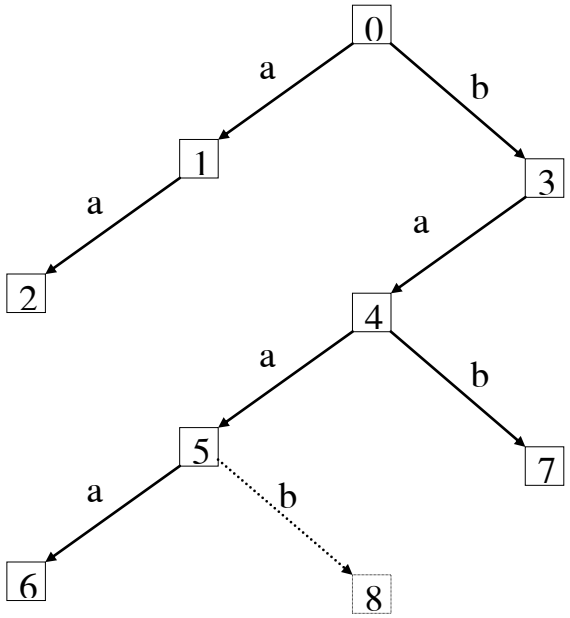
Tiếp theo đóng gói dãy trên theo cách sử dụng tập phân tách để mã các con số còn các chữ thì dùng mã có độ dài cố định 1 byte.

Thuật toán trên có thể biểu diễn thông qua sơ đồ cây mã và giải mã. Bắt đầu một đoạn copy mới ta đi dọc theo các nhánh cây đến khi nào không đi được nữa thì sẽ xuất hiện một nhánh mới và trên nhánh đó có dán một chữ cái mới là chữa cuối cùng của đoạn copy.



Hình 4.2

Ví dụ đỉnh 4 là đoạn “ba” hay là 3+a, đỉnh 7 là đoạn “bab” hay là 4+b. Như vậy bản mã nén của văn bản là một đồ thị định hướng có các đỉnh là số thứ tự của các đoạn và các cạnh là các kí tự tiếp theo của đoạn. Giả sử ta có đoạn kí tự tiếp theo là baab khi đó cây đẻ ra thêm một nhánh mới sau



Hình 4.3

Thuật toán nén và giải nén. Ký tự số 0 là ký tự không có gì (empty).

- **Quá trình nén**

Từ trái qua phải tìm tất cả các đoạn copy và thay nó bằng cách biểu diễn dưới dạng tổng.

aaabbabaabaaabab

Input	a
đoạn copy number	1
Output	0+a

aaabbabaabaaabab

Input	a	aa
đoạn copy number	1	2
Output	0+a	1+a

aaabbabaabaaabab

Input	a	aa	b
đoạn copy number	1	2	3
Output	0+a	1+a	0+b

aaabbabaabaaabab

Input	a	aa	b	ba
đoạn copy number	1	2	3	4
Output	0+a	1+a	0+b	3+a

aaabbabaabaaabab

Input	a	aa	b	ba	baa
đoạn copy number	1	2	3	4	5
Output	0+a	1+a	0+b	3+a	4+a

aaabbabaabaaabab

Input	a	aa	b	ba	baa	baaa
đoạn copy number	1	2	3	4	5	6
Output	0+a	1+a	0+b	3+a	4+a	5+a

aaabbabaabaaabab

Input	a	aa	b	ba	baa	baaa	bab
đoạn copy	1	2	3	4	5	6	7

number

Output 0+a 1+a 0+b 3+a 4+a 5+a 4+b

Kết quả mã aaabbabaabaaabab →(0+a)(1+a)(0+b)(3+a)(4+a)(5+a)(4+b)

Giải mã được tiến hành thông qua việc thay liên tiếp các tổng bằng các đoạn copy. Mỗi lần thay ta nhận được một đoạn copy mới (số thứ tự của cột ở dòng thứ 2) cho nên trong quá trình thay các phần số của tổng luôn nhỏ hơn số thứ tự của cột mà nó đứng. Chính vì thế mà ta luôn giải nén được.

