**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

**BÁO CÁO PROJECT I**

**ÁP DỤNG THUẬT TOÁN BURROWS-WHEELER-TRANSFORM TRONG VIỆC NÉN DỮ LIỆU**

**PHẠM MINH KHÔI – 20183566 – IT1-02-K63**

Khoi.pm183566@sis.hust.edu.vn

**Ngành Khoa học Máy tính**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:** | TS. Trần Vĩnh Đức |
| **Bộ môn :** | Project 1 |
| **Viện :** | Công nghệ thông tin và Truyền thông |

**HÀ NỘI, 01/2021**

**Lời cảm ơn**

Trong thời gian làm bài tập lớn môn Project I, em đã nhận được rất nhiều sự giúp đỡ, đóng góp ý kiến của thầy và các bạn.

Em xin gửi lời cảm ơn chân thành đến TS. Trần Vĩnh Đức, giảng viên Viện Công nghệ thông tin và truyền thông, trường Đại học Bách khoa Hà Nội, người trực tiếp hướng dẫn em hoàn thành bài tập lớn môn Project I.

Em cũng xin cảm ơn các thầy cô, bạn bè đã luôn tạo điều kiện quan tâm, giúp đỡ để em hoàn thành học phần này.

Em rất vinh dự nếu nhận được sự góp ý của thầy cô và các bạn để hoàn thiện đề tài này hơn.

Em xin chân thành cảm ơn!

**Tóm tắt nội dung**

* Vấn đề cần thực hiện: áp dụng thuật toán burrows-wheeler transform trong nén dữ liệu.
* Phương pháp thực hiện:
  + Tìm hiểu tổng quan về thuật toán burrows-wheeler và các cải tiến.
  + Cài đặt thử nghiệm chương trình nén bằng ngôn ngữ C++.
* Kết quả: Nén và giải nén thành công các file đầu vào.
* Tính thực tế: Chương trình giúp trong việc tránh mất mát dữ liệu khi nén.
* Định hướng phát triển: Tìm cách cải tiến mới tối ưu tốc độ và bộ nhớ, đồng thời kết hợp với các phương pháp nén khác…

Sinh viên thực hiện

Ký và ghi rõ họ tên

Phạm Minh Khôi

**MỤC LỤC**

[CHƯƠNG 1. PHÁT BIỂU VÀ GIẢI QUYẾT BÀI TOÁN 4](#_Toc60267668)

[1.1 Phát biểu bài toán 4](#_Toc60267669)

[1.2 Phương pháp thực hiện 4](#_Toc60267670)

[1.2.1 Burrows-Wheeler Transform 4](#_Toc60267671)

[1.2.2 Move-to-front Transform 6](#_Toc60267672)

[1.2.3 Run-length Encoding 6](#_Toc60267673)

[1.2.4 Huffman Coding 7](#_Toc60267673)

[CHƯƠNG 2. KẾT QUẢ THỬ NGHIỆM VÀ ĐÁNH GIÁ 9](#_Toc60267674)

[CHƯƠNG 3. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 9](#_Toc60267674)

[TÀI LIỆU THAM KHẢO 10](#_Toc60267677)

**DANH MỤC HÌNH ẢNH**

[Hình 1. Tạo ma trận. 4](#_Toc60267678)

[Hình 2. Sắp xếp các dòng trong ma trận 4](#_Toc60267679)

[Hình 3. Trích xâu từ ký tự cuối 5](#_Toc60267680)

[Hình 4. Thứ tự của xâu gốc trong ma trận sau sắp xếp 5](#_Toc60267681)

[Hình 5. Thuật toán SA-IS. 5](#_Toc60267682)

[Hình 6. Lấy các kí tự trong xâu BWT. 5](#_Toc60267683)

[Hinh 7. Lặp 6](#_Toc60267684)

[Hình 8. Thuật toán MTF thuận 6](#_Toc60267685)

[Hình 9. Thuật toán MTF nghịch 6](#_Toc60267686)

[Hình 10. RLE thuận 7](#_Toc60267687)

[Hình 11. RLE nghịch 7](#_Toc60267688)

[Hình 12. Cây Huffman 7](#_Toc60267689)

[Hình 13. Huffman Coding 8](#_Toc60267690)

[Hình 14. Huffman Decoding 8](#_Toc60267691)

[Hình 15. Chuyển xâu nhị phân thành xâu kí tự. 8](#_Toc60267692)

[Hình 16. Chuyển xâu kí tự thành xâu nhị phân. 8](#_Toc60267693)

[Hình 17. Đọc và ghi file. 9](#_Toc60267693)

**Chương 1.Phát biểu vấn đề và giải quyết bài toán.**

1.1 Phát biểu bài toán.

* Thực hiện được biến đổi Burrows -Wheeler thuận và nghịch.
* Đọc dữ liệu từ file đầu vào, nén và giải nén.
* Cải tiến kỹ thuật bằng cách kết hợp với các phương thức nén khác.

