**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KHOA HỌC MÁY TÍNH**

**BÁO CÁO ĐỒ ÁN**

**TÍNH TOÁN ĐA PHƯƠNG TIỆN**

**- CS232.N21.KHCL –**

**TÊN ĐỀ TÀI:**

**Thực hiện và đoán giá hai thuật toán nén Huff-man và LZW trên kiểu dữ liệu text**

Giảng viên hướng dẫn:

Thầy Đỗ Văn Tiến.

Sinh viên thực hiện:

Từ Trương Tuấn Kiệt – 19521727

Nguyễn Tô Thiên Bảo – 21521857

Hồ Trung Tín – 21521536

**Mục lục:**

[1. ***Giới thiệu đề tài.*** 3](#_Toc139888823)

[2. ***Tìm hiểu thuật toán HUFFMAN.*** 3](#_Toc139888824)

[*a)* *Giới thiệu về thuật toán:* 3](#_Toc139888825)

[*b)* *Ý tưởng:* 3](#_Toc139888826)

[*c)* *Cách xây dựng:* 4](#_Toc139888827)

[*d)* *Hiệu suất nén:* 6](#_Toc139888828)

[3. **Tìm hiểu thuật toán LZW:** 8](#_Toc139888829)

[*a)* *Giới thiệu về thuật toán:* 8](#_Toc139888830)

[*b)* *Ý tưởng:* 8](#_Toc139888831)

[*c)* *Cách xây dựng:* 8](#_Toc139888832)

[*d)* *Hiệu xuất nén:* 10](#_Toc139888833)

[4. ***Triển khai thuật toán:*** 11](#_Toc139888834)

[5. ***Đánh giá dựa trên tập dữ liệu:*** 15](#_Toc139888835)

[6. ***Kết luận:*** 18](#_Toc139888836)

[7. ***Bảng phân công công việc:*** 19](#_Toc139888837)

1. ***Giới thiệu đề tài.***
2. *Nén dữ liệu là gì?*

Thông thường, hầu hết các tập tin trong máy tính có rất nhiều thông tin dư thừa, việc thực hiện nén tập tin thực chất là mã hoá lại các tập tin để loại bỏ các thông tin dư thừa.

1. *Các hình thức nén dữ liệu:*

* Nén bảo toàn thông tin (Lossless Compression): Không mất mát thông tin nguyên thuỷ, hiệu suất nén không cao: 10% - 60%
  + - Các giải thuật tiêu biểu: RLE, Arithmetic, Huffman, LZ77, LZ78,…
* Nén không bảo toàn thông tin (Lossy Compression): Thông tin nguyên thủy bị mất mát, hiệu suất nén cao 40% - 90%
  + - Các giải thuật tiêu biểu: JPEG, MP3, MP4

1. *Phạm vi đề tài:*

Tập trung vào nén dữ liệu dạng văn bản sử dụng hai thuật toán Huffman và LZW. Đây là hai thuật toán nén dữ liệu phổ biến được sử dụng trong việc giảm kích thước của tệp tin.

1. ***Tìm hiểu thuật toán HUFFMAN.***
   * 1. *Giới thiệu về thuật toán:*

Phương pháp Huffman là một thuật toán do David A. Huffman của MIT đưa ra vào năm 1952 để nén dữ liệu dạng văn bản, mã hóa các bytes trong tệp dữ liệu nguồn bằng biến nhị phân.

* + 1. *Ý tưởng:*

Mã hóa các bytes trong tệp dữ liệu nguồn bằng biến nhị phân. Nó tạo mã độ dài biến thiên là một tập hợp các bits. Đây là phương pháp nén kiểu thống kê, những ký tự xuất hiện nhiều hơn sẽ có mã ngắn hơn.

