BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á
KHOA: CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN

HỌC PHẦN: CÔNG NGHỆ ĐA PHƯƠNG TIỆN

NHÓM ĐỀ TÀI: ĐỀ TÀI 1: VĂN BẢN VÀ KỸ THUẬT XỬ LÝ VĂN BẢN

Đề tài: 1.1: Kỹ thuật nén văn bản sử dụng mã hóa Huffman

Sinh viên thực hiện	Lớp	Khóa
Nguyễn Trí Dũng	DCCNTT 13.10.16	13
Hoàng Ngọc Thành	DCCNTT 13.10.16	13
Hoàng Văn Nguyên	DCCNTT 13.10.16	13
Trần Thị Hải Yến	DCCNTT 13.10.16	13
Trương Hồng Nhuần	DCCNTT 13.10.16	13

Bắc Ninh, năm 2023

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á
KHOA: CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN

HỌC PHẦN: CÔNG NGHỆ ĐA PHƯƠNG TIỆN

NHÓM ĐỀ TÀI: ĐỀ TÀI 1: VĂN BẢN VÀ KỸ THUẬT XỬ LÝ VĂN BẢN

Đề tài: 1.1: Kỹ thuật nén văn bản sử dụng mã hóa Huffman

STT	Sinh viên thực hiện	Mã sinh viên	Điểm bằng số	Điểm bằng chữ
1	Nguyễn Trí Dũng	20223155		
2	Hoàng Ngọc Thành	20223011		
3	Hoàng Văn Nguyên	20223153		
4	Trần Thị Hải Yến	20222897		
5	Trương Hồng Nhuần	20222540		

CÁN BỘ CHẤM 1
(Ký và ghi rõ họ tên)

CÁN BỘ CHẤM 2
(Ký và ghi rõ họ tên)

LỜI MỞ ĐẦU

Trong lĩnh vực công nghệ đa phương tiện, kỹ thuật nén văn bản sử dụng mã hóa Huffman nổi bật với khả năng giảm kích thước tệp văn bản mà vẫn duy trì tính chất thông tin. Mã hóa Huffman là một phương pháp nén dữ liệu hiệu quả, chủ yếu dựa trên tần suất xuất hiện của các ký tự trong văn bản. Nhờ vào việc áp dụng nguyên tắc "mã ngắn cho phổ biến, mã dài cho hiếm" mã hóa Huffman tạo ra mã đặc trưng cho mỗi ký tự, ưu tiên sự ngắn gọn cho các ký tự xuất hiện thường xuyên, giảm độ dài của dãy mã cho các ký tự ít xuất hiện.

Qua quá trình này, không chỉ tăng tốc quá trình truyền tải dữ liệu mà còn giảm bớt yêu cầu về không gian lưu trữ. Kỹ thuật nén văn bản này đặc biệt quan trọng trong các ứng dụng yêu cầu tốc độ truyền tải nhanh như truyền thông trực tuyến, ứng dụng di động và lưu trữ đám mây. Việc sử dụng mã hóa Huffman trong kỹ thuật nén văn bản không chỉ là sự cải thiện về hiệu suất mà còn là sự đáp ứng linh hoạt đối với nhu cầu ngày càng đa dạng của người sử dụng trong thế giới đa phương tiện ngày nay.

Em xin trân thành cảm ơn cô Nguyễn Bình Hải đã tận tình giúp đỡ nhóm em làm bài tập lớn này. Em xin chân thành cảm ơn khoa Công Nghệ Thông Tin - Trường Đại Học Công Nghệ Đông Á đã tạo điều kiện thuận lợi trong quá trình học tập. Em cảm ơn các cô đã tận tình giảng dạy, trang bị cho chúng em những kiến thức quý báo trong thời gian học vừa qua.

DANH MỤC BẢNG BIỂU VÀ SƠ ĐỒ

DANH MỤC CÁC HÌNH ẢNH

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

1.1 Giới thiệu.

Thế giới công nghệ hiện nay đang trải qua một sự phát triển vượt bậc, đưa ra những đổi mới mạnh mẽ và ảnh hưởng sâu rộng đến mọi khía cạnh của cuộc sống. Công nghệ thông tin và truyền thông đang đóng vai trò quan trọng, từ việc kết nối mọi người trên khắp thế giới đến cách chúng ta làm việc, giáo dục, giải trí và thậm chí là cách chúng ta tương tác với thế giới xung quanh.

Một trong những xu hướng đáng chú ý nhất là sự gia tăng của trí tuệ nhân tạo (AI) và máy học (machine learning). Các hệ thống này đang ngày càng trở nên thông minh và linh hoạt, từ ứng dụng trong y tế, tài chính, đến quản lý nguồn lực và duyệt web. Blockchain, nền tảng công nghệ đằng sau tiền điện tử như Bitcoin, cũng đang tạo ra một cuộc cách mạng trong lĩnh vực tài chính và giao dịch trực tuyến.

Công nghệ 5G đang mở ra những cơ hội mới với tốc độ truyền dẫn dữ liệu nhanh chóng, tạo điều kiện cho sự phát triển của Internet of Things (IoT). Thiết bị kết nối với nhau và truyền thông thông qua mạng lưới 5G, mở ra một thế giới mới với các ứng dụng từ nhà thông minh, ô tô tự lái đến sản xuất thông minh.

Một trong những khía cạnh quan trọng của công nghệ hiện nay là bảo mật thông tin. Khi mà thông tin truyền tải qua các mạng truyền thông và internet ngày càng tăng, việc bảo vệ dữ liệu trở nên càng quan trọng. Một trong những công nghệ đóng góp vào lĩnh vực này là kỹ thuật mã hóa.

Kỹ thuật mã hóa là quá trình chuyển đổi thông tin từ dạng thông thường sang dạng mã hóa, làm cho thông tin trở nên không đọc được nếu không có khóa giải mã. Mã hóa Huffman là một phương pháp mã hóa mà bạn có thể tìm thấy rộng rãi trong việc nén dữ liệu.

Thuật toán Huffman được phát minh bởi David A. Huffman vào năm 1952 và đã trở thành một trong những phương pháp nén dữ liệu hiệu quả nhất. Ý tưởng chính của nó là sử dụng các mã độ dài khác nhau cho các ký tự khác nhau trong dữ liệu, sao cho các ký tự xuất hiện thường xuyên sẽ có mã ngắn hơn và ngược lại. Điều này giúp giảm dung lượng của dữ liệu mà không làm mất mát thông tin.

Mã hóa Huffman đặc biệt hữu ích trong việc truyền tải dữ liệu qua mạng, lưu trữ dữ liệu trên các thiết bị có tài nguyên hạn chế và giảm băng thông cần thiết. Đồng thời nó cũng được sử dụng trong nhiều ứng dụng khác nhau như lưu trữ đám mây, truyền hình số, và nén âm thanh.

Với sự kết hợp của những đổi mới trong lĩnh vực mã hóa và các tiến bộ trong công nghệ thông tin, thế giới ngày nay đang trở nên an toàn hơn và hiệu quả hơn trong việc quản lý thông tin quan trọng. Mã hóa Huffman chỉ là một trong những ví dụ minh họa cho sự đa dạng và tiên tiến của các kỹ thuật bảo mật thông tin trong thời đại công nghệ ngày nay.

1.2 Lý do chọn đề tài.

Chúng em đã chọn đề tài kỹ thuật mã hóa Huffman vì nó là một lĩnh vực quan trọng và thú vị trong lĩnh vực khoa học máy tính. Kỹ thuật này không chỉ là một phần quan trọng của lĩnh vực mã hóa thông tin mà còn đề cập đến khá nhiều khái niệm toán học và thuật toán, điều mà chúng em rất quan tâm và mong muốn tìm hiểu sâu hơn.

Mã hóa Huffman là một phương pháp nén dữ liệu hiệu quả, giúp giảm kích thước của dữ liệu mà vẫn giữ nguyên thông tin quan trọng. Điều này có ý nghĩa lớn trong thế giới ngày nay với sự gia tăng nhanh chóng của dữ liệu và nhu cầu truyền tải thông tin một cách hiệu quả qua các kênh có băng thông hạn chế, nó còn là một phương pháp mã hóa hiệu quả giúp bảo vệ dữ liệu truyền tải và lưu trữ một cách an toàn.

Ngoài ra, việc nghiên cứu và hiểu rõ về kỹ thuật Huffman cũng mang lại cho chúng em kiến thức vững về cấu trúc dữ liệu và thuật toán,nó cũng giúp chúng em hiểu rõ hơn về nguyên lý hoạt động của các thuật toán mã hóa và nâng cao kiến thức về lĩnh vực kỹ thuật máy tính, hai lĩnh vực quan trọng trong ngành công nghiệp công nghệ thông tin. Việc áp dụng được kiến thức này không chỉ giúp chúng em phát triển kỹ năng lập trình mà còn mở ra cơ hội tham gia vào các dự án lớn và thách thức trong lĩnh vực công nghiệp.

