ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

CHALLENGE 01

TÌM HIỂU THUẬT TOÁN GIẢI NÉN HUFFMAN VÀ LZW

Giáo viên hướng dẫn:

Thầy Văn Chí Nam
Thầy Lê Thanh Tùng
Cô Phan Thị Phương Uyên

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT - 21CTT3

Hồ Chí Minh, ngày 18 tháng 11 năm 2022

THÔNG TIN ĐỒ ÁN

THÔNG TIN NHÓM - LỚP 21CTT3

MSSV	Họ và tên
21120275	Huỳnh Cao Khôi
21120307	Trần Đình Nhật
21120308	Phạm Lê Tú Nhi

THỜI HẠN DỰ ÁN: 16/11/2022 - 20/11/2022

LỜI CẢM ƠN:

Chúng em xin gửi lời cảm ơn tới thầy Văn Chí Nam vì đã hướng đãn tận tình và đưa lại kiến thức cần thiết cũng như làm nguồn cảm hứng để học hỏi thêm về cấu trúc dữ liệu và giải thuật nói riêng và về ngành công nghệ thông tin nói chung. Chúng em cũng xin gửi lời cảm ơn cô Phan Thị Phương Uyên vì những hướng dẫn kĩ lưỡng và những lời khuyên của cô trong quá trình học và thầy Lê Thanh Tùng vì đã giành thời gian soạn thảo tài liệu và đề bài kĩ lưỡng cùng với cơ hội được tìm hiểu chuyên sâu về cách chủ đề khác nội dung học tập - như là đồ án này.

Mục lục

1 Thuật toán Huffman tĩnh

1.1 Pha nén

1.1.1 Ý tưởng

Ý tưởng của thuật toán Huffman Encoding dựa trên việc gán mã cho mỗi ký tự đầu vào, độ dài của mã được gán phụ thuộc vào tần suất xuất hiện của các ký tự tương ứng, những phần tử xuất hiện nhiều nhất sẽ được gán mã có độ dài ngắn nhất, các phần tử xuất hiện ít hơn sẽ có mã dài hơn.

1.1.2 Trình bày thuật toán

Thuật toán Huffman Encoding gồm các bước thực hiện sau:

- Bước 1: Đếm tần suất xuất hiện của các phần tử trong chuỗi đầu vào.
- Bước 2: Xây dựng cây nhị phân Huffman (cây nhị phân mã hóa).
- Bước 3: Từ cây Huffman, ta có được các giá trị mã hóa, từ đó ta xây dựng chuỗi mã hóa dựa trên các giá trị này.

Các mã dùng để gán cho các ký tự đầu vào được gọi là Prefix code (Mã tiền tố). Hiểu một cách đơn giản, mã tiền tố là một tập các giá trị mà không phần tử nào bắt đầu bằng một phần tử khác.

Dựa trên mã tiền tố, ta sẽ xây dựng cây nhị phân Hufman theo nguyên tắc sau: Các ký tự cần mã hóa sẽ nằm ở nút lá, và giá trị mã hóa sẽ được thể hiện trên đường đi từ gốc đến nút đó, với nhánh bên trái thể hiện giá trị 0, nhánh bên phải thể hiện giá trị 1.

Quá trình xây dựng được thể hiện như sau:

- Bước 1: Tạo danh sách chứa các nút lá bao gồm phần tử đầu vào và trọng số nút là tần suất xuất hiện của nó.
- Bước 2: Từ danh sách này, lấy ra 2 phần tử có tần suất xuất hiện ít nhất.

- Bước 3: Gắn 2 nút vừa lấy ra vào một nút gốc mới với trọng số là tổng của 2 trọng số ở nút vừa lấy ra để tạo thành một cây và thêm lại cây vừa tạo vào danh sách.

- Bước 4: Lặp lại các bước 2 và 3 cho đến khi danh sách chỉ còn lại một nút. Đó chính là nút gốc của cây.

Ở bước 3, quá trình in ra các giá trị mã hóa cho từng phần tử được thể hiện như sau:

- Bước 1: Tạo một mảng phụ để lưu giá trị của mã.

- Bước 2: Duyệt cây Huffman theo kiểu Preorder, khi đi qua cây con bên trái, thêm 0 vào mảng, khi qua cây con bên phải, thêm 1 vào mảng.

- Nếu gặp nút lá thì in ra giá trị có trong mảng. Đó chính là mã đã qua mã hóa của phần tử tại nút lá.

