Nén tĩnh (Static Huffman)

## **Mã hóa Huffman** ([David A. Huffman](http://vi.wikipedia.org/w/index.php?title=David_A._Huffman&amp;action=edit&amp;redlink=1))là một thuật toán mã hóa dùng để nén dữ liệu.

Dựa trên bảng tần suất xuất hiện các kí tự cần mã hóa để xây dựng một bộ mã nhị phân cho các kí tự đó sao cho dung lượng (số bit) sau khi mã hóa là nhỏ nhất.

Ý tưởng:

* Giảm số bit để biểu diễn một ký tự thay cho chuỗi 8 bit trong bảng mã ASCII.
* Dùng chuỗi bit ngắn hơn để biểu diễn ký tự xuất hiện với tần suất nhiều.

|  |  |
| --- | --- |
| **Ký tự Mã huffman** | |
| A | 000 |
| B | 1 |
| C | 01 |
| D | 011 |
| E | 100 |

|  |  |
| --- | --- |
| **Ký tự Tần suất** | |
| A | 9 |
| B | 15 |
| C | 10 |
| D | 6 |
| E | 7 |

|  |  |
| --- | --- |
| **Ký tự Mã bit** | |
| A | 01000001 |
| B | 01000010 |
| C | 01000011 |
| D | 01000100 |
| E | 01000101 |

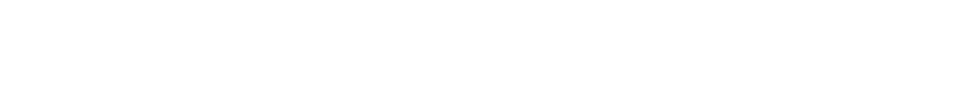
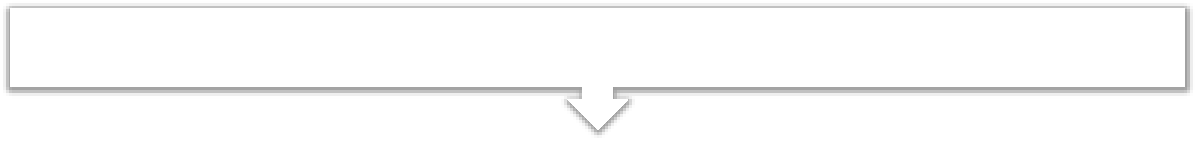
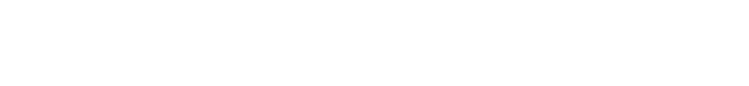
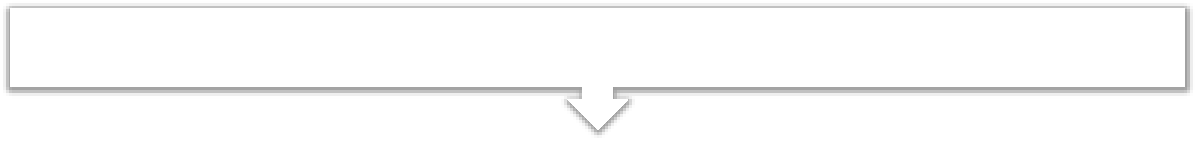
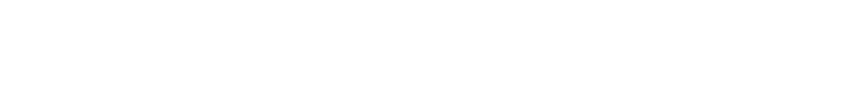
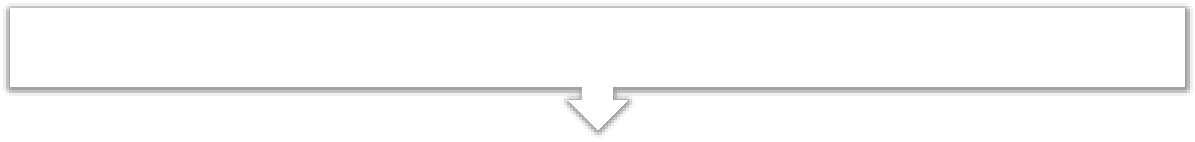
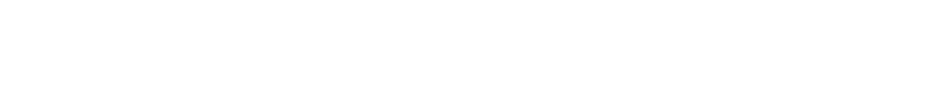
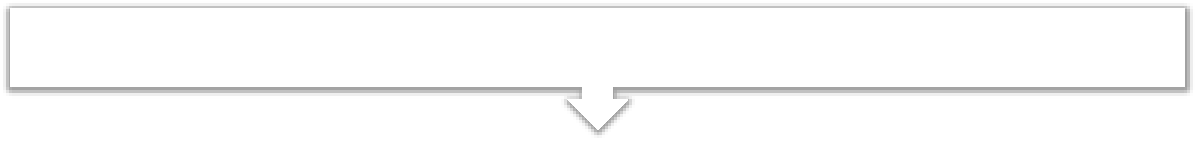
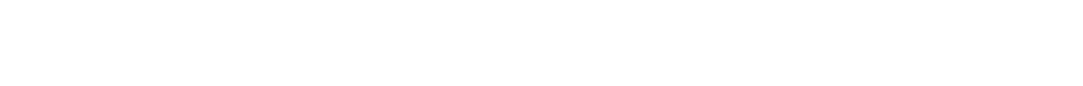
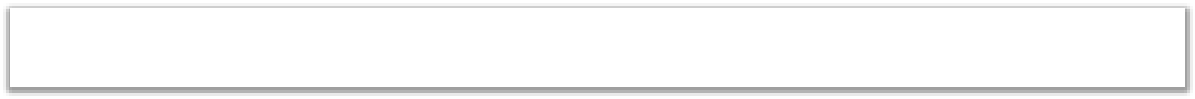
Cây Huffman