4.2.2. LZW

Mã LZW giống hệt như LZ78, ngoại trừ kí tự cuối của đoạn copy này là kí tự đầu của đoạn copy tiếp theo. Mỗi đoạn copy thu được do duyệt liên tiếp các kí tự kể từ kí tự đầu tiên của nó (tức là kí tự cuối cùng của đoạn copy trước) cho đến khi, trừ kí tự cuối cùng còn thì nó trùng với một đoạn copy nào đó dài nhất có thể được trước đó. Mỗi đoạn copy như thế ta gọi là một móc xích.

Ta xét sơ đồ mã có cải tiến của LZ78 trong đó output chỉ là số hiệu các đoạn copy chứ không có các chữ nữa.

Ví dụ 4.4. Xét mã nén sau

văn bản

aababababaaababb

từ điển

0	a
1	b

Ta lần lượt tách các móc xích ra khỏi xâu aababababaaababb và đưa vào từ điển. Từ điển sẽ lớn dần lên.

aababababaaababb

aababababaaababb

0	a	
1	b	
2	aa	0+a
3	ab	0+b
4	ba	1+a
5	aba	3+a
6	abaa	5+a

aababababaaababb

aababababaaababb

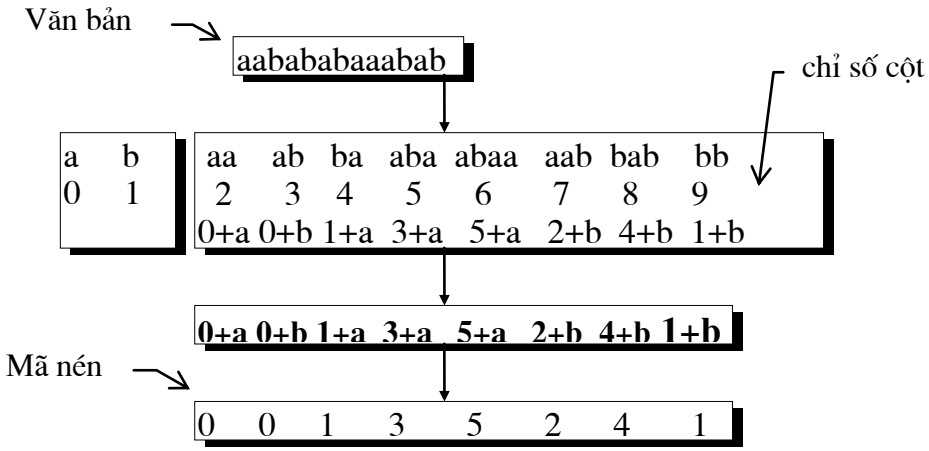
aababababaaababb

aababababaaababb

aababababaaababb

aababababaaababb

Quá trình nén văn bản



4.2.3. Thuật toán nén LZW

Bước 1 Cắt văn bản mới thành các đoạn copy. Nếu bảng chữ cái có m chữ thì các chữ cái là m đoạn copy đầu tiên được đánh số từ 0 đến m -1.

Bước 2 Bỏ tất cả phần chữ ta thu được mã nén.

Lưu ý rằng các đoạn copy lần lượt được tạo ra và phần số của nó luôn nhỏ hơn số hiệu cột mà nó đứng.

4.2.4. Thuật toán giải nén LZW.

Bắt đầu là các cột đầu tiên (trong ví dụ là cột thứ 2) lặp lại thao tác sau cho đến hết.