A picture containing text, screenshot, number, font

Description automatically generated

* + 1. *Cách xây dựng:*
* Bước 1: Đếm tần suất xuất hiện của các phần tử trong chuỗi đầu vào.
* Bước 2: Xây dựng cây Huffman (cây nhị phân mã hóa) với quy ước bên trái mã 0, bên phải mã 1.
* Bước 3: Từ cây Huffman, ta có được các giá trị mã hóa. Lúc này, ta có thể xây dựng chuỗi mã hóa từ các giá trị này.
* Quá trình xây dựng cây Huffman gồm các bước sau:
* Tạo danh sách chứa các nút lá bao gồm phần tử đầu vào và trọng số nút là tần suất xuất hiện của nó.
* Từ danh sách này, lấy ra 2 phần tử có tần suất xuất hiện ít nhất. Sau đó gắn 2 nút vừa lấy ra vào một nút gốc mới với trọng số là tổng của 2 trọng số ở nút vừa lấy ra để tạo thành một cây.
* Đẩy cây mới vào lại danh sách.
* Lặp lại bước 2 và 3 cho đến khi danh sách chỉ còn 1 nút gốc duy nhất của cây.
* Nút còn lại chính là nút gốc của cây Huffman.

**Vd**: Thực hiện nén chuỗi “Hellooo!”.

* *Bước 1*:

*A group of circles with letters and numbers

Description automatically generated with low confidence*Sau khi đếm tần suất xuất hiện các phần tử đầu vào. Chúng ta tạo danh sách các nút lá với trọng số là tần suất xuất hiện. Danh sách sẽ có 5 phần tử như bên dưới.

* *Bước 2*: Xây dựng cây Huffman
  + - *A picture containing circle, diagram, design

      Description automatically generatedBước 2.1:* Chọn 2 nút có trọng số thấp nhất, tạo nút gốc mới có trọng số bằng tổng 2 trọng số nút con. Sau đó gắn 2 nút con vào nút gốc và đẩy lại vào danh sách. Danh sách cần được biểu diễn đặc biệt để có thể lấy ra các nút trọng số nhỏ nhất một cách tối ưu nhất.
* A picture containing sketch, diagram, circle, drawing

  Description automatically generated*Bước 2.2:* Lặp lại bước 2.1 cho đến khi chỉ còn lại 1 nút trong danh sách, nút này chính là cây Huffman.
* *A picture containing diagram, drawing, sketch, circle

  Description automatically generatedBước 3:* Từ cây Huffman, ta có thể suy ra các giá trị mã hóa của từng phần tử bằng cách duyệt cây nhị phân mã hóa.
  + 1. *Hiệu suất nén:*
* Hiệu suất nén của thuật toán Huffman phụ thuộc vào cấu trúc của dữ liệu đầu vào và tần suất xuất hiện của các ký tự trong dữ liệu đó. Ký tự thường xuất hiện nhiều hơn sẽ được mã hóa bằng một số lượng nhỏ hơn các bit, trong khi ký tự ít xuất hiện sẽ được mã hóa bằng một số lượng bit lớn hơn.
* Hiệu suất nén = (Kích thước dữ liệu ban đầu - Kích thước dữ liệu sau khi nén) / Kích thước dữ liệu ban đầu \* 100

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Kí tự** | **a** | **e** | **l** | **n** | **o** | **s** | **t** |
| **Codeword** | 110 | 10 | 0110 | 111 | 0111 | 010 | 00 |
| **Codeword bits** | 3 | 2 | 4 | 3 | 4 | 3 | 2 |
| **Tần số xuất hiện của các ký tự** | 45 | 65 | 13 | 45 | 18 | 22 | 53 |
| **Tổng bits** | 135 | 130 | 52 | 135 | 72 | 66 | 106 |

* + **Cách tính chi tiết:** *với mẫu là một chuỗi gồm 261 kí tự với bảng tần số và codeword như dưới đây*
* Tổng số bits của dữ liệu khi được nén: 696 bits
* Bảng mã:

7

**a**110

**e**10

**l**0110

**n**111

**o**0111

**s**010

**t**00

* Tổng số bits để lưu bảng mã:

bits(7=111) + bits(kí tự) + bits(codewords) = 3 + 7\*8 + 21 = 80

* Tổng số bits của file sau khi nén: 696 + 80 = 776;
  + - Tổng số bits với chuỗi: 261 \* 8 = 2088
    - Tỉ lệ nén: 2088/776 = 2.69
    - **Hiệu suất nén: 62,84%**

1. **Tìm hiểu thuật toán LZW:**
2. *Giới thiệu về thuật toán:*

Thuật toán LZW là một trong những thuật toán nén dữ liệu phổ biến nhất. Nó được phát triển vào những năm 1970 bởi Alistair Moffat, Terry Welch và Abraham Lempel, Toda, và Welch. Thuật toán này được sử dụng rộng rãi trong các định dạng tệp nén như GIF, TIFF, và PDF..

