ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO THỬ THÁCH 1

MÔN HỌC:

CẤU TRÚC DỮ LIỆU
VÀ GIẢI THUẬT

GIÁO VIÊN HƯỚNG DẪN

GIÁO VIÊN LÝ THUYẾT:

VĂN CHÍ NAM

Thành phố Hồ Chí Minh 2022

MỤC LỤC

THÔNG TIN THÀNH VIÊN

MSSV	HỌ VÀ TÊN	EMAIIL
21120313	Trần Nam Phương	21120313@student.hcmus.edu.vn
21120317	Nguyễn Phan Anh Quốc	21120317@student.hcmus.edu.vn
21120321	Nguyễn Văn Siêu	21120321@student.hcmus.edu.vn

MỨC ĐỘ HOÀN THÀNH

STT	HỌ VÀ TÊN	CÔNG VIỆC ĐƯỢC PHÂN CÔNG	ĐÁNH GIÁ
1	Nguyễn Văn Siêu	Trả lời câu hỏi 1.1 Viết báo cáo	100%
2	Nguyễn Phan Anh Quốc	Viết code giải nén (Uncompress.h và Uncompress.cpp)	100%
3	Trần Nam Phương	Viết code nén(Compress.h, Compress.cpp, Header.h và Main.cpp)	100%

THUẬT TOÁN NÉN STATIC HUFFMAN CODING(SHC)

LỊCH SỬ:

Thuật toán được đề xuất bởi David A. Huffman khi ông còn là sinh viên Ph.D. tại MIT, và công bố năm 1952 trong bài báo "A Method for the Construction of Minimum-Redundancy Codes"

Để mã hóa các ký hiệu (ký tự, chữ số,...) ta thay chúng bằng các xâu nhị phân, được gọi là từ mã của ký hiệu đó. Chẳng hạn bộ mã ASCII, mã hóa cho 256 ký hiệu là biểu diễn nhị phân của các số từ 0 đến 255, mỗi từ mã gồm 8 bit. Trong ASCII từ mã của ký tự "a" là 1100001, của ký tự "A" là 1000001. Trong cách mã hóa này các từ mã của tất cả 256 ký hiệu có độ dài bằng nhau (mỗi từ mã 8 bit). Nó được gọi là mã hóa với độ dài không đổi.

Giải thích thuật toán:

Giả sử ta có một chuỗi ký tự: AAAABBBDDDDEEEEFFFFFFSSSSS

Pha nén:

Chiều dài là: 26 ký tự => có 26*8 = 208 bit

Bước 1:

Thống kế tần suất xuất hiện của các ký tự có trong chuỗi trên:

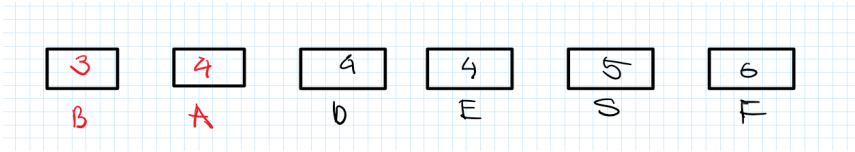
4A, 3B,4D,4E,6F,5S

Bước 2:

Sắp xếp dãy trên theo tăng dần lần xuất hiện

3B, 4A,4D,4E,5S,6F

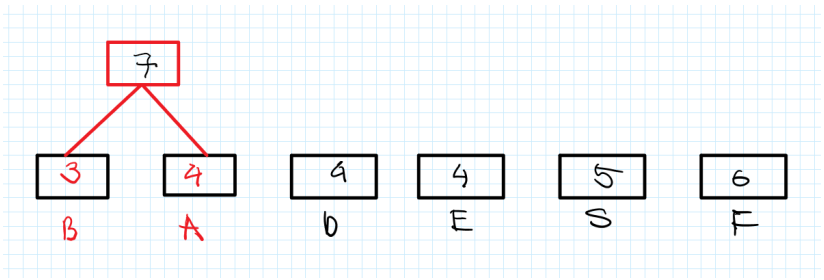
Và chuyển chúng thành các node lá trong cây nhị phân

