ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO

HỌC PHẦN PROJECT I

ĐỀ TÀI:

TÌM HIỂU VÀ CÀI ĐẶT THUẬT TOÁN BURROWS-WHEELER TRANSFORM

Sinh viên thực hiện : NGUYỄN VĂN LỘC – MSSV: 20183576

Lớp : Khoa học máy tính 04

Giảng viên hướng dẫn : TS.TRẦN VĨNH ĐỨC

Hà Nội, ngày tháng năm 2020

ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO

HỌC PHẦN PROJECT I

ĐỀ TÀI:

TÌM HIỂU VÀ CÀI ĐẶT THUẬT TOÁN BURROWS-WHEELER TRANSFORM

Hà Nội, ngày tháng năm 2020

MỤC LỤC

Table of Contents

PHẦN 1: THUẬT TOÁN BURROWS-WHEELER DATA COMPRESSION

1.1 Tổng quan về thuật toán Burrows-Wheeler Data Compression

Burrows và Wheeler đã công bố "Block-sorting Lossless Data Compressing Algorithm" vào năm 1994, thuật toán mang tính cách mạng này giải nén gzip và PKZIP, nó tương đối dễ thực hiện, và không bị bảo vệ bởi bất kì bằng sáng chế nào. Nó tạo thành nền tảng của bzip2 tiện dụng cho nén Unix. Thuật toán nén Burrows-Wheeler bao gồm 3 thành phần thuật toán khác nhau, được áp dụng liên tiếp:

1. *Burrows-Wheeler transform (BWT) :* đầu vào là một tệp văn bản tiếng Anh bình thường, nó sẽ được biến đổi sau đó cho ra đầu ra là một tệp văn bản trong đó chuỗi các chữ cái giống nhau sẽ xuất hiện gần nhau nhiều lần.
2. *Move-to-front encoding :* đầu vào là một tệp văn bản trong đó chuỗi các chữ cái giồng nhau xuất hiện gần nhau nhiều lần, đầu ra là tệp văn bản trong đó các số nguyên nhỏ xuất hiện thường xuyên hơn các số lớn.
3. *Huffman encoding ;* đầu vào là một tệp văn bản trong đó các ký hiệu nhất định xuất hiện thường xuyên hơn những ký hiệu khác, thuật toán mã hóa các chữ cái phổ biến bằng các từ mã ngắn và các chữ cái hiếm với các từ mã dài.

1.2 Thuật toán Burrows-Wheeler transform

BWT là một thuật toán chuyển đổi dữ liệu, cấu trúc lại dữ liệu theo cách sao cho thông điệp được chuyển đổi dễ nén hơn. Về mặt kỹ thuật, nó là một hoán vị có thể đảo ngược từ vựng của các ký tự trong một chuỗi. Đây là bước đầu tiên trong 3 bước thực hiện liên tiếp của thuật toán Burrows-Wheeler Data Compression, tạo cơ sở cho tiện tích nén Unix bzip2.

Ý tưởng của BWT là xây dựng một mảng có các hang đề là sự thau đổi theo chu kỳ của chuỗi đầu vào theo thứ tự từ điển và trả về cột cuối cùng của mảng có xu hướng có các ký tự giống hệt nhau. Lợi ích của việc này là khi các ký tự đã được nhóm lại với nhau, chúng sẽ có một thứ tự hiệu quả, điều này có thể làm cho chuỗi của chúng ta dễ nén hơn đối với các thuật toán khác như run-length encoding và Huffman Coding.

Điều đáng chú ý về BWT là biến đổi cụ thể này có thể đảo ngược với chi phí dữ liệu tối thiểu.

2.2.1 Burrows-Wheeler transform

Chúng ta sẽ sử dụng chuỗi "compress$"

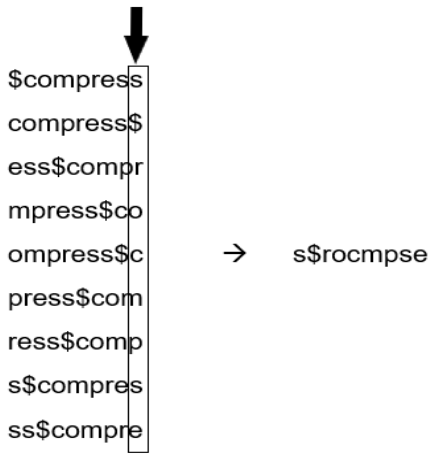
- Bước 1: Thực hiện hoán vị ký tự cuối chuỗi lên đầu chuỗi để thu được các chuỗi khác nhau thu được sau qua trình hoán vị.