Lấy hai số liên tiếp của bản mã ví dụ là X, Y thay nó về dạng X+? và Y+$. Trong đó kí tự đầu tiên của Y+$ là kí tự cuối cùng của X+?. Dấu ? và $ là thay cho một kí tự chưa biết. Vì X và Y không thể lớn hơn chỉ số cột mà nó đứng cho nên ta hoàn toàn tìm được giá trị đoạn copy ứng với cột có chỉ số X, Y và thay đoạn copy vào X+? và Y+$ tương ứng. Giá tri ? là kí tự đầu của Y+$ cho nên luôn luôn xác định. Như thế ta tìm được X+?.

Ví dụ 4.5. Nén theo LZW

Bước 1 aababababaaababb

thay a→0

được 0abababaaababb

từ điển

0	a	
1	b	

đoạn copy mới aababababaaababb

Bước 2 aababababaaababb

thay a→0

được 00bababaaababb

từ điển

0	a	
1	b	
2	aa	0+a

đoạn copy mới aababababaaababb

Bước 3 00bababaaababb

thay b→1

được 001ababaaababb

từ điển

0	a	
1	b	
2	aa	0+a
3	ab	0+b

đoạn copy mới aababababaaababb

Bước 4 001ababaaababb

thay ab→3

được 0013abaaababb

từ điển

0	a	
1	b	
2	aa	0+a
3	ab	0+b
4	ba	1+a
5	aba	3+a

đoạn copy mới aababababaaababb

0	a	
1	b	
2	aa	0+a
3	ab	0+b
4	ba	1+a
5	aba	3+a
6	abaa	5+a

Bước 5 0013abaaababb

thay aba→5

được 00135aababb

từ điển

đoạn copy mới aababababaaababb

0	a	
1	b	
2	aa	0+a
3	ab	0+b
4	ba	1+a
5	aba	3+a
6	abaa	5+a
7	aab	2+b

Bước 6 00135aababb

thay aa→2

được 001352babb

từ điển

đoạn copy mới aababababaaababb

0	a	
1	b	
2	aa	0+a
3	ab	0+b
4	ba	1+a
5	aba	3+a
6	abaa	5+a
7	aab	2+b
8	bab	4+b

Bước 7 001352babb

thay ba→4

được 0013524bb

từ điển

đoạn copy mới aababababaaababb

0	a	
1	b	
2	aa	0+a
3	ab	0+b
4	ba	1+a
5	aba	3+a
6	abaa	5+a
7	aab	2+b
8	bab	4+b
9	bb	1+b

Bước 8 0013524bb

thay b→1

được 00135241b

từ điển

đoạn copy mới aababababaaababb

Bước 9 00135241b

thay b→1

được 001352411

từ điển

đoạn copy mới aababababaaababb

0	a	
1	b	
2	aa	0+a
3	ab	0+b
4	ba	1+a
5	aba	3+a
6	abaa	5+a
7	aab	2+b
8	bab	4+b
9	bb	1+b

Kết quả nén của aababababaaababb là 0 0 1 3 5 2 4 1 1

Trình ví dụ nén theo thuật toán LZW.

```
Const
   Z:string = 'aababaaaaababbbbaaaaaababababaaaaababababbbbb';
Label BD;
Var
  S :array[255..1000] of word;
  C :array[255..1000] of char;
  X:word; a:char;
  n,m,H:word;
  W:string;
Begin
{Nen}
H:=255; n:=1; x:=ord(Z[1]);
Repeat
  BD: n:=n+1; a:=Z[n];
    for m:=256 to H do
```

```
        if (S[m]=x)and(C[m]=a) then
          begin
            x:=m;Goto BD;
          end;
    write(x,'.');
    H:=H+1; S[H]:=x; C[H]:=a;
    x:=ord(a);
Until n > Length(Z);
{Giai nen}
writeln;
write(chr(S[256]));
for m:=257 to H do
   begin
     W:='';x:=S[m];while x>255 do Begin W:=C[x]+W;x:=S[x]end;
     C[m-1]:=char(x); W:=char(x)+W;
     if S[m]=m-1 then W[length(W)]:=char(x);
     write(W);
   end;
end.
```