1.2 Pha giải nén

Để giải mã dữ liệu đã được mã hóa, ta cần dựa vào cây Huffman. Sau đó từ cây Huffman để giải mã bằng cách thực hiện các bước sau:

- Bước 1: Bắt đầu từ nút gốc và duyệt lần lượt các bit cho đến khi tìm thấy một nút lá.

- Bước 2: Nếu bit hiện tại là 0, ta di chuyển đến nút bên trái cây, nếu bit hiện tại là 1, di chuyển đến nút bên phải cây

- Bước 3: Nếu bắt gặp một nút lá, in giá trị của nút đó ra và lặp lại quá trình trên từ bước 1

1.3 Điểm mạnh và điểm yếu

1.3.1 Điểm mạnh

Thuật toán static Huffman có một số điểm mạnh sau:

- Dễ cài đặt, dễ hiểu trong hai quá trình encode và decode.
- Tiết kiệm được một lượng lớn không gian lưu trữ, bởi các mã nhị phân tạo ra có độ dài thay đổi, cho phép lưu trữ các ký tự thường xuyên xuất hiện với chi phí thấp hơn các ký tự khác.

Bên cạnh đó, thuật toán trên vẫn có một số điểm yếu như:

- Là một thuật toán nén không mất dữ liệu, thuật toán Huffman đạt được độ nén ít hơn so với các thuật toán mất dữ liệu. Do đó, nó chỉ phù hợp để mã hóa văn bản hay chương trình và không phù hợp để mã hóa hình ảnh kỹ thuật số.
- Là một thuật toán tương đối chậm hơn so với những thuật toán nén không mất dữ liệu khác. Bởi nó yêu cầu hai tiến trình thực hiện
- Vì độ dài của mã nhị phân là khác nhau dẫn đến khó phát hiện xem dữ liệu mã hóa có bị hỏng hay không, dẫn đến việc giải mã sai và có thể đưa ra kết quả sai.

1.4 Ứng dụng

Thuật toán Huffman có một số ứng dụng như sau:

- Mã hóa Huffman có thể được sử dụng trong trường hợp các ký tự cần mã hóa thường xuyên được sử dụng với tần suất xuất hiện lớn.
- Mã hóa Huffman được sử dụng rộng rãi trong các định dạng nén như GZIP, PKZIP (winzip) and BZIP2.
- Các định dạng đa phương tiện như JPEG, PNG and MP3 sử dụng mã hóa Huffman (chính xác hơn là các mã tiền tố).

1.5 Biến thể và cải tiến

Một số biến thể của thuật toán Huffman tĩnh có thể kể đến như n-ary Huffman coding, Adaptive Huffman coding, ...

2 Thuật toán LZW

2.1 Pha nén

2.1.1 Ý tưởng

Ý tưởng của thuật toán LZW encoding dựa trên việc tạo ra một mảng rất lớn để lưu giá trị của các mã(mảng này có thể được gọi là "từ điển")

256 phần tử đầu tiên trong "từ điển" là 256 kí tự của mã Ascii. Các phân tử tiếp theo được tạo cho trường hợp lặp chuỗi trong dữ liệu.

Trong quá trình nén, các kí tự sẽ được nhóm lại với nhau cho đến khi kí tự tiếp theo thêm vào tạo thành một chuỗi chưa xuất hiện trong "từ điển", chuỗi đó sẽ được thêm vào "từ điển".

2.1.2 Trình bày thuật toán

Thuật toán LZW encoding bao gồm các bước sau:

- Bước 1: Tạo một mảng gồm 256 phần tử, mảng này có thể được gọi là "từ điển", với các phần tử đại diện cho các ký tự trong bảng mã Ascii.
- Bước 2: Duyệt lần lượt từng ký tự của chuỗi đầu vào. Gọi ký tự ta đang xét là C và cho một ký tự rỗng P.
- Bước 3: Ta sẽ kiểm tra trong "từ điển" sự xuất hiện của chuỗi C + P . Khi đó sẽ có hai trường hợp xảy ra:
  - Nếu chuỗi C + P đó đã có trong từ điển, gán P = P + C.
  - Ngược lại, xuất ra mã của P và thêm chuỗi P + C vào trong từ điển. Sau đó gán P thành C
- Bước 4: Lần lượt lặp lại bước 3 cho tất cả các ký tự trong chuỗi cần mã hóa, cho đến cuối chuỗi.