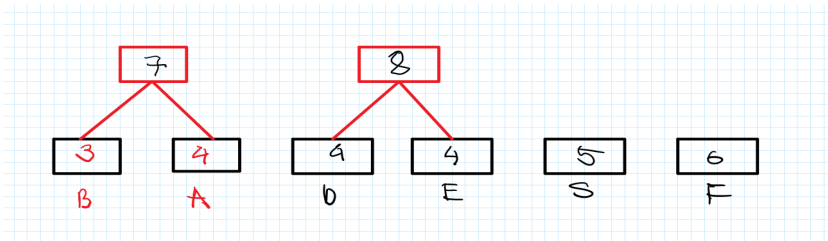


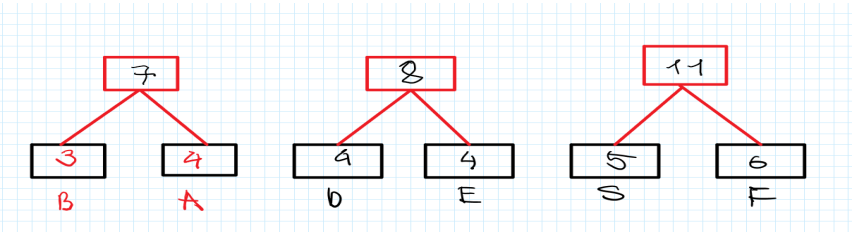
THUẬT TOÁN NÉN STATIC HUFFMAN CODING(SHC)

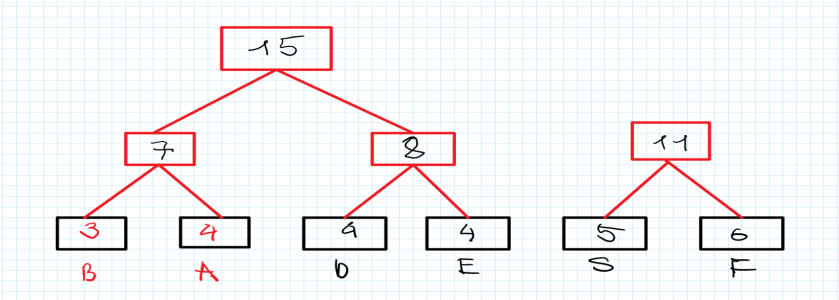
Bước 3:

Lần lượt tạo node cha từ 2 node con kề nhau sao cho tổng của 2 node con đó là nhỏ nhất