1. *Ý tưởng:*

Người mã hoá và người giải mã cùng xây dựng bảng mã. Bảng mã này không cần được lưu kèm với dữ liệu trong quá trình nén, mà khi giải nén, người giải nén sẽ xây dựng lại nó.

Bảng mã được xây dựng với ý tưởng sau:

* Mã từ 0 đến 255 miêu tả 1 dãy ký tự thay thế cho ký tự 8-bit tương ứng.
* Mã từ 256 đến 4095 được tạo bên trong 1 từ điển cho trường hợp lặp chuỗi trong dữ liệu.
* Mỗi bước trong khi nén, byte nhập vào được tập hợp lại thành một chuỗi cho đến khi ký tự tiếp theo sẽ tạo thành 1 chuỗi chưa tồn tại trong từ điển, một mã mới cho sẽ được thêm vào từ điển và kí tự cuối cùng của chuỗi sẽ được giữ lại để tạo thành chuỗi mới.

1. *Cách xây dựng:*

* Quá trình nén sẽ được thực hiện qua 5 bước:
  + Khởi tạo từ điển chứa tất cã chuỗi có 1 ký tự;
  + Tìm chuỗi W dài nhất trong từ điển đối chiếu với dữ liệu nhập hiện tại;
  + Xuất vị trí từ điển của W ra file output và xóa W khỏi dữ liệu nhập;
  + Thêm W và ký tự tiếp theo trong dữ liệu nhập vào từ điển
  + Lặp lại bước 2 cho đến khi hết dữ liệu nhập.
* Quá trình giải nén:

Ta sẽ đọc giá trị từ dữ liệu nhập đã mã hóa và xuất ra chuỗi tương ứng từ từ điển đã được khởi tạo.

* + Tìm chuỗi W có giá trị tương ứng với dữ liệu nhập trong từ điển:
    - Trường hợp 1: tìm được chuỗi W, ta sẽ lấy giá trị đó xuất ra output
    - Trường hợp 2: không tìm được, ta sẽ gán chuỗi W bằng giá trị chuỗi output trước đó + kí tự đầu tiên của chuỗi đó;
  + Thêm một chuỗi gồm chuỗi output trước đó và kí tự đầu của chuỗi W vào từ điển;
  + Thực hiện liên tục cho đến khi hết dữ liệu cần giải nén.
* Ví dụ: thực hiện nén và giải nén chuỗi: BABAABAAA
  + Nén:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Current** | **Next** | **Output** | **Dictionary** | |
| **Codeword** | **String** |
| B | A | 66 | 256 | BA |
| A | B | 65 | 257 | AB |
| BA | A | 256 | 258 | BAA |
| AB | A | 257 | 259 | ABA |
| A | A | 65 | 260 | AA |
| AA | NULL | 260 |  |  |

* *Kết quả nhận được sau khi nén: 66 65 256 257 65 260*
  + Giải nén: 66 65 256 257 65 260

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Current** | **Last output** | **Output** | **Dictionary** | |
| **Codeword** | **String** |
| 66 | - | B | - | - |
| 65 | B | A | 256 | BA |
| 256 | A | BA | 257 | AB |
| 257 | BA | AB | 258 | BAA |
| 65 | AB | A | 259 | ABA |
| 260 | A | AA | 260 | AA |

* *Kết quả nhận được sau khi nén: BABAABAAA*

1. *Hiệu xuất nén:*

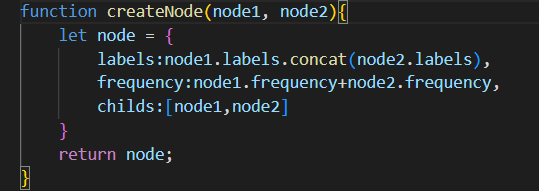
Trong thuật toán nén này, phần lớn thời gian khi bắt đầu nén chủ yếu mất vào việc tạo "từ điển". Khi "từ điển" đủ lớn, xác suất gặp chuỗi ở bộ đệm chứa trong "từ điển" tăng lên và càng nén được nhiều hơn.