Phương pháp Huffman không chỉ được sử dụng trong lĩnh vực bảo mật, mà còn trong nhiều ứng dụng khác như nén dữ liệu và truyền tải dữ liệu trên mạng. Việc nắm vững kỹ thuật này có thể mở ra nhiều cơ hội nghề nghiệp và áp dụng trong nhiều lĩnh vực. Mã hóa Huffman không chỉ là một đề tài lý thuyết mà còn mang lại giá trị thực

tế khi áp dụng vào nhiều ứng dụng thực tế. Việc hiểu rõ về ứng dụng của nó sẽ giúp chúng em thấy rõ giá trị và ý nghĩa của công việc mình đang thực hiện.

Cuối cùng, chúng em tin rằng việc nghiên cứu và triển khai thành công kỹ thuật mã hóa Huffman sẽ giúp chúng em có cái nhìn sâu sắc hơn về cách các hệ thống thông tin hoạt động và làm thế nào chúng có thể được tối ưu hóa để đáp ứng các yêu cầu ngày càng cao của xã hội hiện đại.

CHƯƠNG 2: LÝ THUYẾT TỔNG QUAN VỀ NÉN DỮ LIỆU

2.1 Khái niệm về nén dữ liệu

Nén dữ liệu (tiếng Anh: Data compression) là việc chuyển định dạng thông tin sử dụng ít bit hơn cách thể hiện ở dữ liệu gốc. Tùy theo dữ liệu có bị thay đổi trước và sau khi giải nén không, người ta chia nén thành hai loại: Nguyên vẹn (lossless) và bị mất dữ liệu (lossy). Nén mất dữ liệu giảm số lượng bit bằng cách xác định các thông tin không cần thiết và loại bỏ chúng.

Nén dữ liệu là cần thiết vì giảm được nguồn tài nguyên cũng như dung lượng lưu trữ hay băng thông đường truyền. Tuy nhiên, vì dữ liệu nén cần được giải nén nên sẽ đòi hỏi nhiều phần cứng và xử lý.

2.2 Một số khái niệm cơ bản

2.2.1 Tỉ lệ nén

Tỉ lệ nén là một đo lường cho sự giảm kích thước của một tệp tin hoặc dữ liệu so với kích thước ban đầu. Nó được tính bằng cách so sánh kích thước trước và sau khi áp dụng kỹ thuật nén.

$$\text{Tỉ lệ nén} = (1 - \frac{\text{Kích thước dữ liệu sau khi nén}}{\text{Kích thước dữ liệu ban đầu}}) * 100$$

Tỉ lệ nén thường được biểu diễn dưới dạng một phần trăm hoặc một tỷ lệ số, và càng cao thì dữ liệu đã nén càng hiệu quả.

2.2.2 Độ dư thừa dữ liệu

Trong kỹ thuật nén dữ liệu, độ dư thừa (redundancy) là sự lặp lại hoặc dư thừa của thông tin trong dữ liệu. Đối với nén dữ liệu, mục tiêu chính là giảm độ dư thừa để giảm kích thước của dữ liệu mà vẫn giữ được thông tin cần thiết. Có bốn loại độ dư thừa chính trong kỹ thuật nén dữ liệu:

a, Sự lặp lại của những kí tự

Trong một nguồn dữ liệu, thường có những kí tự và chuỗi kí tự lặp lại nhiều lần liên tiếp nhau. Khi đó, nguồn dữ liệu có thể được mã háo một cách cô đọng hơn bằng cách thay đổi các kí tự đó bằng mã của chúng và số kí tự lặp lại.

b, Sự phân bố của các kí tự

Xét trong một dãy kí tự, ta thường thấy có một số kí tự xuất hiện với tần suất cao hơn so với các kí tự khác. Như vậy, ta có thể giảm bớt lượng dữ liệu bằng cách mã hóa những kí tự xuất hiện thường xuyên với từ mã ngắn, những kí tự ít xuất hiện hơn sẽ được mã hóa bằng những từ mã dài hơn. Kiểu dư thừa này đặc biệt phù hợp với phương pháp mã hóa Huffman.

c, Độ dư thừa vị trí

Có nhiều trường hợp, dữ liệu trong một nguồn dữ liệu có sự phụ thuộc lẫn nhau, do đó nếu biết được kí hiệu xuất hiện tại một vị trí nào đó, ta có thể phỏng đón trước một cách hợp lý sự xuất hiện của các kí hiệu khác ở những vị trí khác nhau.

Ví dụ, ảnh biểu diễn trong một lưới 2 chiều, một số điểm ở hàng dọc lại xuất hiện trong cùng vị trí đó ở các hàng khác nhau. Như vậy, thay vìu lưu trữ dữ liệu ta chỉ lưu lại vị trí hàng và cột. Phương pháp nén khai thác kiểu dư thừa này gọi là phương pháp mã hóa dự đoán.

d, Những mẫu sử dụng mật độ cao

Thông thường, trong các văn bản dạng text, sự tuần tự của những kí tự bào đó sẽ tái xuất hiện với tần suất tương đối cao. Vì vậy, có thể biểu diễn bằng dãy bit ngắn hơn.

Để dánh giá một thuật toán nén có hiệu quả hay không, người ta sẽ dựa vào cách mà thuật toán xử lý các kiểu dư thừa như trên. Thực tế cho thấy rằng, hầu hết các ký thuật nén đều không đủ mềm dẻo để xử lý tất cả các kiểu dư thừa. Mỗi chiến lược nén áp dụng thường chỉ cứng nhắc cho từng kiểu số liệu mà thôi.

Độ dư thừa số liệu có thể định lượng bằng toán học. Với L_1, L_2 là hai đại lượng số liệu cùng được dùng để biểu diễn một lượng tin cho trước thì độ dư số liệu tương đối R_D của tập số liệu thứu nhất so với tập số liệu thứ hai là:

$$R_D = 1 - \frac{1}{\frac{L_1}{L_2}}$$

Trong đó:

L_1 / L_2 được gọi là tỉ lệ nén.

R_D là độ dư số liệu tương đối.

2.2.3 Độ dài trung bình của từ mã

Giá trị trung bình thống kê của tất cả các từ mã trong một bộ mã được gọi là độ dài trung bình của một từ mã. C.E Shannon đã chỉ ra rằng:" Độ dài trung bình của một từ mã không bao giờ nhỏ hơn entropy của nguồn số liệu được mã hóa". Do đó, một bộ mã tối ưu (cho hiệu suất nén cao) là bộ mã có độ dài trung bình của từ mã tiến gần đến Entropy của nguồn số liệu.

2.2.4 Nén số liệu = Mô hình hóa + Mã hóa

Nén số liệu là là quá trình chuyển đổi một luồng các kí hiệu thành một luồng các từ mã tương ứng . Nếu hiệu ứng nén xảy ra thì luồng các từ mã sẽ nhỏ hơn luồng các kí hiệu ban đầu. Việc quyết định đưa ra một từ mã nhất định cho mỗi kí hiệu hoặc một tập kí hiệu dự trên một mô hình.

Mô hình là một tập hợp số liệu các nguyên tắc được sử dụng để xử lý các kí hiệu từ luồng nhập và xuất ra các từ mã, nó có nhiệm vụ xác định xác suất xuất hiện của từng kí tự hoặc chuỗi kí tự và bộ phận mã hóa sẽ tạo ra các từ mã dựa trên xác suất đó.

2.2.5 Các kỹ thuật nén số liệu

a, Nén không tổn hao (Lossless Compression)

Nén không tổn hao còn gọi là nén chính xác hay nén không mất thông tin. Đây là phương pháp nén mà sau khi giải nén ta thu được một bản sao chính xác của dữ liệu

gốc. Phương pháp nén này thường được áp dụng đối với các nguồn số liệu mà nội dung thông tin cần được bảo toàn như các văn bản dạng text, các bảng tính hay là cơ sở dữ liệu…

Dạng nén mà em dùng trong bài này là dạng nén không tổn hao.

b, Nén tổn hao (Lossy Compression)

Nén tổn hao còn được gọi là nén có mất mát thông tin. Kỹ thuật nén này chấp nhận mất mát một lượng thông tin nhất định để thu được hiệu suất nén cao hơn, do vậy sau khi giải nén ta không thu được dữ liệu gốc.

Nén hao tổn thường được áp dụng cho các tập tin hình ảnh hay âm thanh được số hóa. Bởi vì đối với các tập tin thuộc loại này thì việc mất mát một ít thông tin là điều có thể chấp nhận được.