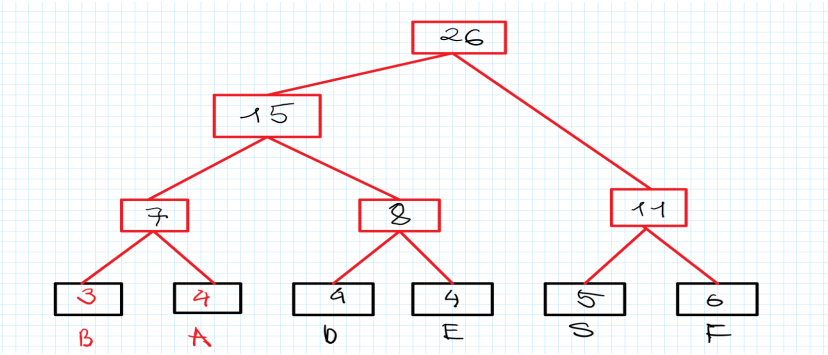






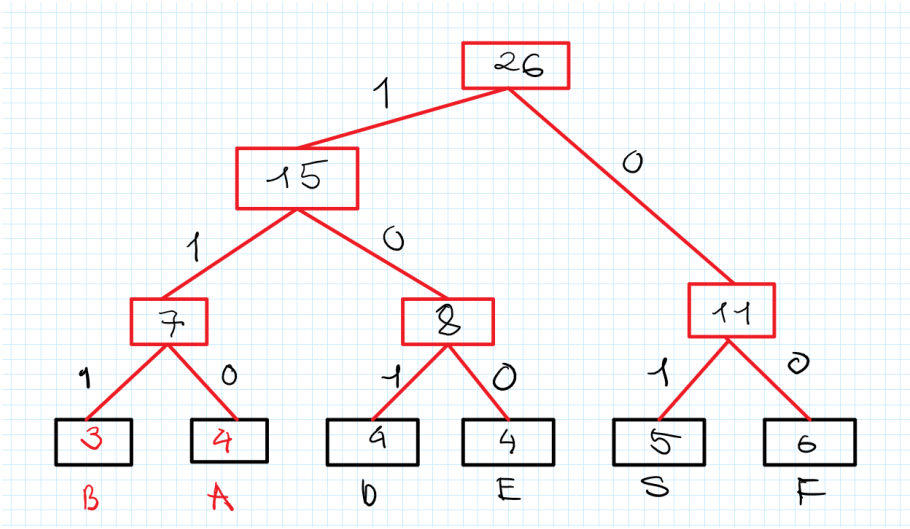


THUẬT TOÁN NÉN STATIC HUFFMAN CODING(SHC)



Bước 4:

Đánh số nhị phân 1 0 theo cạnh nối các node, cạnh trái là 1, cạnh phải là 0;



Bước 5:

Tìm đường đi ngắn nhất từ node gốc đến node lá chứa tần suất xuất hiện của các ký tự, đi qua cạnh nào thì ta lấy bit đó, cuối cùng, ta được bit mã hóa của các ký tự:

THUẬT TOÁN NÉN STATIC HUFFMAN CODING(SHC)

B	A	D	E	S	F
111	110	101	100	01	00
X3	X4	X4	X4	X5	X6
9 bit	12 bit	12 bit	12 bit	10 bit	10 bit

Tổng: 65 bit

Ban đầu: 208 bit

Suy ra tiết kiệm được: $\frac{208-65}{208} = 68,75\%$

Bước 6:

Viết lại chuỗi trên theo các bit: 110 110 110 110 - 111 111 111 – 101 101 101 101 – 100 100 100 100 – 00 00 00 00 00 00 – 01 01 01 01 01

(lưu ý : các dấu cách và "–" chỉ thêm vào để giúp dễ nhìn không bao gồm trong output thực tế)

Bước 7:

Xuất ra quy ước giải nén:

B	A	D	E	S	F
111	110	101	100	01	00
X3	X4	X4	X4	X5	X6

Quy ước này gồm:

6 ký tự: 8*6 = 48 bit;

Bit nén: 3 + 3 + 3 + 3 + 2 + 2 = 16 bit

Tổng: 48 + 16 = 64 bit

Tổng cộng: 64 + 65 = 129 < 208 bit ban đầu

Tiết kiệm: $\frac{208-129}{208} = 37,98\%$

THUẬT TOÁN NÉN STATIC HUFFMAN CODING(SHC)

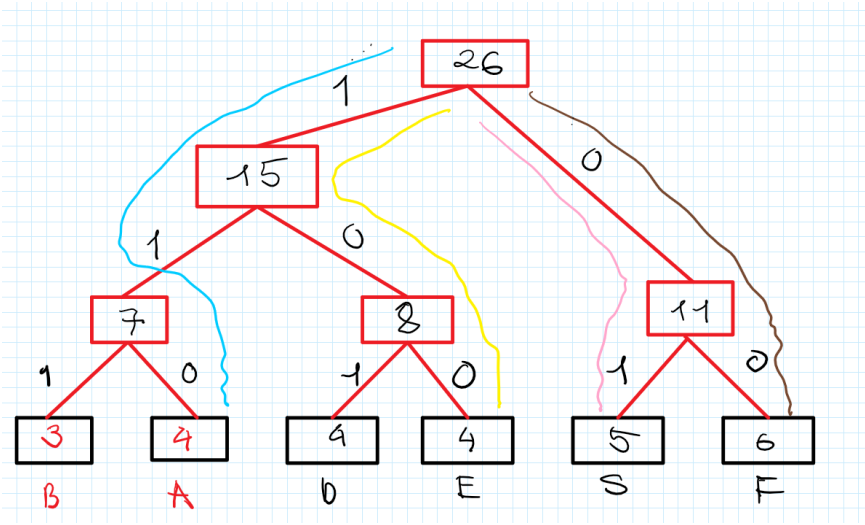
Pha giải nén:

Dựa vào quy ước giải nén của pha nén:

(Vẫn dựa vào cây nhị phân ở trên để giải nén)

Giả sử ta có đoạn bit nén:

110 01 01 100 00



Xét từ trái qua phải:

Bít đầu là 1, ta sẽ đi từ node gốc sang trái, tìm đường đi sao cho bit trả về trùng với bit nén, ta được ký tự đầu tiên

Xét tương tự lần lượt cho các bit sau đó ta được:

ASSEF

THUẬT TOÁN NÉN LZW

LỊCH SỬ:

LZW là một phương pháp nén được phát minh bởi Lempel - Ziv và Welch. Nó hoạt động dựa trên một ý tưởng rất đơn giản là 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.

Nguyên tắc hoạt động của nó như sau:

- Một xâu ký tự là một tập hợp từ hai ký tự trở lên.

- Nhớ tất cả các xâu ký tự đã gặp và gán cho nó một dấu hiệu (token) riêng.

- Nếu lần sau gặp lại xâu ký tự đó, xâu ký tự sẽ được thay thế bằng dấu hiệu của nó.

Phần quan trọng nhất của phương pháp nén này là phải tạo một mảng rất lớn dùng để lưu giữ các xâu ký tự đã gặp (Mảng này được gọi là "Từ điển"). Khi các byte dữ liệu cần nén được đem đến, chúng liền được giữ lại trong một bộ đệm chứa (Accumulator) và đem so sánh với các chuỗi đã có trong "từ điển". Nếu chuỗi dữ liệu trong bộ đệm chứa không có trong "từ điển" thì nó được bổ sung thêm vào "từ điển" và chỉ số của chuỗi ở trong "từ điển" chính là dấu hiệu của chuỗi. Nếu chuỗi trong bộ đệm chứa đã có trong "từ điển" thì dấu hiệu của chuỗi được đem ra thay cho chuỗi ở dòng dữ liệu ra. Có bốn quy tắc để thực hiện việc nén dữ liệu theo thuật toán LZW là:

Quy tắc 1: 256 dấu hiệu đầu tiên được dành cho các ký tự đơn (0 - 0ffh).

Quy tắc 2: Cố gắng so sánh với "từ điển" khi trong bộ đệm chứa đã có nhiều hơn hai ký tự.

Quy tắc 3: Các ký tự ở đầu vào (Nhận từ tập tin sẽ được nén) được bổ sung vào bộ đệm chứa đến khi chuỗi ký tự trong bộ đệm chứa không có trong "từ điển".

Quy tắc 4: Khi bộ đệm chứa có một chuỗi mà trong "từ điển" không có thì chuỗi trong bộ đệm chứa được đem vào "từ điển". Ký tự cuối cùng của chuỗi ký tự trong bộ đệm chứa phải ở lại trong bộ đệm chứa để tiếp tục tạo thành chuỗi mới.

Giải thích thuật toán:

Ví dụ: Ta có file text: BABAABAAA

Dấu # là một dấu hiệu được sử dụng để kết thúc đoạn tin. Do đó, ta có 26 ký tự văn bản gốc (A – Z)

THUẬT TOÁN NÉN LZW

Và ký tựu # đại diện cho ký tự dừng. Ta quy ước 1 – 26 cho các chữ cái và 0 cho #.

Máy tính cần hiển thị chúng dưới dạng chuỗi bit. Ở trường hợp này 5bit là đủ để dùng cho bộ 27 giá trị. Khi từ điển phát triển, các mã phải phát triển theo chiều rộng. 5 bit chỉ có thể cung cấp 2^5 = 32 tổ hợp, do đó khi từ thứ 33 được tạo, thuật toán phải ngay lập tức chuyển đổi từ 5 bit sang 6 bit (đối với tất cả giá trị đầu ra trước đó không bị ảnh hưởng). Chú ý rằng, do mã toàn không được sử dụng (00000), và nó được đánh dấu là 0, cho nên từ thứ 33 được đánh dấu là 32, và các mã trước không bị thay đổi, tuy nhiên nếu đã sử dụng đến bit thứ 6 thì phải chỉnh lại độ rộng, điều đó làm thay đổi các mã tiếp theo đó.