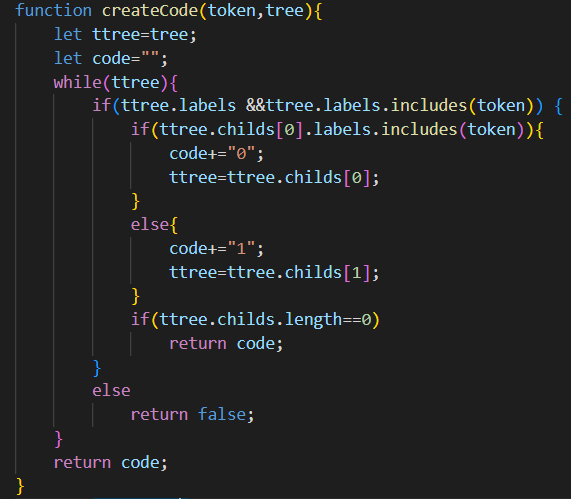
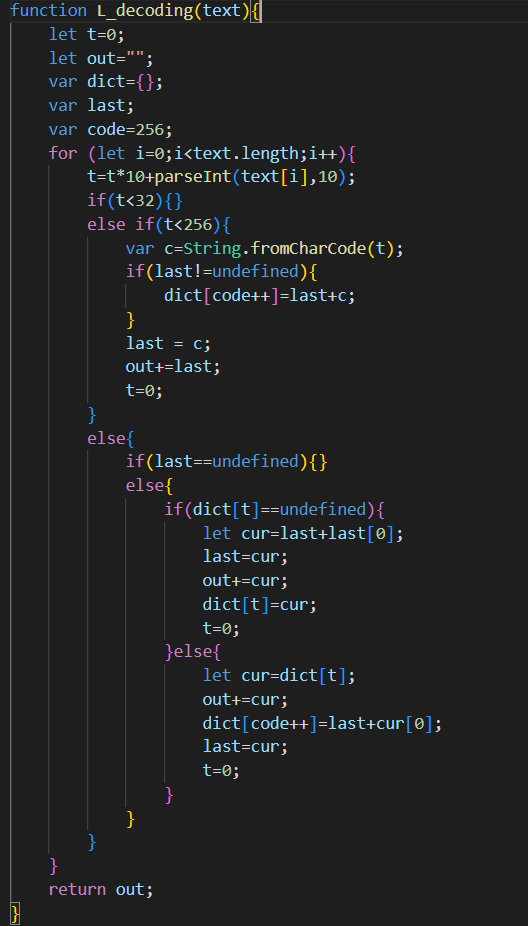
Hiệu suất nén = (Kích thước dữ liệu ban đầu - Kích thước dữ liệu sau khi nén) / Kích thước dữ liệu ban đầu \* 100%

* *Với kích thước dữ liệu ban đầu bằng: length(input) \* 8 bits (vì mỗi kí tự trong bảng mã ASCII đều cần 8 bits)*
* *Kích thước dữ liệu sau khi nén bằng tổng số bits của từng chuỗi:*
* *Nếu chuỗi có 1 kí tự thì cần: 8 bits*
* *Nếu chuỗi có nhiều hơn 1 kí tự cần: bits\_length(codeword)*
* Ví dụ: Thực hiện nén chuỗi:
  + Input: BABAABAAA
  + Output 66 65 256 257 65 260
* Tổng số bits input: length(input) \* 8 = 9 \* 8 = 72 bits
* Tổng số bits output: 3\*8 + 3\*9 = 51 bits
* Tỉ lệ nén: 72 / 51 = 1.41
* **Hiệu suất nén: 26,34%**

1. ***Triển khai thuật toán:***

*Demo:* [*https://jbz26.github.io/LZW\_HUFFMAN/*](https://jbz26.github.io/LZW_HUFFMAN/)

1. Thuật toán Huffman:
   * Hàm tạo một node:
   * A picture containing text, screenshot, font

     Description automatically generatedHàm tạo cây Huffman:
   * Hàm tạo mã cho các kí tự:
2. Thuật toán LZW:
   * Hàm tạo bảng mã LZW:
   * Hàm giải mã:
3. ***Đánh giá dựa trên tập dữ liệu:***