2.3 Lý thuyết về mã hóa

2.3.1 Định nghĩa mã hóa

Mã hóa là quá trình biến đổi thông tin từ dạng có thể đọc được sang dạng không thể đọc được nếu không có khóa hoặc thuật toán để giải mã. Mục đích của mã hóa là bảo vệ tính bí mật, toàn vẹn và khả dụng của thông tin khi truyền, lưu trữ hoặc xử lý. Mã hóa có thể được thực hiện bằng nhiều cách khác nhau, tùy thuộc vào mức độ an toàn, tốc độ và chi phí mong muốn. Một số loại mã hóa phổ biến hiện nay là:

Mã hóa cổ điển: là loại mã hóa đơn giản nhất, không cần khóa bảo mật, chỉ cần người gửi và người nhận biết thuật toán mã hóa. Ví dụ: mật mã Caesar, mật mã Vigenère, mật mã Playfair, v.v.

Mã hóa đối xứng: là loại mã hóa sử dụng cùng một khóa cho cả quá trình mã hóa và giải mã. Khóa này phải được chia sẻ bí mật giữa người gửi và người nhận. Ví dụ: mật mã DES, mật mã AES, mật mã RC4, v.v.

Mã hóa bất đối xứng: là loại mã hóa sử dụng hai khóa khác nhau cho quá trình mã hóa và giải mã. Một khóa được gọi là khóa công khai, có thể được phổ biến cho mọi người, và một khóa được gọi là khóa bí mật, chỉ được giữ riêng cho chủ sở hữu. Ví dụ: mật mã RSA, mật mã ECC, mật mã ElGamal, v.v.

2.3.2 Một số khái niệm cơ bản

a, Chiều dài từ mã

Chiều dài từ mã hóa (đôi khi được gọi là mã dài) thường được sử dụng trong ngành khoa học máy tính và lĩnh vực mã hóa hóa. Đây là một khái niệm quan trọng trong kích thước xác định của dữ liệu được mã hóa.

Trong bối cảnh này, chiều dài của mã hóa thường được sử dụng để mô tả số lượng bitcas hoặc ký tự trong một mã hóa chuỗi. Chiều dài từ mã là số kí hiệu của bộ mã dùng để mã hóa cho mã đó.

b, Trọng lượng từ mã

Trọng lượng từ mã trong mã hóa Huffman là tổng số bit cần thiết để mã hóa một ký tự cụ thể trong một chuỗi. Nó được xác định bằng chiều dài của mã Huffman được gán cho ký tự đó. Trong mã hóa Huffman, mỗi ký tự được gán một mã nhị phân duy nhất.

Mã này được xây dựng dựa trên tần suất xuất hiện của các ký tự trong chuỗi. Các ký tự xuất hiện thường xuyên hơn sẽ được gán mã ngắn hơn. Các ký tự xuất hiện ít hơn sẽ được gán mã dài hơn. Trọng lượng từ mã của một ký tự bằng số lượng bit cần thiết để mã hóa ký tự đó.

Tổng trọng lượng từ mã của tất cả các ký tự trong chuỗi bằng độ dài trung bình của mã Huffman. Độ dài trung bình của mã Huffman càng nhỏ thì mã hóa càng hiệu quả. Trọng lượng từ mã là tổng số các kí hiệu khác 0 của từ mã đó.

VD: Từ mã 1011010 có trọng lượng là 4.

c, Khoảng cách mã

Trong mã hóa Huffman, khoảng cách mã (code distance) là số lượng bit ít nhất giữa hai mã Huffman khác nhau. Khoảng cách mã quan trọng vì nó ảnh hưởng đến hiệu suất chung của mã Huffman.

Khoảng cách mã lý tưởng của một mã Huffman là 2, vì điều này đảm bảo rằng mỗi ký tự được mã hóa bằng một mã duy nhất và không có hai mã nào chồng lấn lên

nhau. Tuy nhiên, trong thực tế, không phải lúc nào cũng có thể đạt được khoảng cách mã là 2. Khi không thể đạt được khoảng cách mã là 2, thì khoảng cách mã càng lớn thì mã Huffman càng hiệu quả hơn.

Có hai yếu tố chính ảnh hưởng đến khoảng cách mã của một mã Huffman:

Tần suất xuất hiện của các ký tự: Các ký tự xuất hiện thường xuyên hơn sẽ có mã ngắn hơn, trong khi các ký tự xuất hiện ít thường xuyên hơn sẽ có mã dài hơn. Điều này dẫn đến sự chồng chéo giữa các mã của các ký tự khác nhau và giảm khoảng cách mã.

Số lượng ký tự: Số lượng ký tự trong một bảng chữ cái cũng ảnh hưởng đến khoảng cách mã. Với số lượng ký tự càng lớn, thì khoảng cách mã càng khó đạt được.

Để cải thiện khoảng cách mã của một mã Huffman, có thể sử dụng một số kỹ thuật, bao gồm:

Cân bằng lại cây mã: Cây mã Huffman có thể được cân bằng lại để giảm số lượng mã có độ dài khác nhau. Điều này có thể giúp tăng khoảng cách mã.

Ghép các nút lá: Các nút lá có mã giống nhau có thể được ghép lại thành một nút duy nhất. Điều này làm giảm số lượng mã trong cây và tăng khoảng cách mã.

Sử dụng bảng chữ cái thay đổi: Bảng chữ cái có thể được thay đổi để giảm số lượng ký tự trong bảng chữ cái. Điều này làm tăng khoảng cách mã.

2.3.3 Một số phương pháp biểu diễn mã thông dụng

a, Phương pháp liệu kê

Liệt kê trong một bảng những thông tin của nguồn và kèm theo là các từ mã tương ứng. Ưu điểm của phương pháp này là rõ ràng, đơn giản nhưng không phù hợp với những bộ mã lớn.

VD: Nguồn tin X = {a, b, c, d}. Các lớp tin được mã hóa như sau:

Bảng 1 Ví dụ phương pháp liệt kê

Tin	a	b	c	d
Từ mã	10	01	110	001

b, Phương pháp đồ hình kết cấu

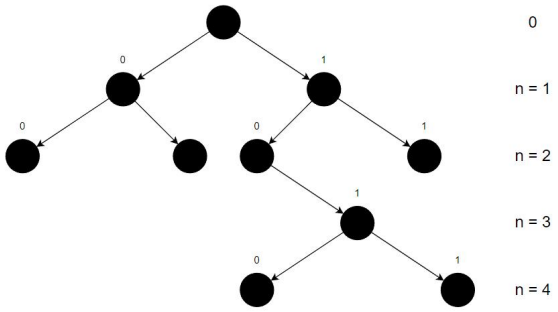
Phương pháp này biểu diễn mà bằng một cây mã rút gọn bao gồm các nút và các nhánh có hướng. Một vòng kín bắt đầu tại nút gốc , đi theo các nhánh theo chiều mũi tên, qua các nút trung gian và kết thúc tại nút gốc sẽ biểu diễn cho 1 từ mã. Thứ tự các nhánh trên dường đi chính là thư tự giá trị các kí hiệu.

c, Phương pháp cây

Cây mã được biểu diễn bao gồm gốc và các nhánh. Trong cây có chứa các nút. Nút gốc chính là gốc của cây (mức 0). Nút là nằm tận cùng của nhánh. Trừ nút gốc và các nút lá ra, các nút còn lại là các nút nhánh. Từ một nút nhánh có thể phát đi nhiều nhất m nhánh (ứng với cơ số m của mã). Mỗi nhánh biểu diễn cho 1 từ mã. Từ mã đó thứ tự các trị kí hiệu đi từ gốc, qua từ nút nhánh và dừng lại ở nút lá tương ứng của nhánh.

Dựa vào cây mã, chúng ta có thể nhận biết mã đã cho là mã đều (các nút lá có cùng bậc), hay không đều, mã đầy hay vơi. Mã là đầy khi mọi nút nhánh bậc trước các nút là đều có m nhánh.

VD: Cho bộ mã 00, 01, 11, 1010, 1011. Cây mã biển diễn của bộ mã này là:



Hình 1 Hình minh họa cây nhị phân

2.3.4 Điều kiện để mã phân tách được

Mã được gọi là có tính phân tách nếu như khi nhận được một chuỗi kí hiệu trong quá trình tạo mã, chúng ta có thể tách ra được các thành phần cơ bản là các từ mã và cách tách đó là đúng đắn và duy nhất (vì nếu không, bộ giải mã có thể sẽ nhầm lẫn trong quá trình làm việc). Để có tính phân tách được, bộ mã phải thỏa mãn điều kiện cần và đủ: Bất kỳ dãy các mã nào của bộ mã cũng không được trùng với một dãy từ mã khác của cùng bộ mã.

Các bước xây dựng bảng thử mã phân tách:

1.Sắp xếp các từ mã thành 1 cột , cột này được đánh số 1.

2. Đối sánh các từ mã ngắn với các từ mã dìa hơn trong cột 1, nếu từ mã ngắn trùng với phần đầu của từ mã dài hơn thì lấy phần còn lại của từ mã dài ghi vào cột thứ 2.

3. Lặp lại bước 2 , với cột k là cột chứa kết quả đối sánh giữa cột 1-k với cột 2-k. Tiếp tục thực hiện bước 3 cho đến khi cột k trống rỗng.