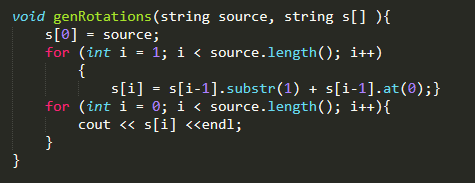
1.2 Phương pháp thực hiện.

1.2.1. Burrows-Wheeler Transform (BWT).

* Đầu vào: Văn bản X gồm n kí tự X[1….n] trên một bảng chữ cái (ASCII)
* Đầu ra: một chuỗi chuyển đổi BWT.

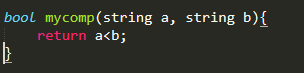
Từng bước xác định chuỗi chuyển đổi BWT được thực hiện :

Bước 1: Tạo một ma trận L\*L (với L là độ dài xâu ký tự) chứa các xâu dịch chuyển vòng tròn.



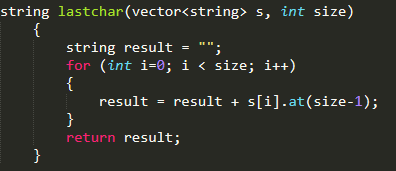
Hình 1. Tạo ma trận.

Bước 2: Sắp xếp các dòng của ma trận theo thứ tự từ điển.

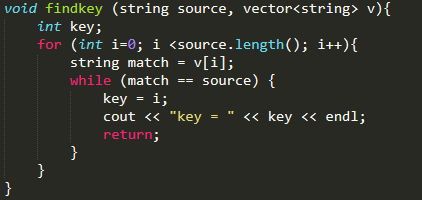


Hình 2. Sắp xếp các dòng trong ma trận.

Bước 3: Trích xâu từ các kí tự cuối mỗi dòng và cho biết thứ tự của xâu gốc trong ma trận sau sắp xếp.

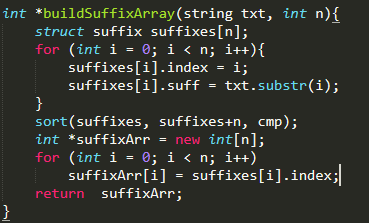


Hình 3. Trích xâu từ ký tự cuối



Hình 4. Thứ tự của xâu gốc trong ma trận sau sắp xếp

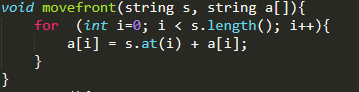
Để hạn chế việc tốn kém dung lượng khi lưu trữ và sắp xếp ma trận xoay vòng, kỹ thuật BWT đã được cải tiến bằng việc sử dụng thuật toán SA-IS (Suffix Array- Induced Sorting) với thời gian thực hiện là tuyến tính.



Hình 5. Thuật toán SA-IS.

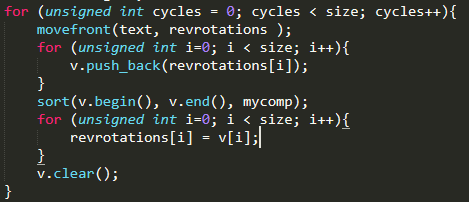
Khôi phục văn bản gốc với chuyển đổi BWT nghịch được đơn giản thực hiện như sau:

Bước 1: Lấy các ký tự trong xâu BWT, sắp xếp lại theo thứ tự từ điển.



Hình 6. Lấy các kí tự trong xâu BWT.

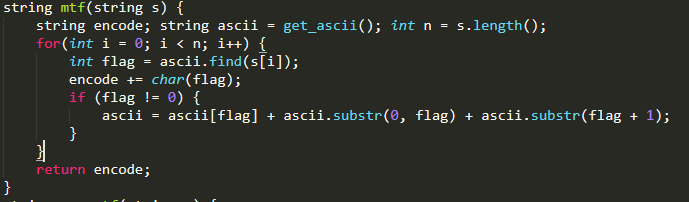
Bước 2: Lấy các ký tự đó thêm vào xâu đã được sắp xếp, lặp lại cho đến khi nhận kích thước ban đầu.