2.1.3 Minh họa thuật toán

Dữ liệu minh họa: **msg = AABABCABBA**

P	C	Tìm P + C	Kết quả	Truyền mã	Từ điển
-	A	A	Tìm thấy	-	-
A	A	AA	Không tìm thấy	65 - A	256 - AA
A	B	AB	Không tìm thấy	65 - A	257 - AB
B	A	BA	Không tìm thấy	66 - B	258 - BA
A	B	AB	Tìm thấy	-	-
AB	C	ABC	Không tìm thấy	257 - AB	259 - ABC
C	A	CA	Không tìm thấy	67 - C	260 - CA
A	B	AB	Tìm thấy	-	-
AB	B	ABB	Không tìm thấy	257 - AB	261 - ABB
B	A	BA	Tìm thấy	-	-
BA	-end-	-	-	258 - BA	-

Dự liệu được mã hóa: **65-65-66-257-67-257-258**

2.2 Pha giải nén

2.2.1 Ý tưởng

Mã cần được giải nén sẽ là một mảng các chữ số đại diện cho các ký tự đã được mã hóa. Công việc cần làm là dựa vào các chữ số đó để trả về một chuỗi các ký tự thích hợp.

Giống như pha nén, trong quá trình giải nén cũng cần tạo ra một "từ điển" chứa các mã, vừa thực hiện quá trình giải mã, vừa tạo lại từ điển tương ứng.

2.2.2 Trình bày thuật toán

Quá trình giải mã được thực hiện qua các bước sau:

- Bước 1: Tạo một "từ điển" mới gồm 256 phần tử đại diện cho các phần tử của bảng mã Ascii.
- Bước 2: Lấy phần tử đầu tiên của mảng cần được giải mã, sau đó tìm kiếm trong "từ điển" và xuất ra chuỗi đại diện cho phần tử đó trong mảng.
- Bước 3: Tìm kiếm sự xuất hiện trong "từ điển" của phần tử tiếp theo trong mảng. sẽ có hai trường hợp xảy ra:
  - Nếu phần tử đó xuất hiện trong "từ điển". Khi đó chỉ cần xuất ra chuỗi đại diện cho phần tử đó, đồng thời thêm vào "từ điển" chuỗi bao gồm chuỗi trước đó và ký tự đầu tiên của chuỗi đại diện.
  - Ngược lại, xuất ra chuỗi bao gồm chuỗi trước đó và ký tự đầu tiên của nó, đồng thời thêm lại chuỗi đó vào từ điển.
- Bước 4: Lần lượt duyệt qua từng phần tử tiếp theo và thực hiện các bước tương tự.

2.2.3 Minh họa thuật toán

Với dữ liệu minh họa **AABABCABBA**, ta có mảng mã hóa : 65 65 66 257 67 257 258

Từ mảng mã hóa trên, ta sẽ decode ngược lại để kiểm tra xem có tạo được chuỗi như ban đầu hay không.

Dữ liệu nhận	Từ điển	Giải mã
65	-	A
65	Từ điển[256] = AA	A
66	Từ điển[257] = AB	B
257	Từ điển[258] = BA	AB
67	Từ điển[259] = ABC	C
257	Từ điển[260] = CA	AB
258	Từ điển[261] = ABB	BA

Như vậy, sau khi hoàn thành quá trình decode, ta được chuỗi ban đầu AABABCABBA.

2.3 Điểm mạnh và điểm yếu

2.3.1 Điểm mạnh

Thuật toán LZW có các điểm mạnh sau:

- LZW không yêu cầu thông tin trước về luồng dữ liệu đầu vào.
- LZW có thể nén luồng đầu vào trong một lần chạy.
- LZW không yêu cầu phải lưu lại dữ liệu trong "từ điển" mới có thể decode được, bởi trong quá trình decode có thể tự tái tạo lại, từ đó tiết kiệm một lượng bộ nhớ đáng kể.

2.3.2 Điểm yếu

Bên cạnh đó, thuật toán LZW vẫn tồn tại một số điểm yếu sau:

- Thuật toán này không thật sự hiệu quả đối với những văn bản không có ký tự lặp lại nhiều. Lúc này văn bản sẽ không được nén lại nhiều.

- Thuật toán này không thật sự phù hợp với một số loại tệp khác mà hầu như chỉ hoạt động tốt trên tệp văn bản.

2.3.3 Ứng dụng

Thuật toán LZW có một số ứng dụng sau:

- Nén LZW đã trở thành phương thức nén dữ liệu phổ biến trên máy tính.Một file text English có thể được nén thông qua LZW để giảm $\frac{1}{2}$ dung lượng gốc.
- LZW đã được sử dụng trong phần mềm nén mã nguồn mở,nó đã trở thành 1 phần không thể thiếu trong HDH UNIX CIRCA 1986.
- LZW đã trở nên phổ biến khi nó được sữ dụng làm 1 phẩn của file GIF năm 1987.Nó cũng có thể được sử dụng trong TIFF và PDF file.