* ***Test 1:*** Ta thử với chuỗi “1234567890qwertyuiop”

**A picture containing text, number, parallel, receipt

Description automatically generatedA picture containing text, number, screenshot, parallel

Description automatically generatedHuffman LZW**

* ***Test 2:*** Ta thử với 5 lần chuỗi  “1234567890qwertyuiop”

**A picture containing text, number, parallel, line

Description automatically generatedA picture containing text, screenshot, number, parallel

Description automatically generated Huffman LZW**

* ***Test 3:*** Ta thử với 10 lần chuỗi  “1234567890qwertyuiop”

A picture containing text, number, line, receipt

Description automatically generatedA picture containing text, number, parallel, line

Description automatically generated**Huffman LZW**

Qua 3 test ở trên, ta có thể thấy được thuật toán LZW sẽ hoạt động tốt ở những tập dữ liệu lớn và có nhiều sự trùng lặp, còn thuật toán huffman có tính ổn định cao hơn.

Điều này là bởi vì thuật toán LZW cần một từ điển đủ lớn để có  tỉ lệ nén tốt, nếu dữ liệu quá nhỏ hoặc thiếu sự trùng lặp thì việc tạo một từ điển đầy đủ sẽ trở nên khó khăn hơn dẫn đến tỉ lệ nén không được cao. Còn đối với những tập dữ liệu lớn, có nhiều sự trùng lặp thì việc xây dựng từ điển sẽ trở nên dễ dàng hơn giúp số lượng bit về sau giảm đi đáng kể làm tăng cao tỉ lệ nén.

Còn đối với thuật toán Huffman sẽ hoạt động tốt ở những tập dữ liệu bé hơn vì việc thực hiện tạo cây Huffman tương đối đơn giản và ổn định do tỉ lệ nén chỉ tăng lên khi cây có nhiều nút khác nhau dẫn đến chiều cao cây tăng làm số lượng bit của mỗi phần tử tăng nên nếu xảy ra nhiều sự trùng lặp thì cây vẫn giữ nguyên và không thay đổi.

1. ***Kết luận:***

Thuật toán Huffman có ưu điểm là hệ số nén tương đối cao và ổn định, phương pháp thực hiện tương đối đơn giản, đòi hỏi ít bộ nhớ hơn. Tuy nhiên trong mỗi tập tin nén của nó đều phải chứa cả bảng mã thì phía nhận mới có thể giải mã được, điều này cũng có thể gây bất lợi cho việc sử dụng thuật toán này.

Điều này là bởi vì thuật toán LZW sử dụng khá nhiều thời gian cho việc tạo từ điển nên thuật toán này sẽ hoạt động không tốt với những tập dữ liệu nhỏ và thiếu sự trùng lặp vì việc tạo từ điển sẽ chiếm phần lớn thời gian khiến tỉ lệ nén không được cao. Còn đối với những tập dữ liệu lớn, có nhiều sự trùng lặp thì lúc này từ điển đã được tạo khá đầy đủ giúp nén được nhiều hơn ở giai đoạn sau.

Qua ví dụ ở trên, ta có thể thấy rằng trong 2 thuật toán LZW và Huffman thì không có thuật toán nào vượt trội hơn hẳn thuật toán còn lại mà còn tuỳ vào những trường hợp, những loại tệp tin mà chúng ta mới có thể lựa chọn thuật toán tối ưu nhất để mang lại hiệu suất tốt nhất.

|  |  |  |
| --- | --- | --- |
| ***MSSV*** | ***Thành viên*** | ***Nội dung*** |
| 19521727 | Từ Trương Tuấn Kiệt | Tìm hiểu thuật toán Huffman |
| Thực hiện phần thuyết trình |
| 21521857 | Nguyễn Tô Thiên Bảo | Tìm hiểu thuật toán LZW |
| Viết bài báo cáo file word |
| Thực hiện demo code |
| 21521536 | Hồ Trung Tín | Thực hiện so sánh, đánh giá hai thuật toán |
| Tạo trang trình bày |

1. ***Bảng phân công công việc:***