2.3.5 Mã có tính tiền số (prefix)

Phần tiền tố của một mã có độ dài 1 là một bộ phận của từ mã đó sau khi bỏ đi k kí hiệu cuối cùng ($0 < k < 1$). Một bộ mã được gọi là có tính chất tiền tố nếu mọi từ mã thuộc bộ mã đều không phải là phần đầu của 1 từ mã khác trong cùng bộ mã.

Nhờ vào tính chất tiền tố này mà mã có tính prefix thường được sử dụng để làm mã nén dữ liệu. Ta có thể nhận thấy rằng, khi biển diễn mã bằng cây mã, mã có tính chất tiền tố khi các từ mã chỉ là nút lá.

VD: Từ mã 1001101 có các tiền tố là: 100110, 10011, 1001, 100, 10 và 1.

2.3.6 Định lí về độ dài trung bình từ mã

Định lí này cho biết giới hạn dưới của độ dài trung bình từ mã khi mã hóa một nguồn tin bất kỳ. Nếu một bộ mã nào đó có độ dài trung bình từ mã bằng với entropi của nguồn tin, thì bộ mã đó được gọi là mã tối ưu. Một ví dụ về mã tối ưu là mã Huffman, một phương pháp mã hóa thường được sử dụng trong nén dữ liệu. Bên cạnh đó có thể tạo được bộ mã có độ dài trug bình của từ mã không lớn hơn tỷ số Entropy của nguồn được mã hóa trên lượng tin trung bình cực đại chứa trong 1 kí hiệu mã cộng thêm 1 đơn vị.

2.4 Mã thống kê tối ưu

Như đã nói, tiêu chuẩn của mã thông kê tối ưu là chiều dài trung bình từ mã tối thiểu. Do xác suất xuất hiện của các từ trong nguồn tin là khác nhau nên việc dùng các từ mã ngắn để mã hóa cho các từ có xác suất xuất hiện cao và ngược lại, dùng các từ mã dài để mã hóa cho các từ có xác suất xuất hiện thấp sẽ làm cho số kí hiệu cần thiết để mã hóa nguồn tin giảm đi. Nguyên tắc cơ bản của mã thông kê tối ưu là dựa trên cơ

sở độ dài từ mã n (tỉ lệ nghịch với xác suất xuất hiện p), tức là các tin có xác suất xuất hiện thấp sẽ được biểu diễn bằng các từ mã dài và ngược lại.

2.4.1 Mã Shannon-Fano

Phương pháp mã hóa đầu tiên được nhiều người biết đến vào cuối những năm 1940 là phương pháp mã hóa Shannon-Fano. Phương pháp này được hai nhà nghiên cứu Claude Shannon và Robert Fano đưa ra gần như đồng thời. Phương pháp mã hóa này chưa dựa trên tần suất hiện của mỗi ký tự trong nguồn số liệu vậy mà được mô tả chi tiết trong sách của Claude Shannon và Warren Weaver "The Mathematical Theory of Communication". Từ đây một số thuật toán khác đã xuất hiện có tính chất tương tự. Kỹ thuật mã hóa này dựa trên tần suất xuất hiện của mỗi ký tự trong nguồn số liệu cần được mã hóa. Từ bảng chứa các tần suất đó, bảng mã sẽ được xây dựng vào các tính chất quan trọng sau:

• Các mã khác nhau có các bit biểu diễn khác nhau.

• Kí tự cơ bản xuất hiện càng cao thì mã càng ngắn (ít bit) và ngược lại.

• Các mã có dạng dãy bit khác nhau.

Mã sẽ được xây dựng dựa trên cấu trúc cây nhị phân, dựa vào Thuật toán xây dựng mã Shannon-Fano:

Vào: Bảng tần số xuất hiện của từng các kí tự có mặt trong nguồn số liệu (Bảng đã được sắp xếp theo thứ tự tăng dần hoặc giảm dần của tần số).

Ra: Cây nhị phân biển diễn mã.

Bước 1. Tách bảng thành hai bảng con sao cho hiệu giữa tổng các tần số trong mỗi bảng con là nhỏ nhất.

Bước 2. Bảng con phía trên được gán giá trị nhị phân 0, bảng con phía dưới được gán trị nhị phân 1.

Bước 3. Tiếp tục thực hiện tuần tự hai bước 1 và 2 cho mỗi bảng con được tách ra cho đến khi các bảng thành phần không thể phân chia được nữa

Nguyên tắc chính là sử dụng phương pháp đệ qui để xây dựng cây mã.

2.4.2 Mã số học

Như đã nói, độ dài từ mã của mã Shannon-Fano phải là một số nguyên các bit. Người ta đã cải tiến nhược điểm này bằng cách đưa ra một loại mã khác, đó là mã số học. Phương pháp mã hóa số học hoàn hảo hơn các phương pháp mã hóa khác ở chỗ

nó không tạo ra một từ mã dơn lẻ cho mỗi kí hiệu mà nó chỉ tạo ra một từ mã duy nhất cho toàn bộ nguồn số liệu. Nghĩa là một kí hiệu có thể được mã hóa bằng 3.5 bit.

Nguyên tắc chính của phương pháp mã hóa này là mã hóa toàn bộ lượng số liệu thành một số. Mỗi kí tự / xâu kí tự của luồng nhập sẽ được biến đổi thành một số thực có giá trị thuộc nửa khoảng [0;1). Việc biến đổi này tuân theo ánh xạ 1-1.

Đối với phương pháp này, trước hết, chúng ta cần lập bảng thống kê tần số xuất hiện của các kí tự. Sau đó gán cho mỗi kí tự một khoảng biên thiên và gọi là hàng của kí tự đó.

2.4.3 Mã Huffma

2.5 Mô hình hóa nguồn số liệu

Như ta đã biết, Entropy của nguồn số liệu phụ thuộc vào xác suất, trong khi đó, xác suất lại phụ thuộc vào mô hình. Do đó, xác suất sẽ thay đổi nếu như chúng ta thay đổi mô hình và Entropy cũng biến đổi theo. Như vậy, có thể thấy rằng hiệu quả nén phụ thuộc rất nhiều vào mô hình.

Nhìn chung, quá trình nén không thể hoàn hảo được thực hiện dựa vào một trong hai kiểu mô hình khác nhau: mô hình thông kê (Statistical) và mô hình từ điển (Dictionary-based). Nén theo mô hình thông kê sẽ mã hoá mỗi lúc một kí hiệu dựa vào tần suất xuất hiện của nó. Nén theo mô hình từ điển sẽ mã hoá mỗi lúc một chuỗi kí hiệu chỉ bằng một từ mã. Như vậy, vai trò của mô hình là vô cùng quan trọng. Một mó hinh tốt sẽ cho hiệu quả nén cao và ngược lại.

2.5.1 Mô hình thống kê

Dạng đơn giản nhất của mô hình này, đúng như tên gọi của nó, là thống kê các khối số liệu điển hình nào đó để có được một bảng tĩnh liệt kê các giá trị tần suất. Dựa vào bảng này, một cây mã tĩnh được xây dựng sẵn và lưu trữ để có thể sử dụng nhiều lần. Một mô hình như thế được gọi là mô hình thống kê tĩnh (Static statistical model).

2.5.2 Mô hình từ điển

Đặc điểm chung của các mô hình thông kê là mã hóa (và giải mã) một lúc một kí hiệu. Còn các mô hình từ điển thì tạo mã cho một cụm hoặc loạt kí hiệu. Nguyên

tắc của chúng là tạo một ảnh xạ từ một chuỗi kí hiệu thành mã sao cho kích thước của mã nhỏ hơn kích thước của chuỗi kí hiệu đó. Khi mã hóa, dữ liệu được đọc vào và thuật toán tìm xem có nhóm kí hiệu tương tự nào xuất hiện trong từ điển hay không. Nếu có, nó sẽ xuất ra một mã ảnh xạ đến nhóm kí hiệu đó. Dữ liệu vào càng trùng hợp với các nhóm kí hiệu trong từ điển hoặc kich thước nhóm ki hiệu được ảnh xạ càng lớn thì hiểu quả nén càng cao. Ở đây, vai trò của mô hình hóa là cực kỳ quan trọng, công việc mã hoá chỉ đóng vai trò thứ yếu.

Thực tế cho thấy, so với các kỹ thuật nén sử dụng mô hình thống kê, các kỹ thuật nén áp dụng mô hình từ điển cho một hiệu quả cao hơn nhiều, cả về tỉ số nén, tốc độ né và giải nén. Đó là lý do chúng được sử dụng phổ biến hiện nay.