Symbol Binary Decimal

#	00000	0
A	00001	1
B	00010	2
C	00011	3
D	00100	4
E	00101	5
F	00110	6
G	00111	7
H	01000	8
I	01001	9
J	01010	10
K	01011	11
L	01100	12
M	01101	13
N	01110	14
O	01111	15
P	10000	16
Q	10001	17
R	10010	18
S	10011	19
T	10100	20
U	10101	21

THUẬT TOÁN NÉN LZW

V	10110	22
W	10111	23
X	11000	24
Y	11001	25
Z	11010	26

Pha nén:

Nhập vào bộ nhớ tạm thời các ký tự vào chuỗi Ω cho tới khi Ω + ký tự tiếp theo không tồn tại trong từ điển. Xuất mã cho Ω, và nhập Ω + ký tự tiếp theo vào từ điển. Bắt đầu nhập tạm thời lần nữa đối với các ký tự tiếp theo.

THUẬT TOÁN NÉN LZW

Current Sequence	Next Char	Output		Extended Dictionary	Comments
		Code	Bits		
NULL	T				
T	O	20	10100	27: TO	27 = first available code after 0 through 26
O	B	15	01111	28: OB	
B	E	2	00010	29: BE	
E	O	5	00101	30: EO	
O	R	15	01111	31: OR	
R	N	18	10010	32: RN	32 requires 6 bits, so for next output use 6 bits
N	O	14	001110	33: NO	
O	T	15	001111	34: OT	
T	T	20	010100	35: TT	
TO	B	27	011011	36: TOB	
BE	O	29	011101	37: BEO	
OR	T	31	011111	38: ORT	
TOB	E	36	100100	39: TOBE	
EO	R	30	011110	40: EOR	
RN	O	32	100000	41: RNO	
OT	#	34	100010		# stops the algorithm; send the cur seq
		0	000000		and the stop code

Độ dài khi chưa nén: 25 ký tự * 5 bit = 125 bit

Độ dài mã hóa: (6 mã * 5 bit) + (11 mã * 6 bit) = 96 bit

Sử dụng LZW tiết kiệm được 23% bit. Nếu đoạn tin dài hơn thì các từ sẽ đại diện trong đoạn văn bản dài hơn.

Pha giải nén:

THUẬT TOÁN NÉN LZW

Input		Output Sequence	New Dictionary Entry		Comments
Bits	Code		Full	Conjecture	
10100	20	T		27: T?	
01111	15	O	27: TO	28: O?	
00010	2	B	28: OB	29: B?	
00101	5	E	29: BE	30: E?	
01111	15	O	30: EO	31: O?	
10010	18	R	31: OR	32: R?	created code 31 (last to fit in 5 bits)
001110	14	N	32: RN	33: N?	so start reading input at 6 bits
001111	15	O	33: NO	34: O?	
010100	20	T	34: OT	35: T?	
011011	27	TO	35: TT	36: TO?	
011101	29	BE	36: TOB	37: BE?	36 = TO + 1st symbol (B) of
011111	31	OR	37: BEO	38: OR?	next coded sequence received (BE)
100100	36	TOB	38: ORT	39: TOB?	
011110	30	EO	39: TOBE	40: EO?	
100000	32	RN	40: EOR	41: RN?	
100010	34	OT	41: RNO	42: OT?	
000000	0	#			

SO SÁNH

LZW à thuật toán dựa trên tự điển, nó nén bằng việc thay thế các chuỗi ký tựu con mà đã xuất hiện trước đó với tham chiếu trong từ điển

Do đó nó sẽ có các điểm mạnh sau:

LZW không yêu cầu thông tin trước về luồng dữ liệu đầu vào (không yêu cầu bảng decode).

LZW có thể nén luồng đầu vào trong một lần chạy.

Một ưu điểm khác của LZW là tính đơn giản, cho phép thực hiện nhanh chóng.