2.4 Biến thể và cải tiến

Một số biến thể của thuật toán Huffman tĩnh có thể kể đến như LZMW, MZAP, LZWL...

3 Các lưu ý về lập trình

3.1 Trình bày thuật toán LZW

Về việc giải quyết các vấn đề liên quan tới phần lập trình *Nén và Giải nén file văn bản sử dụng thuật toán nén LZW* thì đã được trình bày ở mục "**2. Thuật toán LZW**".

Để tạo từ điển thì ta sẽ có 2 hàm riêng tương ứng với 2 cấu trúc dữ liệu *map* trong ngôn ngữ C++. Việc sử dụng *map* để truy cập từ điển sẽ khiến ta tiết kiệm được rất nhiều thời gian cũng như không gian cho cả 2 phần nén và giải nén, từ việc thêm, xóa, truy vấn các phần tử trong từ điển. Nhận thấy cấu trúc dữ liệu *map* trong C++ được triển khai dưới dạng cấu trúc dữ liệu là cây đỏ đen tự cân bằng.

Ta dùng string để lưu dữ liệu chuỗi vì C++ hỗ trợ khá nhiều cho việc sử dụng string khiến ta dễ dàng hơn trong việc xử lí dữ liệu.

Ta dùng vector để lưu dữ liệu nhị phân để tiện hơn trong việc thêm, truy cập, cũng như lấy độ dài của vector.

3.2 Xử lí Command

Từ việc 2 phần nén và giải nén khác nhau, dẫn đến việc xử lí command cũng chia làm 2 hướng giải quyết khác nhau cho cả 2 phần.

Cách tối ưu và hiệu quả nhất là chia ra 2 trường hợp, sau đó đi giải quyết các trường hợp đơn lẻ khác trong từng trường hợp đó, nếu như dữ liệu đưa vào bị lỗi thì in ra thông báo và kết thúc chương trình.

3.3 Cấu trúc file trong thư mục Source

File được chia làm 3 phần:

- *command_line.cpp* và *command_line.h*: xử lí toàn bộ dữ liệu input đưa vào để xác định command, cũng như cho ra output tương ứng.

- *lzw.cpp* và *lzw.h*: xử lí thuật toán nén và giải nén LZW.
- *main.cpp*: chứa hàm main để chạy chương trình.

3.4 Các thư viện đặc biệt được sử dụng

- <string.h>: xử lí các vấn đề liên quan đến kiểu dữ liệu chuỗi.
- <map>: dùng trong việc tạo từ điển.
- <fstream>: xử lí đọc dữ liệu từ file và ghi kết quả vào file.
- <vector>: tạo mảng để chứa các phần tử nhị phân.

4 Tài liệu tham khảo

1. LZW (Lempel–Ziv–Welch) Compression technique - GeeksforGeeks. (2017, April 26). Retrieved from GeeksforGeeks website: https://www.geeksforgeeks.org/lzw-lempel-ziv-welch-compression-technique/

2. Tuấn, N. (2021, July 2). Thuật toán nén Huffman Coding. Retrieved November 19, 2022, from Nguyễn Tuấn's Blog website: https://chidokun.github.io/2021/07/huffman-coding-p1/

3. Huffman Coding | Greedy Algo-3. (2012, November 3). Retrieved from GeeksforGeeks website: https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/

4. LZW coding 1. (n.d.). Retrieved from www.youtube.com website: $https://www.youtube.com/watch?v=rZ-JRCPv_O8$

5. MIT OpenCourseWare. (2012). MIT 6.02 - Introduction to EECS II - Digital communication systems fall 2022. Retrieved November 18, 2022, from $https://ocw.mit.edu/courses/6-02-introduction-to-eecs-ii-digital-communication-systems-fall-2012/resources/mit6_02f12_chap03$

6. Wikipedia. (2006, September). Mã hóa Huffman. Retrieved November 19, 2022, from Wikipedia.org website: https://vi.wikipedia.org/wiki/M%C3%A3_h%C3%B3a_Huffman

7. Wikipedia. (n.d.). Lempel–Ziv–Welch. Retrieved November 18, 2022, from https://en.wikipedia.org/wiki/Lempel%E2%80%93Ziv%E2%80%93Welch

8. Hash Table vs STL Map. (2022, November 8). Retrieved November 19, 2022, from https://www.geeksforgeeks.org/hash-table-vs-stl-map/