* Là cây nhị phân, mỗi nút chứa ký tự và trọng số (tần suất của ký tự đó).
* Mỗi ký tự được biểu diễn bằng 1 nút lá (tính tiền tố).
* Nút cha có tổng ký tự, tổng trọng số của 2 nút con.
* Các nút có trọng số, ký tự tăng dần từ trái sang phải.
* Các nút có trọng số lớn nằm gần nút gốc.
* Các nút có trọng số nhỏ nằm xa nút gốc hơn.

Mã Huffman

* Là chuỗi nhị phân được sinh ra dựa trên cây Huffman.
* Mã Huffman của ký tự đó là đường dẫn từ nút gốc đến nút lá chứa ký tự đó.
* Sang trái ta được bit 0
* Sang phải ta được bit 1
* Có độ dài biến đổi
* Các ký tự có tần suất ít hơn có mã dài hơn
* Các ký tự có tần suất lớn hơn có mã ngắn hơn

Thuật toán nén tĩnh



**B1**: Duyệt file, lập bảng thống kê tuần suất xuất hiện của mỗi ký tự.

**B2**: Xây dựng cây Huffman dựa vào bảng thống kê.

**B3**: Sinh mã Huffman cho mỗi ký tự dựa vào cây Huffman.

**B4**: Duyệt file, thay toàn bộ ký tự bằng mã Huffman tương ứng.

**B5**: Lưu lại cây Huffman (bảng mã) dùng cho việc giải nén. Xuất file đã nén.



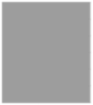
B5

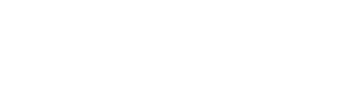
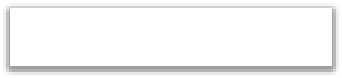
B4

B3

B2

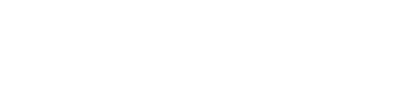
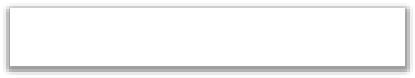
B1





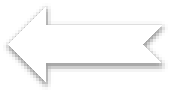
Chuỗi ký tự cần nén

F = “ABABBCBBDEEEABABBAEEDDCCABBBCDEEDCBCCCCDBBBCAAA”