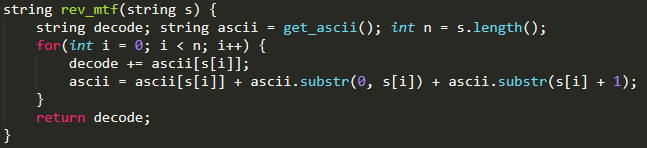
Hình 7. Lặp

**1.2.2. Move-to-front transform (MTF).**

Chuyển đổi BWT không làm chuỗi ban đầu ngắn đi, nhưng lại làm xuất hiện các ký tự giống nhau hoặc rất gần với sự xuất hiện trước đó. Do đó, kỹ thuật nén Move-to-front được sử dụng ngay sau BWT để xếp hạng các kí tự theo tần số tương quan của chúng, nhằm đạt hiệu suất cao hơn cho việc mã hóa.

Ý tưởng của MTF là đưa ra các kí tự có tần xuất sử dụng nhiều lên phía trước trong bảng chữ cái, cuối cùng xâu được mã hóa thành một chuỗi số, và các số này có xu hướng là các số đơn vị (là số nhỏ).

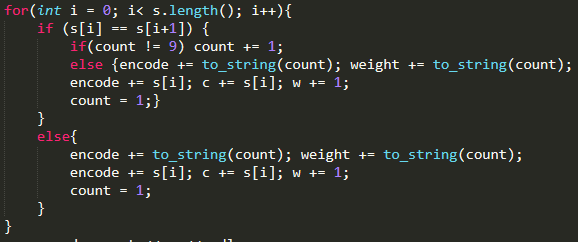
Hình 8.Thuật toán MTF thuận.



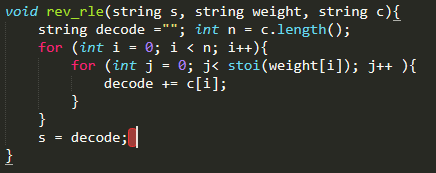
Hình 9. Thuật toán MTF nghịch.

**1.2.3. Run-length Encoding.**

Sau biến đổi MTF, chuỗi kết quả sẽ có nhiều kí tự liên tiếp giống nhau, do đó có thể áp dụng RLE để làm giảm kích thước. Để dễ mã hóa, xâu kết quả được chia làm hai: một xâu chứa ký tự và một xâu chứa số lần lặp. Ở đây các kí tự sẽ được lưu tối đa là 9 để đảm bảo sự bằng nhau của kích thước hai xâu.



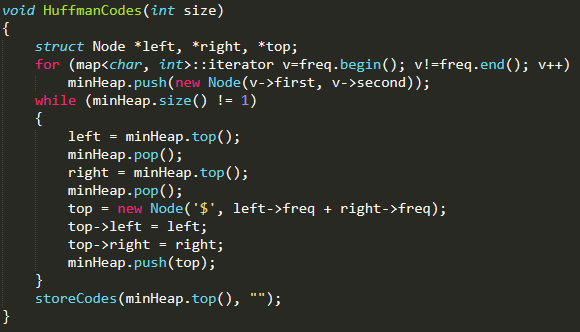
Hình 10. RLE thuận.



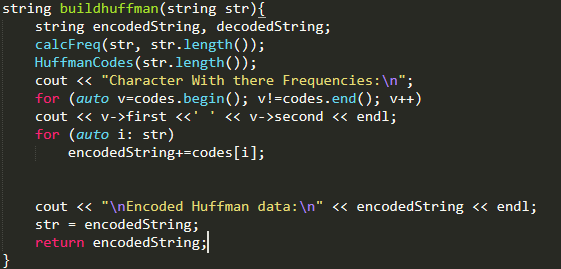
Hình 11. RLE nghịch.

**1.2.4. Huffman Coding.**

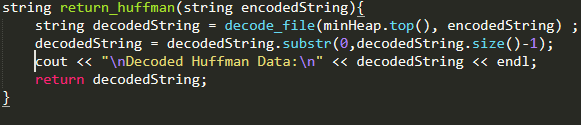
Sau khi đã tính được tần suất của các kí tự, mã hóa huffman sẽ dựa trên đó để xây dựng một bộ mã nhị phân sao cho dung lượng (số bit) là ít nhất, sau đó hoàn tất việc nén file gốc vào thành 1 file txt.