Điểm yếu của LWZ là:

Nếu nó chưa lặp lại lần nào, LWZ sẽ không nén tốt (37,98 % > 23 %)

Ngoài ra do cần lập từ điển nên nó cần sử dụng bộ nhớ đệm với dung lượng rất lớn

Ngược lại, Huffman Coding vẫn có thể nén dữ liệu một cách đáng kể nếu một số ký hiệu (ký tự hoặc byte) xuất hiện thường xuyên hơn các ký hiệu khác.

Ưu điểm của mã hóa Huffman :

nó sử dụng ít không gian lưu trữ hơn(so với LWZ) cho một ký tự xuất hiện thường xuyên đồng thời sử dụng nhiều không gian lưu trữ hơn(so với LWZ) cho các ký tự ít xuất hiện. Điều này cho phép ta lưu trữ các ký tự xuất hiện thường xuyên hơn với chi phí thấp.

Nhược điểm:

Nó cần tham chiếu cho giải nén

Áp dụng thực tế:

LZW:

Nó được sử dụng trong file GIF và có thể trong PDF và TIFF. Lệnh nén Unix, và trong các ứng dụng khác

SHC:

Chúng được sử dụng để truyền fax và văn bản.

Chúng được sử dụng bởi các định dạng nén thông thường như PKZIP, GZIP, v.v.

Các codec đa phương tiện như JPEG, PNG và MP3 sử dụng mã hóa Huffman

CÁC BIẾN THỂ

LZW

LZMW (1985, bởi V. Miller, M. Wegman) – Tìm kiếm đầu vào cho chuỗi dài nhất đã có trong từ điển (khớp "hiện tại"); thêm phần nối của từ khớp trước với từ khớp hiện tại vào từ điển. (Các mục từ điển do đó phát triển nhanh hơn; nhưng sơ đồ này phức tạp hơn nhiều để thực hiện.) Miller và Wegman cũng đề xuất xóa các mục nhập tần số thấp khỏi từ điển khi từ điển đầy.

LZAP (1988, bởi James Storer) – sửa đổi của LZMW: thay vì chỉ thêm phần nối của từ khớp trước với từ khớp hiện tại vào từ điển, hãy thêm từ nối của từ khớp trước với mỗi chuỗi con ban đầu của từ khớp hiện tại ("AP" là viết tắt của "tất cả các tiền tố"). Ví dụ: nếu kết quả khớp trước đó là "wiki" và kết quả khớp hiện tại là "pedia", thì bộ mã hóa LZAP sẽ thêm 5 chuỗi mới vào từ điển: "wikip", "wikipe", "wikiped", "wikipedi" và "wikipedia ", trong đó bộ mã hóa LZMW chỉ thêm một chuỗi "wikipedia". Điều này giúp loại bỏ một số sự phức tạp của LZMW, với cái giá phải trả là thêm nhiều mục từ điển hơn.

LZWL là một biến thể dựa trên âm tiết của LZW..

STATIC HUFFMAN CODING

n-ary Huffman coding

Thuật toán Huffman n-ary sử dụng bảng chữ cái {0, 1,..., n − 1} để mã hóa thông điệp và xây dựng cây n-ary

Adaptive Huffman coding

Tính toán xác suất một cách linh hoạt dựa trên các tần số thực tế gần đây trong chuỗi ký hiệu nguồn và thay đổi cấu trúc cây mã hóa để phù hợp với các ước tính xác suất được cập nhật.

Huffman template algorithm

Thông thường, các trọng số được sử dụng trong việc triển khai mã hóa Huffman đại diện cho các xác suất số, nhưng thuật toán đưa ra ở trên không yêu cầu điều này; nó chỉ yêu cầu các trọng số tạo thành một monoid giao hoán có thứ tự hoàn toàn, nghĩa là một cách để sắp xếp thứ tự các trọng số và cộng chúng lại.

TÀI LIỆU THAM KHẢO

Mã hóa Huffman – Wikipedia tiếng Việt

Huffman coding - Wikipedia

Lempel–Ziv–Welch - Wikipedia

LZW (Lempel–Ziv–Welch) Compression technique - GeeksforGeeks

(79) Huffman Encoding & Decoding - YouTube

(79) LZW Compression Algorithm Explained | An introduction to data compression - YouTube