		compress$
		$compress
		s$compres
		ss$compre
compress$	→	ess$compr
		ress$comp
		press$com
		mpress$co
		ompress$c

- Bước 2: Sắp xếp các chuỗi thu được từ bước 1 theo từ điển.

compress$		$compress
$compress		compress$
s$compres		ess$compr
ss$compre		mpress$co
ess$compr	→	ompress$c
ress$comp		press$com
press$com		ress$comp
mpress$co		s$compres
ompress$c		ss$compre

- Bước 3: Chuỗi tạo thành từ những ký tự cuối cùng của các chuỗi được sắp xếp ở bước 2 là chuỗi BWT ta cần thu được. Đối với ví dụ “compress$” thì chuỗi BWT thu được sẽ là “s$rocmpse”.



Tại sao trong ví dụ lại xuất hiện ký tự "$" ở cuối chuỗi?

Việc xuất hiện của "$" ở đây đúng là không có ý nghĩa trong việc biến đổi BWT, và nếu không có "$" thì quá trình biến đổi BWT của chúng ta vẫn diễn ra như các bước phía trên.

Tuy nhiên, việc xuất hiện của "$" không phải là vô ích, mà nó có ý nghĩa đối với việc ta thực hiện thuật toán biến đổi ngược BWT để tìm lại chuỗi ban đầu từ chuỗi BWT ta thu được phía trên, thuật toán biến đổi ngược này sẽ được trình bày ngay dưới đây.

2.3 Thuật toán Move-to-front encoding

Ý tưởng chính của thuật toán Move-to-front encoding là duy trì một chuỗi các ký hiệu pháp lý có thứ tự và được đọc lặp đi lặp lại các ký hiệu từ đầu vào, in ra vị trí mà ký hiệu đó xuất hiện và di chuyển ký hiệu đó lên đầu danh sách.

Thuật toán yêu cầu cung cấp 1 tập các ký tự có thứ tự ban đầu (thường lấy bảng chữ cái alphabet) làm "mốc", đối chiếu với chuỗi đầu vào để sinh ra output là tập các số nguyên. Sau đây, chúng ra sẽ lấy một ví dụ đơn giản để minh họa cho thuật toán này.

Ví dụ: với đầu vào là chuỗi BWT ở phần trên "s$rocmpse", ta chọn tập ký tự có thứ tự là *$ c e m o p r s*

Move-to-front	input	output
$ c e m o p r s	s	7
s $ c e m o p r	$	1
$ s c e m o p r	r	7
r $ s c e m o p	o	6
o r $ s c e m p	c	4
c o r $ s e m p	p	7
p c o r $ s e m	m	7
m p c o r $ s e	s	6
s m p c o r $ e	e	7
e s m p c o r $		

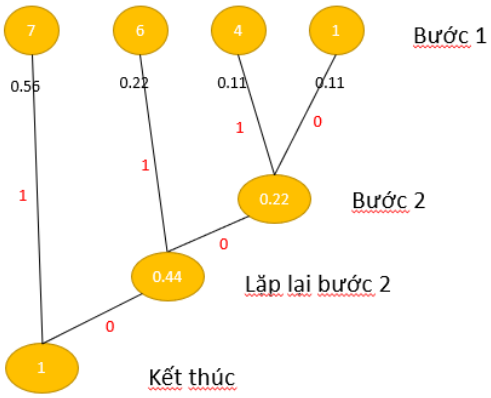
2.4 Thuật toán Huffman compression

Ý tưởng của thuật toán Huffman encoding là dựa vào bảng tần suất xuất hiện các ký tự cần mã hóa để xây đựng một bộ mã nhị phân cho các ký tự sao cho dung lượng (số bit) sau khi mã hóa là nhỏ nhất.

Ở đây ta lựa chọn giải thuật tham lam để xây dụng cây mã tiền tố tối ưu của Huffman, ở mỗi bước ta chọn 2 ký tự có tần suất thấp nhất để mã hóa bằng từ có mã dài nhất.

- Bước 1: Khởi tạo n rừng (n là số ký tự khác nhau xuất hiện trong đầu vào)
- Bước 2: Chọn 2 nốt có tần suất nhỏ nhất, lấy chúng làm 2 node con và tạo cây với root có trọng số bằng tổng tần suất 2 node con và cập nhật trạng thái danh sách các rừng.
- Bước 3: Lặp lại bước 2 cho đến khi chỉ còn 1 rừng.