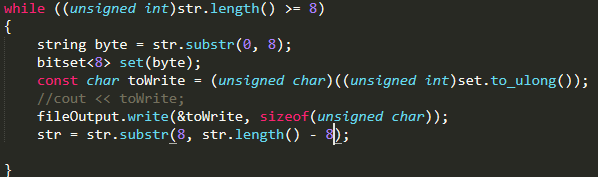
Hình 12.Xây dựng cây Huffman.



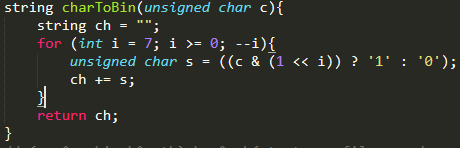
Hình 13. Huffman Coding.



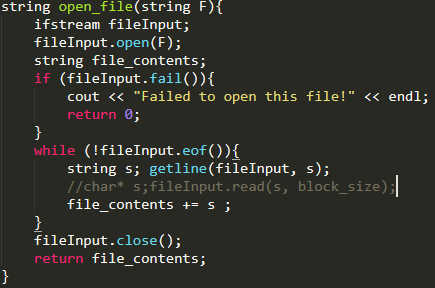
Hình 14. Huffman Decoding.



Hình 15. Chuyển xâu nhị phân thành xâu kí tự.



Hình 16. Chuyển xâu kí tự thành xâu nhị phân.



Hình 17. Đọc và ghi file.

**Chương 2.Kết quả thử nghiệm và đánh giá.**

Chương trình được chạy thử trên máy tính với CPU Intel Core(TM) i5-4300 2GHz, RAM 4GiB. Hệ điều hành Win10Pro, 64bit.

|  |  |  |  |
| --- | --- | --- | --- |
| Dung lượng file đầu vào | Dung lượng file nén | Thời gian nén | Thời gian giải nén |
| 3,237 B | 1,571 B | 0,015s | 0,013s |
| 74,239 B | 44,025 B | 0,51s | 0,22s |
| 196,183 B | 116,725 B | 12,773s | 2,542s |
| 1MB | 731KB | 80s | 12s |

Đánh giá: Chương trình chạy tương đối ổn định, xử lý gọn các file có dung lượng thấp, các file lớn có thời gian chạy tính bằng phút. Vẫn tồn tại các lỗi như không đọc được ký tự đặc biệt từ file nén, chưa thể kết hợp được rle với bwt, thời gian nén và giải nén chưa tối ưu….Vì vậy cần phải cải thiện nhiều để có thể ứng dụng trong thực tế. Tuy nhiên, em cũng đã cố gắng hoàn thành trong thời gian được giao và thu được kết quả như trên.

**Chương 3.Kết luận và hướng phát triển.**

* Về cơ bản đã xây dựng được một chương trình giải quyết được đầy đủ các yêu cầu của bài toán.
* Ưu điểm là cách thức hoạt động dễ nhìn, dễ hiểu; các thành phần chương trình được phân chia một cách rõ ràng, có thể hoạt động được độc lập.
* Tuy nhiên chương trình có thể phát triển thêm được theo nhiều hướng:
  + Sử dụng wchar, wstring để nén 32bit (UTF-16) trong huffman.
  + Chia nhỏ file dung lượng lớn (tầm vài Gb) thành các block, xử lý giải nén huffman trong các block.
  + Cải thiện trong việc nén các ký tự đặc biệt.

Source code: <https://github.com/ghuioio/project1/tree/main/project1%20v1.1>

**TÀI LIỆU THAM KHẢO**

[1] Burrows, Michael; Wheeler, David J. (1994), A block sorting lossless data compression algoritm, Technical Report 124, Digital Equipment Corporation.

[2] Manzini, Giovanni(1999-08-18). “The Burrows-Wheeler Transform: Theory and Practice”. Mathematical Foundations of Computer Science 1999: 24th International Symposium, MFCS’99 Szklarska Poreba, Poland, September 6-10, 1999 Proceedings. Springer Science & Business Media. ISBN 973540664086.

[3] <https://github.com/boss14420/BWT>

[4] <https://www.geeksforgeeks.org/inverting-burrows-wheeler-transform/>

[5] <https://www.geeksforgeeks.org/burrows-wheeler-data-transform-algorithm/>