CHƯƠNG 3: PHƯƠNG PHÁP MÃ HÓA HUFFMAN

Bước 1. Đọc dữ liệu đầu vào: Sử dụng C++ để đọc dữ liệu từ một tệp tin ngoài. (lấy dữ liệu từ ngoài)

```
string inputFile = "CNDPT_Nhom6.txt";

   // Bước 1: Đọc dữ liệu từ tệp tin đầu vào
   ifstream inFile(inputFile, ios::binary | ios::ate);
   streampos fileSize = inFile.tellg();
      inFile.seekg(0, ios::beg);
   string data((unsigned int)fileSize, ' ');
   inFile.read(&data[0], fileSize);
   inFile.close();
```

Bước 2. Xây dựng bảng thống kê tần xuất suất hiện của các ký tự: Sử dụng một map để lưu tần suất xuất hiện của từng ký tự trong dữ liệu.

```
// Bước 2: Lập bảng thống kê
   map<char, int> freqTable;
   for (char c : data) {
      freqTable[c]++;
   }
```

Bước 3. Xây dựng cây huffman từ bang thống kê: Sử dụng một hàng đợi ưu tiện (priority_queue) để xây dựng cây huffman từ các nút lá có tần suất thấp nhất.

```
// Định nghĩa một nút trong cây Huffman
struct Node {
   char data;
   int freq;
   Node* left, *right;

   Node(char data, int freq) : data(data), freq(freq), left(nullptr),
right(nullptr) {}
};
// So sánh để sử dụng trong priority_queue
struct Compare {
```

```
   bool operator()(Node* left, Node* right) {
      return left->freq > right->freq;
   }
};
```

```
// Tạo cây Huffman từ bảng thống kê
Node* buildHuffmanTree(map<char, int>& freqTable) {
   priority_queue<Node*, vector<Node*>, Compare> pq;
   for (auto& pair : freqTable) {
      pq.push(new Node(pair.first, pair.second));
   }
   while (pq.size() > 1) {
      Node* left = pq.top(); pq.pop();
      Node* right = pq.top(); pq.pop();
      Node* merged = new Node('$', left->freq + right->freq);
      merged->left = left;
      merged->right = right;
      pq.push(merged);
   }
   return pq.top();
}
```

Bước 4. Xây dựng bảng mã huffman

Sử dụng đệ quy để duyệt cây huffman và xây dựng bảng mã cho từng ký tự.

```
// Tạo bảng mã Huffman
void buildHuffmanCodes(Node* root, string code, map<char, string>&
huffmanCodes) {
   if (root == nullptr) return;

   if (root->data != '$') {
      huffmanCodes[root->data] = code;
   }
   buildHuffmanCodes(root->left, code + "0", huffmanCodes);
   buildHuffmanCodes(root->right, code + "1", huffmanCodes);
}
```

Bước 5. Mã hóa dữ liệu và xuất kết quả:

Sử dụng bảng mã huffman để mã hóa dữ liệu và xuất kết quả ra màn hình.

```
// Mã hóa dữ liệu và trả về kết quả
string encodeData(string data, map<char, string>& huffmanCodes) {
   string encodedData = "";
   for (char c : data) {
      encodedData += huffmanCodes[c];
   }

   return encodedData;
}
```

Bước 6. Tính toán kích thước dữ liệu:

Tính kích thước dữ liệu ban đầu và kích thước dữ liệu đã mã hóa để tính tỉ lệ nén.

```
// Bước 6: Tính toán input bit và output byte
int inputBits = fileSize * 8;
int outputBytes = (encodedData.length()); // làm tròn lên khi cần thiết

// Bước 7: Tính tỷ lệ nén
double compressionRatio = ((double)(inputBits-outputBytes) /
inputBits*100;
// In thông tin
cout << "Input byte: " << inputBits << " bytes" << endl;
cout << "Ouput byte: " << outputBytes << " bytes" << endl;
cout << "Ti le nen: " << compressionRatio << endl;
```

CHƯƠNG 4: CÀI ĐẶT

Module 1:

```
#include <iostream>
#include <fstream>
#include <queue>
#include <map>
#include <bitset>

using namespace std;
// Định nghĩa một nút trong cây Huffman
   struct Node {
   char data;
   int freq;
   Node* left, *right;
   Node(char data, int freq) : data(data), freq(freq), left(nullptr),
right(nullptr) {}
};

// So sánh để sử dụng trong priority_queue
struct Compare {
   bool operator()(Node* left, Node* right) {
      return left->freq > right->freq;
   }
};

// Tạo cây Huffman từ bảng thống kê
Node* buildHuffmanTree(map<char, int>& freqTable) {
   priority_queue<Node*, vector<Node*>, Compare> pq;

   for (auto& pair : freqTable) {
      pq.push(new Node(pair.first, pair.second));
   }
  while (pq.size() > 1) {
      Node* left = pq.top(); pq.pop();
      Node* right = pq.top(); pq.pop();
      Node* merged = new Node('$', left->freq + right->freq);
      merged->left = left;
      merged->right = right;
      pq.push(merged);
   }
```

```
   return pq.top();
}
```

Module 2:

```
// Tạo bảng mã Huffman
void buildHuffmanCodes(Node* root, string code, map<char, string>&
huffmanCodes) {
   if (root == nullptr) return;

   if (root->data != '$') {
      huffmanCodes[root->data] = code;
   }

   buildHuffmanCodes(root->left, code + "0", huffmanCodes);
   buildHuffmanCodes(root->right, code + "1", huffmanCodes);
}

// Mã hóa dữ liệu và trả về kết quả
string encodeData(string data, map<char, string>& huffmanCodes) {
   string encodedData = "";
   for (char c : data) {
      encodedData += huffmanCodes[c];
   }

   return encodedData;
}
```

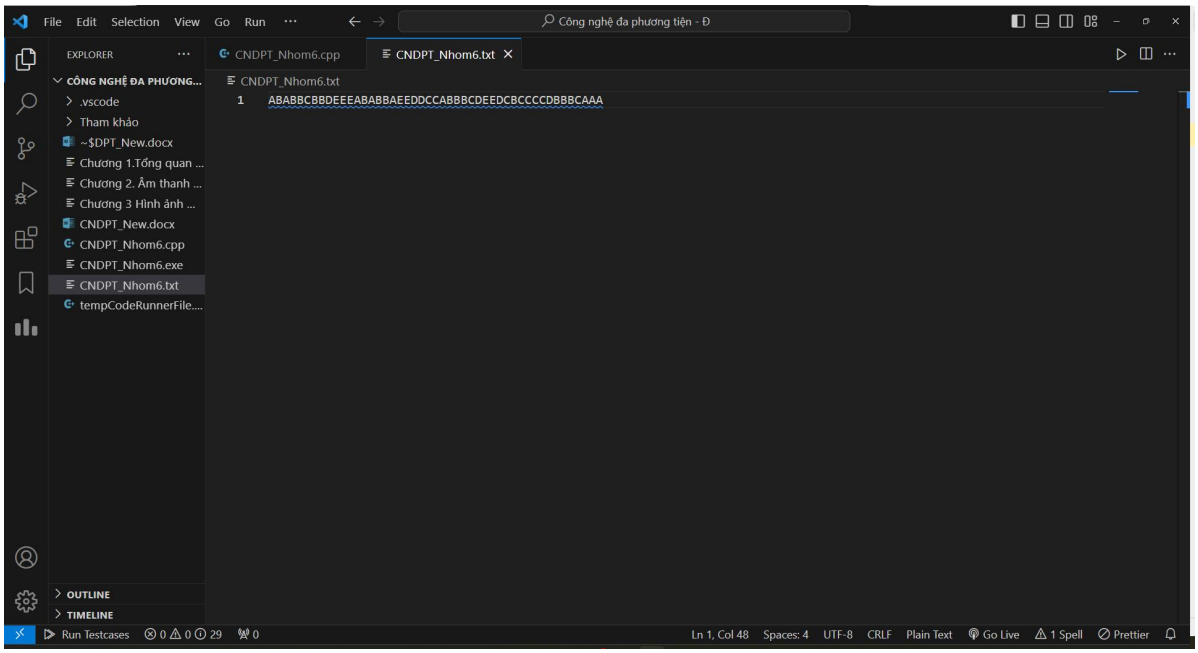
Module 3:

```
int main() {
   string inputFile = "CNDPT_Nhom6.txt";

   // Bước 1: Đọc dữ liệu từ tệp tin đầu vào
   ifstream inFile(inputFile, ios::binary | ios::ate);
   streampos fileSize = inFile.tellg();
   inFile.seekg(0, ios::beg);

   string data((unsigned int)fileSize, ' ');
   inFile.read(&data[0], fileSize);
   inFile.close();
```