Ví dụ: với tập đầu ra ở phần Move-to-front bên trên: *7 1 7 6 4 7 7 6 7* với tần suất của các ký tự là 7 - 0.56; 6 – 0.22; 4 – 0.11; 1 – 0.11. Lúc này quá trình xây dựng cây Huffman diễn ra như sau:



Kết quả, ta thu được bộ mã tối ưu tương ứng: ‘7’ – 1; ‘6’ – 01; ‘4’ – 001; ‘1’ – 000

Thực hiện nén file bằng mã Huffman

Trong các bước trên, giả sử đã xây dựng được bộ mã Huffman của 256 ký hiệu có mã ASCII từ 0 đến 255 chứa trong mảng Code[1..256]. Việc nén file có thể phân tích sơ bộ như sau:

1. Đọc từng byte của file cần nén cho đến khi hết tệp,
2. Chuyển theo bộ mã thành xâu nhị phân,
3. Ghép với xâu nhị phân còn dư từ bước trước,
4. Nếu có đủ 8 bit trong xâu thì cắt ra tám bít đầu tiên ghi vào tệp nén,
5. Nếu đã hết tệp cần nén thì dừng...

PHẦN 2: XÂY DỰNG CHƯƠNG TRÌNH THỬ NGHIỆM THUẬT TOÁN BURROWS-WHEELER DATA COMPRESSION

KẾT QUẢ VÀ ĐÁNH GIÁ

- Ngôn ngữ lập trình sử dụng: *Python*
- Bộ test lấy tại trang web: https://www.cs.princeton.edu/courses/archive/spring04/cos226/assignments/burrows.html
- Cấu hình laptop: Windows 10 Pro

Processor:	Intel(R) Core(TM) i7-4810MQ CPU @ 2.80GHz 2.80 GHz
Installed memory (RAM):	8.00 GB

Kiểm thử:

Kích thước file input	Thời gian nén (xấp xỉ)	Kích thước file output	Ghi chú
1018KB	8.5s	759KB	
2416KB	32s	1782KB	
3953KB	67s	2731KB	
4530KB	57s	2379KB	
6967KB	125s	3744KB	
9766KB	339s	6590KB	
28287KB			Chứa nhiều ký tự đặc biệt

Nhận xét:

- Chương trình chưa tối ưu, thời gian nén tương đối lâu.
- Chương trình chưa thể thực nén với những file chứ nhiều ký tự đặc biệt.
- Các thuật toán thành phần chưa được thực hiện theo cách tối ưu nhất hiện có.
- Nhìn chung, chương trình còn nhiều thiếu xót, song bản than em đã cố gắng nhiều để thu được kết quả kiểm thử như trên.

TÀI LIỆU THAM KHẢO

[1] Bài viết về Data Compressing tại trang web: https://searchstorage.techtarget.com/definition/compression#:~:text=Data%20compression%20is%20a%20reduction,storage%20hardware%20and%20network%20bandwidth.

[2] Bài viết về Data Compressing tại trang web: https://www.netmotionsoftware.com/blog/connectivity/how-does-data-compression-work#:~:text=Data%20compression%20is%20a%20process,with%20a%20process%20called%20decompression).

[3] Bài báo "Burrows Wheeler Compression: Principles and Reflections": https://www.cs.auckland.ac.nz/~peter-f/FTPfiles/BWreflections.pdf

[4] Bài viết về việc triển khai "Burrows-Wheeler Data Compressing Algorithm" trong bài báo "COS 226 Programming Assignment" : https://www.cs.princeton.edu/courses/archive/spring04/cos226/assignments/burrows.html

[5] Bài viết về thuật toán BWT tại GeeksforGeeks: https://www.geeksforgeeks.org/burrows-wheeler-data-transform-algorithm/

[6] Bài viết về thuật toán Huffman tại GeeksforGeeks: https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/

[7] Bài giảng thuật toán BWT trên youtube: https://www.youtube.com/watch?v=4n7NPk5lwbI&t=234s

[8] Bài viết về Mã háo Huffman tại Wikipedia: https://vi.wikipedia.org/wiki/M%C3%A3_h%C3%B3a_Huffman#N%C3%A9n_file_b%E1%BA%B1ng_m%C3%A3_Huffman

[9] Và nhiều nguồn tài liệu nhỏ khác