Nén file (không lớn lắm)

```
Label BD;
Var
```

```
  S :array[255..15000] of word;C :array[255..15000] of byte;
  X:word; a:byte;
  m,n,i,H:Longint;
  W:array[1..100] of byte;
  f:file of byte;
  g:file of word;
Begin
  assign(f,'C.txt');reset(f);
  H:=255; read(f,a); x:=a;
  while not eof(f) do
  begin
  BD:read(f,a);
  for m:=256 to H do if (S[m]=x)and(C[m]=a) then begin x:=m;Goto BD;end;
 H:=H+1; S[H]:=x; C[H]:=a; x:=a;
end;
close(f);
assign(g,'C.nen');rewrite(g);
for m:=256 to H do write(g,S[m]);x:=a;write(g,x); close(g);
assign(g,'C.nen'); reset(g); assign(f,'D.txt'); rewrite(f);
read(g,S[256]); write(f,byte(S[256]));
m:=256;
```

```
while not eof(g) do
 begin
    m:=m+1;read(g,S[m]);
   x:=S[m]; n:=1;
   while x>255 do begin W[n]:=C[x]; x:=S[x]; n:=n+1; end;
   C[m-1]:=byte(x); W[n]:=byte(x);
   if S[m]=m-1 then W[1]:=byte(x);
   for i:=n downto 1 do write(f,W[i]);
 end;
 close(g);close(f);
end.
```

Kết luận

Luận văn đã hoàn thành được nhiệm vụ đặt ra. Cụ thể là:

- Phân loại văn bản dựa vào sự phụ thuộc tin.
- Đã đưa ra được mô hình Markov dùng để mô phỏng văn bản trong thực tế.
- Dựa vào lý thuyết xác suất và lý thuyết truyền tin, đưa ra giới hạn nén của một văn bản và cách tính entropy (giới hạn nén) của một văn bản dựa trên mô hình Markov.
- Đưa ra một số trình ví dụ để tạo ra văn bản và và tính giới hạn nén văn bản dựa trên mô hình Markov, khẳng định được tính đúng đắn của lý thuyết nén văn bản bằng chương trình.
- Đưa ra một số mã nén và các thuật toán nén văn bản và trình minh họa, giúp cho các nhà lập trình tạo ra các trình nén.

Tuy nhiên, luận văn mới chỉ dừng lại ở nén văn bản dựa trên mô hình Markov hiện và nén là nén bảo toàn. Do đó, luận văn có thể phát triển theo hướng nén không bảo toàn, với các loại dữ liệu khác nhau như hình ảnh, âm thanh,… và nén văn bản dựa trên mô hình Markov ẩn.

Tài liệu tham khảo

Tiếng việt

1. Nguyễn Lê Anh, Trần Duy Lai, Phạm Thế Long, Nguyễn Văn Xuất (2001), *Lý thuyết mã nén*.

Tiếng Anh

2. A.M. Yaglom, I.M. Yaglom (1997), *Giới thiệu lý thuyết thông tin*, Nxb khoa học - Kỹ thuật.
3. Donald Samuel Ornstein and Benjamin Weiss (1993), *Entropy and Data Compression Schemes*, IEEE Transactions on Information Theory, Vol.39, No.1, January , pages 78-83.
4. Gyula O. H. Katona, Tibor O. H. Nemetz (1976), *Huffman Codes and Self-Information*, IEEE Transactions on Information Theory, Vol.22, No.3, May , pages 337-339.
5. Ian H. Witten, Radford M. Neal (1987), and John G. Cleary, *Arithmetic coding for data compression*, Communicatio ns of the ACM, June , Volume 30, Number 6, pages 520-540.
6. I. E. Witten, R. M. Neal, J. G. Cleary (1990), *Text Compression*, Prentice Hall.
7. Nelson Mark (1991), *The Data Compression Book*, M&T Books,
8. Obert J. McEliece (1993), *The Theory of Information and Coding*, Cambridge University Press.