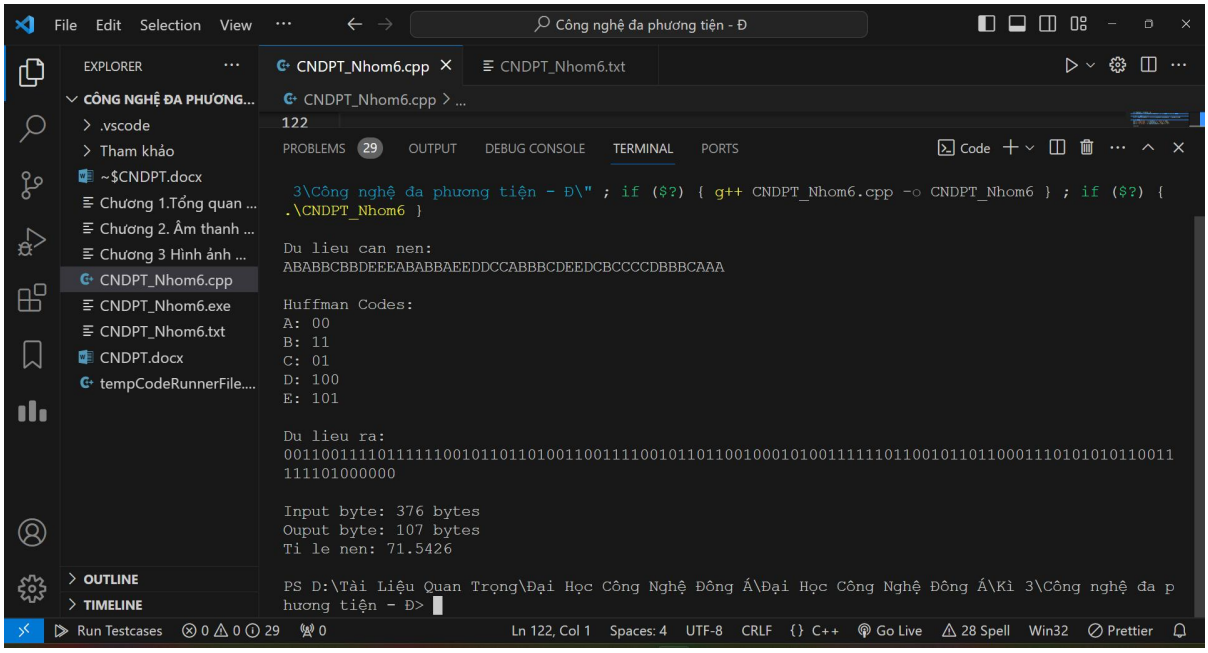
```
   // Bước 2: Lập bảng thống kê
   map<char, int> freqTable;
   for (char c : data) {
      freqTable[c]++;
   }

   // Bước 3: Tạo cây Huffman
   Node* root = buildHuffmanTree(freqTable);

   // Bước 4: Tạo bảng mã Huffman
   map<char, string> huffmanCodes;
   buildHuffmanCodes(root, "", huffmanCodes);

   // Bước 5: Mã hóa dữ liệu và in kết quả ra terminal
   cout << "\nDu lieu can nen:\n" << data << "\n\n";

   cout << "Huffman Codes:\n";
   for (auto& pair : huffmanCodes) {
      cout << pair.first << ": " << pair.second << endl;
   }
   cout << "\n";

   string encodedData = encodeData(data, huffmanCodes);

   // In dãy bit mã hóa Huffman ra terminal
   cout << "Du lieu ra:\n" << encodedData << "\n\n";

   // Bước 6: Tính toán input bit và output byte
   int inputBits = fileSize * 8;
   int outputBytes = (encodedData.length()); // làm tròn lên khi cần thiết

   // Bước 7: Tính tỷ lệ nén
   double compressionRatio = ((double)(inputBits-
outputBytes)/inputBits*100;

   // In thông tin
   cout << "Input byte: " << inputBits << " bytes" << endl;
   cout << "Ouput byte: " << outputBytes << " bytes" << endl;
   cout << "Ti le nen: " << compressionRatio << endl << "\n";

   return 0;
}
```

CHƯƠNG 5: THỰC NGHIỆM

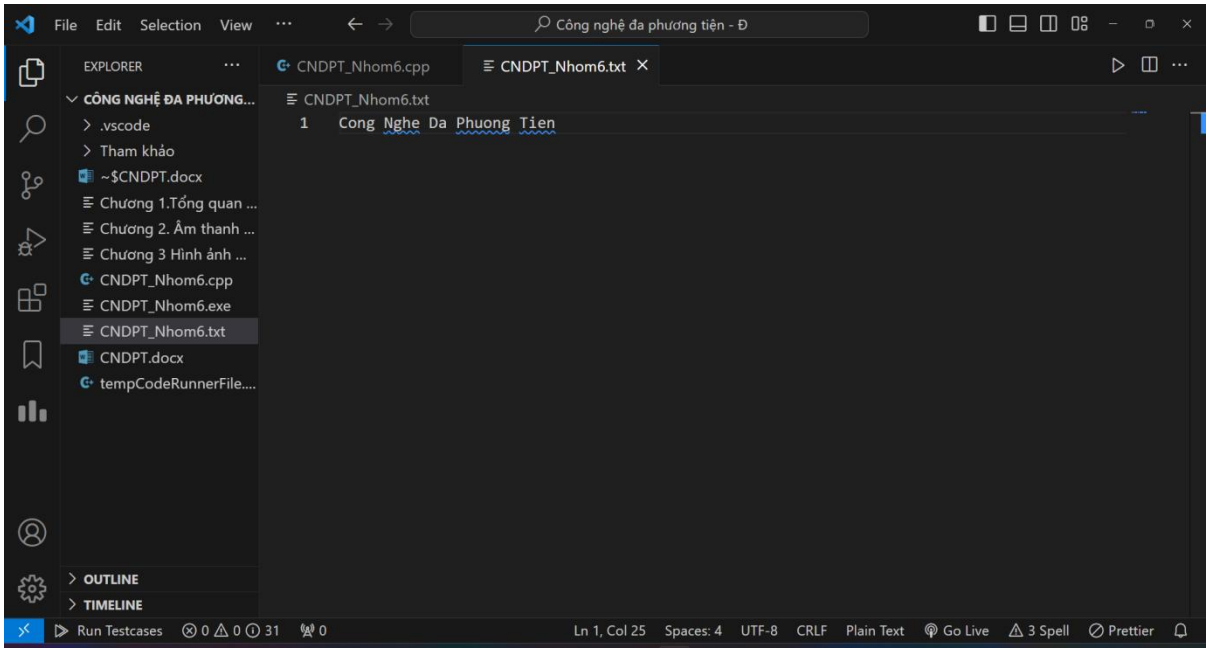
Đây là hình ảnh chạy thực nghiệm kỹ thuật mã hóa huffman:



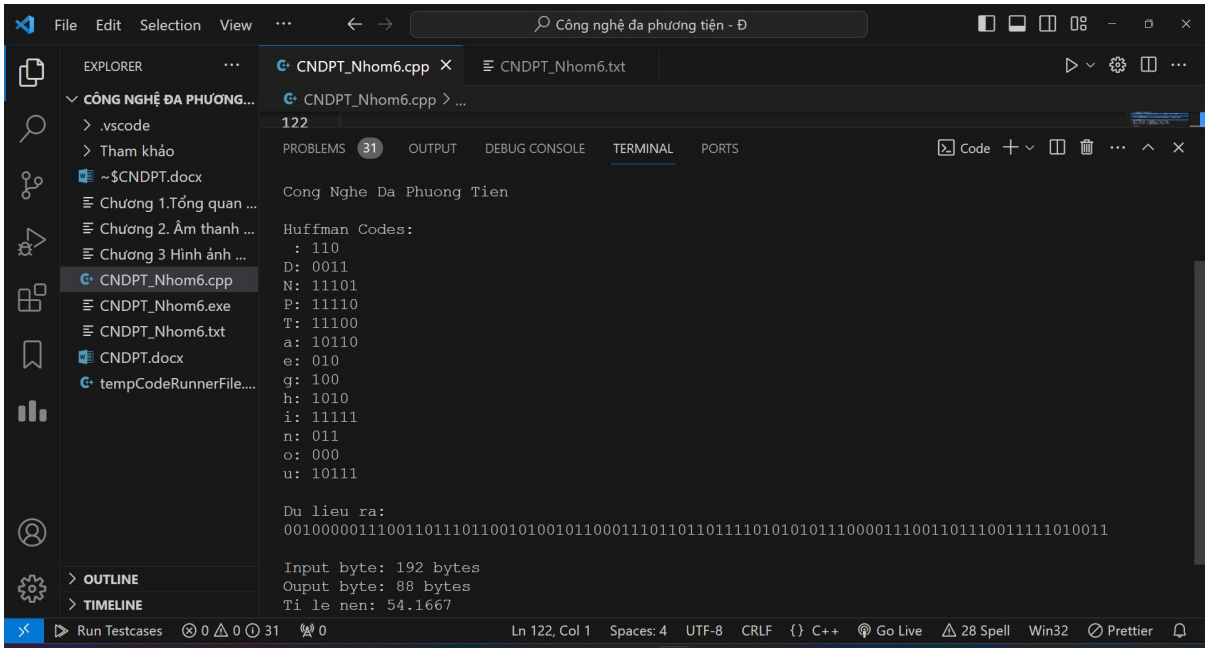
Hình 2 Dãy kí tự cần mã hóa thứ 1



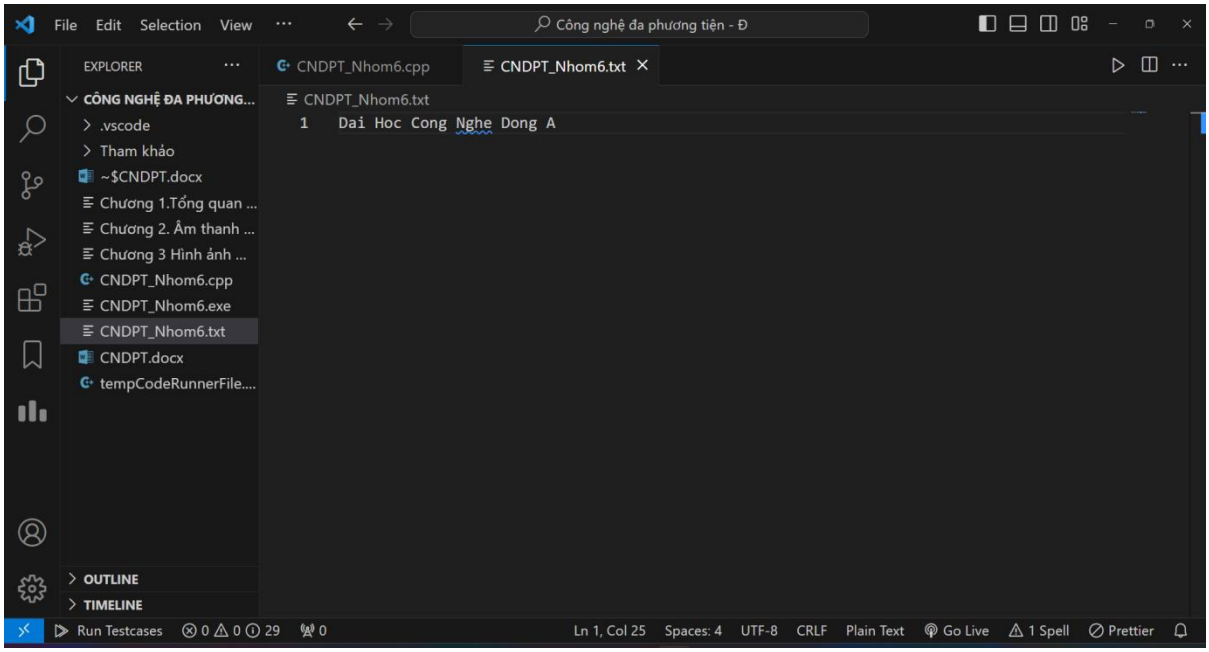
Hình 3 Kết quả dãy kĩ tự được mã hóa thứ 1



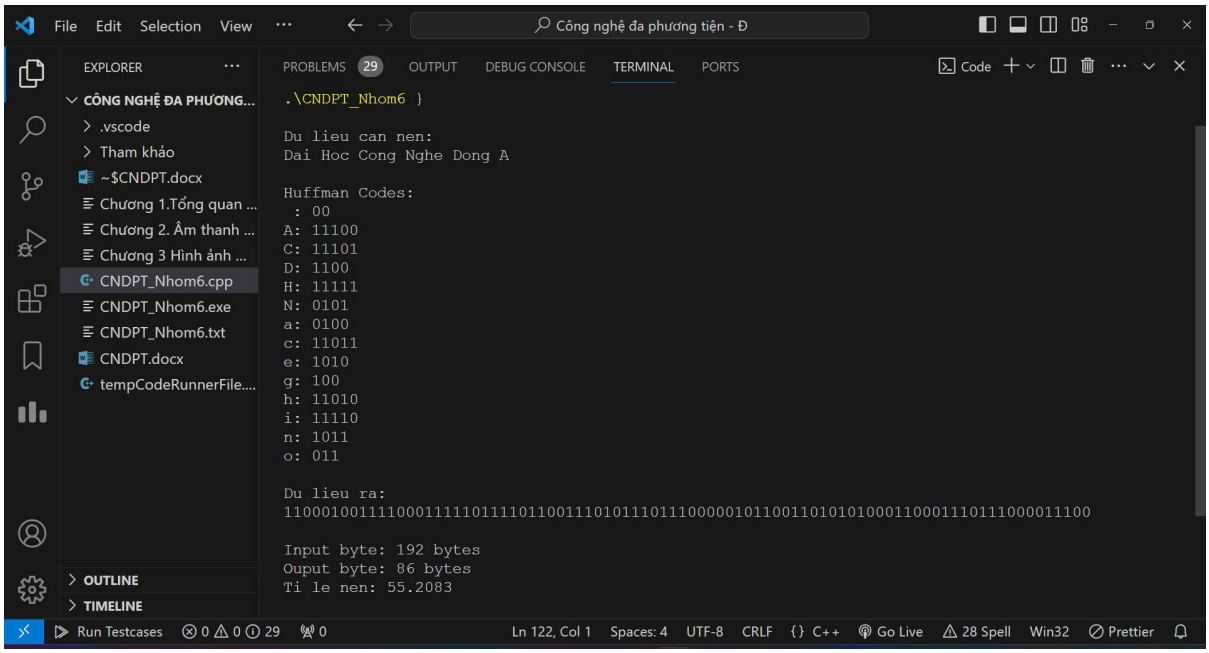
Hình 4 Dãy kí tự cần mã hóa thứ 2



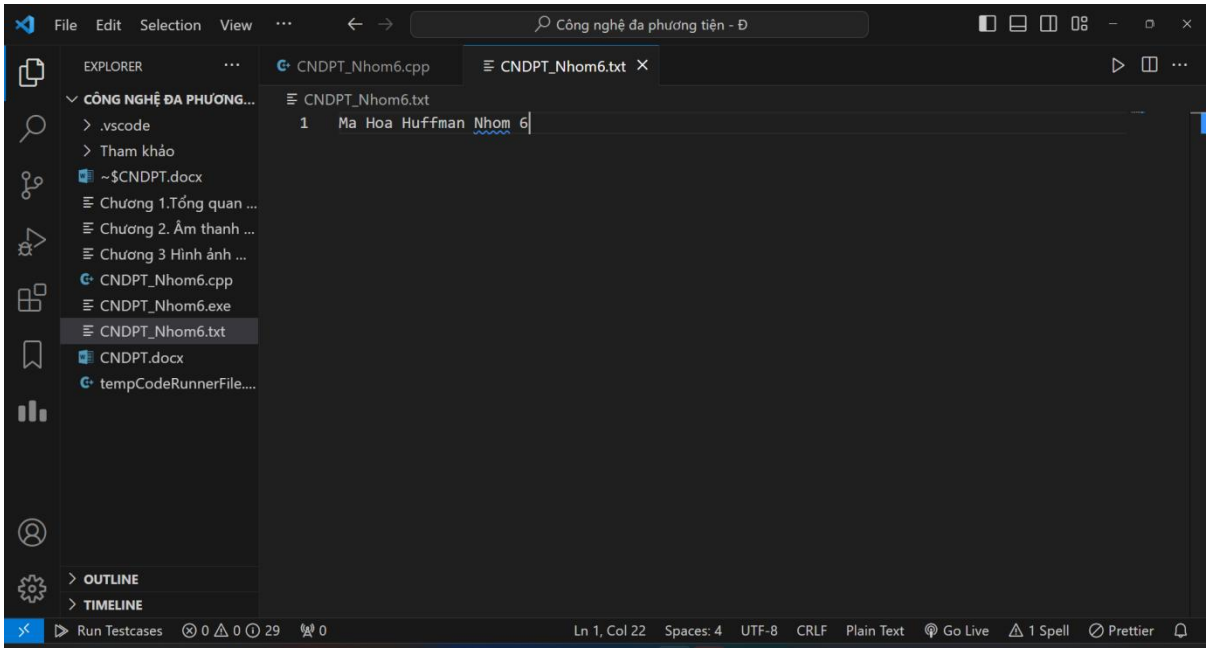
Hình 5 Kết quả dãy kĩ tự được mã hóa thứ 2



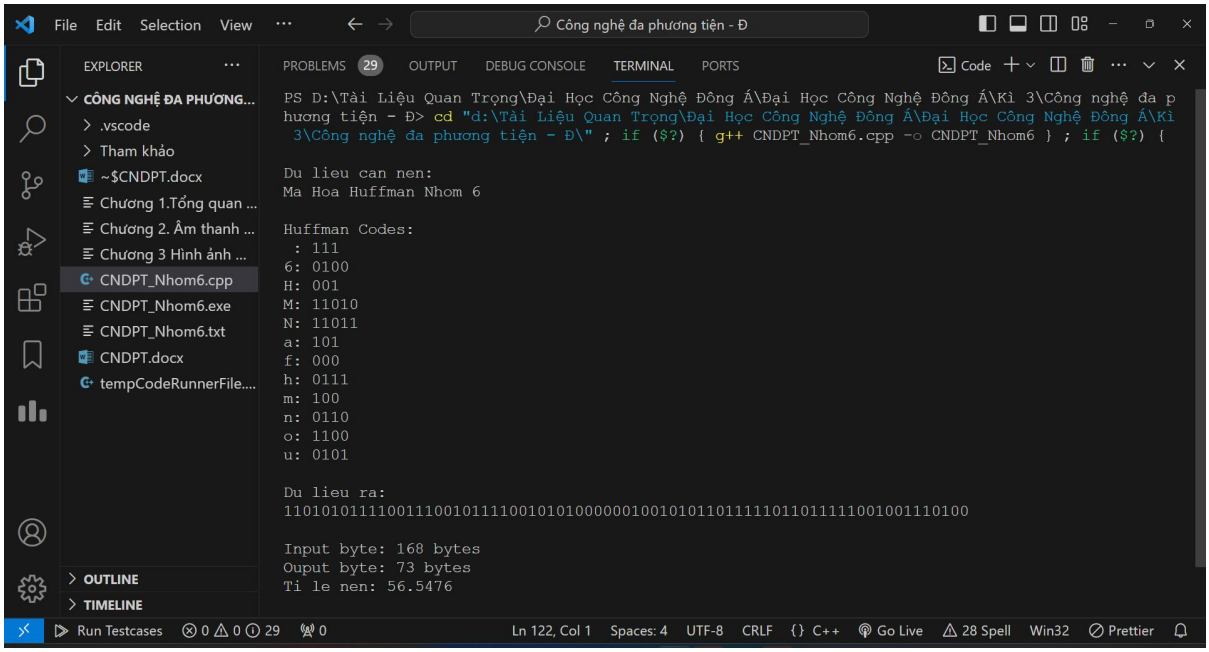
Hình 6 Dãy kí tự cần mã hóa thứ 3



Hình 7 Kết quả dãy kĩ tự được mã hóa thứ 3



Hình 8 Dãy kí tự cần mã hóa thứ 4



Hình 9 Kết quả dãy kĩ tự được mã hóa thứ 4

KẾT LUẬN

Kết quả đạt được.

Mã hóa Huffman là một phương pháp nén dữ liệu hiệu quả, nơi mỗi ký tự trong chuỗi được biểu diễn bằng một mã nhị phân có độ dài khác nhau. Kết quả của quá trình mã hóa Huffman thường là một bảng mã, trong đó các ký tự thường xuyên xuất hiện được biểu diễn bằng các mã ngắn hơn so với các ký tự ít xuất hiện.

Khi áp dụng mã hóa Huffman, ta thu được dữ liệu đã được nén, giảm kích thước so với dữ liệu gốc. Điều này giúp tiết kiệm không gian lưu trữ và tăng tốc quá trình truyền tải dữ liệu qua mạng. Đồng thời, mã hóa Huffman cũng giúp tối ưu hóa việc lưu trữ dữ liệu trên thiết bị và ổ đĩa, đặc biệt là trong các ứng dụng yêu cầu sự hiệu quả về nguồn lực.

Tuy nhiên, để giải mã dữ liệu sau khi đã mã hóa Huffman, cần sử dụng bảng mã tương ứng để phục hồi dữ liệu gốc. Do đó, việc quản lý và truyền tải bảng mã này cũng là một phần quan trọng trong quá trình sử dụng mã hóa Huffman.

Tóm lại, kết quả của mã hóa Huffman là sự giảm kích thước dữ liệu mà vẫn giữ được thông tin quan trọng, tạo ra một biểu diễn nén hiệu quả và thích hợp cho nhiều ứng dụng lưu trữ và truyền tải dữ liệu. Và còn một vài khó khăn thực tế mà nhóm còn gặp phải:

Chưa có kinh nghiệm đưa ra gói tư vấn, chưa đo lường hết tính khả thi của giải pháp.

Các nguồn tài liệu tham khảo chưa nhất quán, không thống nhất trong việc thiết kế giữa các thành viên.

Việc xác định các công việc trong từng yêu cầu đề bài có độ phức tạp lớn hơn so với dự kiến nên nhiều lúc quá trình làm việc bị đình trệ. Trong đề tài này tuy thời gian từ khi nhận đề tài cho đến khi hoàn thành có nhiều nhưng em vẫn chưa thể hoàn thành đề tài được như ý muốn, vẫn không thể tránh khỏi được những sai sót và những lỗi trong quá trình làm dù đã có sự tìm hiểu.

Hướng phát triển.

Mã hóa Huffman đã là một kỹ thuật nén dữ liệu mạnh mẽ và phổ biến trong nhiều ứng dụng. Tuy nhiên, như mọi lĩnh vực công nghệ khác, có những hướng phát triển và cải tiến liên tục để tối ưu hóa hiệu suất và ứng dụng rộng rãi hơn. Dưới đây là một số hướng phát triển có thể xuất hiện trong lĩnh vực mã hóa Huffman:

1. Tối ưu hóa hiệu suất: Nghiên cứu tiếp tục tập trung vào cách tối ưu hóa quá trình mã hóa và giải mã Huffman để cải thiện tốc độ và hiệu suất. Điều này có thể bao gồm thuật toán mới và cải tiến trong việc tạo ra bảng mã Huffman.

2. Mã hóa đa cấp: Các kỹ thuật mã hóa đa cấp có thể được phát triển để áp dụng một loạt các thuật toán nén dữ liệu theo cấp độ khác nhau. Điều này có thể giúp tối ưu hóa quá trình nén cho các loại dữ liệu cụ thể.

3. Mã hóa dựa trên ngữ cảnh: Phát triển các kỹ thuật mã hóa Huffman có khả năng hiểu ngữ cảnh của dữ liệu. Điều này có thể giúp cải thiện hiệu suất nén cho các chuỗi dữ liệu có tính chất đặc biệt hoặc có thứ tự.

4. Tích hợp với các phương pháp khác: Kết hợp mã hóa Huffman với các phương pháp nén khác để tạo ra các kỹ thuật nén mạnh mẽ hơn. Sự kết hợp này có thể bao gồm kỹ thuật nén không mất mát hoặc sử dụng mã hóa chuỗi thời gian.

5. Mã hóa động: Phát triển các phương pháp mã hóa Huffman có khả năng thích ứng động, tức là có thể thay đổi bảng mã trong quá trình mã hóa dữ liệu. Điều này có thể giúp tối ưu hóa việc mã hóa cho các khối dữ liệu có tính chất thay đổi theo thời gian.

6. Tích hợp với trí tuệ nhân tạo: Sử dụng các thuật toán trí tuệ nhân tạo để xác định và dự đoán các mô hình xuất hiện trong dữ liệu, từ đó tối ưu hóa quá trình mã hóa Huffman.

Những hướng phát triển này có thể giúp mã hóa Huffman tiếp tục đóng góp vào lĩnh vực nén dữ liệu, đặc biệt là trong bối cảnh ngày càng tăng về kích thước và lưu trữ dữ liệu.

DANH MỤC THAM KHẢO

1. ChatGPT (openai.com) [12/12/2023 - 11:00PM]

2. Đồ án Kỹ thuật mã hóa Huffman với mô hình từ điển - Luận văn, đồ án, đề tài tốt nghiệp (luanvan.net.vn) [11/12/2023 - 11:00PM]

3. Bài giảng Cấu trúc dữ liệu và giải thuật - Bài 5: Nén Huffman - HCMUS - Luận văn, đồ án, luan van, do an [23/12/2023 - 11:00PM]

4. Mã hóa Huffman – Wikipedia tiếng Việt [19/12/2023 - 11:00PM]

5. Thuật toán nén Huffman Coding (chidokun.github.io) [10/12/2023 - 11:00PM]

6. Cài đặt thuật toán Huffman Coding (chidokun.github.io) [19/12/2023 - 11:00PM]

7. Nén không tổn hao – Wikipedia tiếng Việt [12/12/2023 - 11:00PM]

8. Nén có tổn hao – Wikipedia tiếng Việt [19/12/2023 - 11:00PM]

9. Lý thuyết mã hóa – Wikipedia tiếng Việt [12/12/2023 - 11:00PM]

10. Mã hóa – Mã thống kê tối ưu Khái niệm mã hóa, các thông số của mã - TaiLieu.VN [23/12/2023 - 11:00PM]

11. Mô hình hóa dữ liệu – Wikipedia tiếng Việt [19/12/2023 - 11:00PM]

12. Nén dữ liệu – Wikipedia tiếng Việt [12/12/2023 - 11:00PM]

13. Đồ án Kỹ thuật mã hóa Huffman với mô hình từ điển - Tài liệu, ebook, giáo trình (doc.edu.vn) [12/12/2023 - 11:00PM]