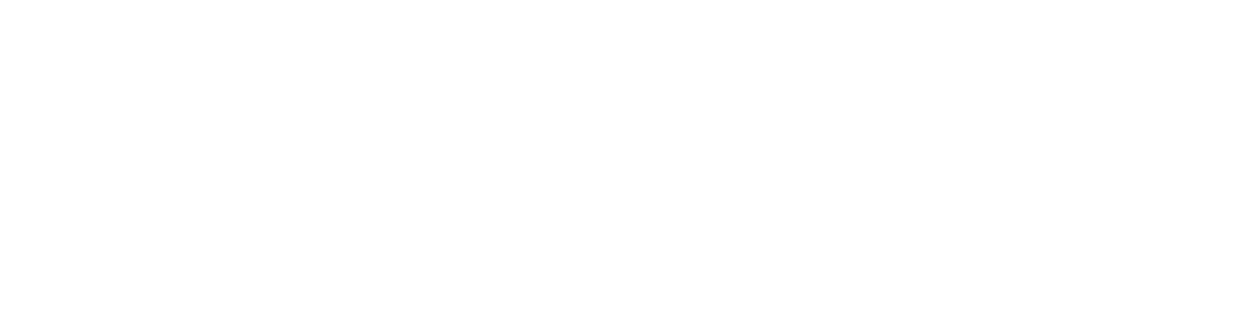
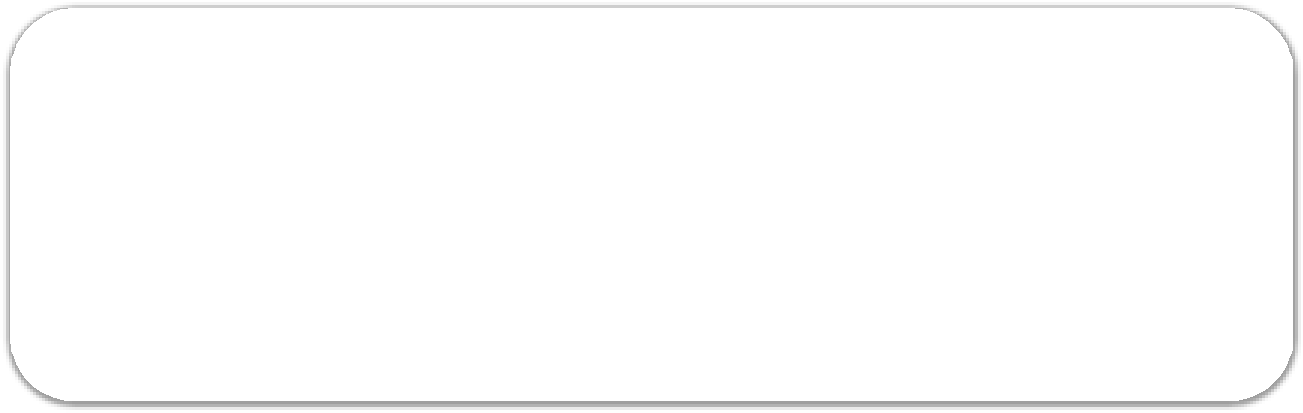
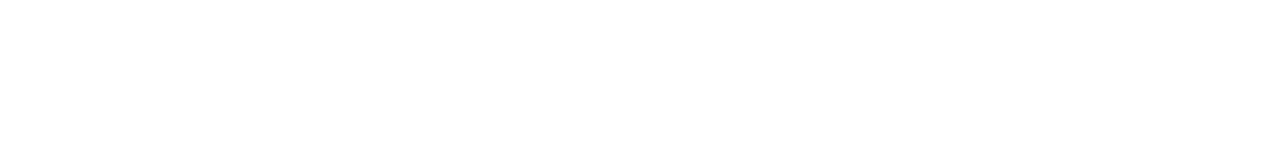
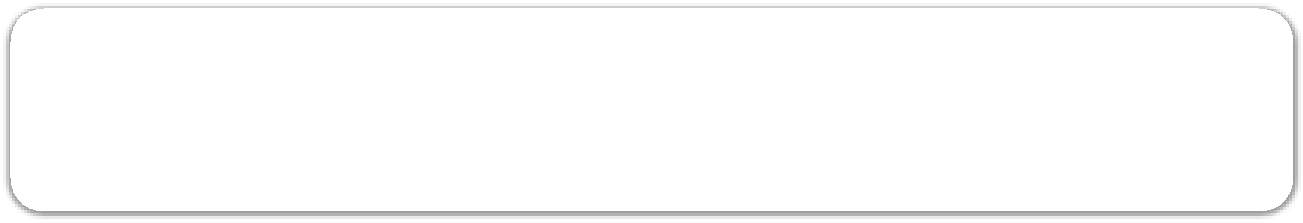
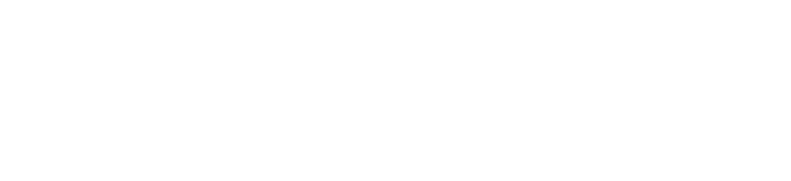
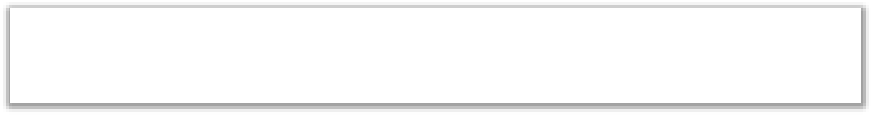
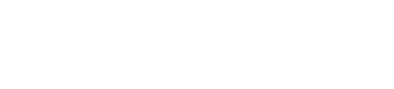
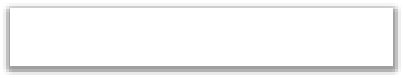


**Bảng tần suất xuất hiện**

|  |  |
| --- | --- |
| **Ký tự Tần suất** | |
| A | 9 |
| B | 15 |
| C | 10 |
| D | 6 |
| E | 7 |



Xây dựng cây Huffman



Thuật toán tham lam

B1: Tạo N cây, mỗi cây chỉ có một nút gốc, mỗi nút gốc chỉ

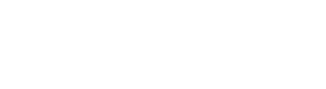
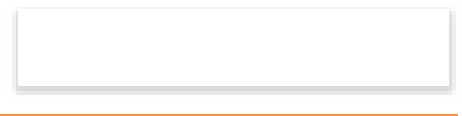
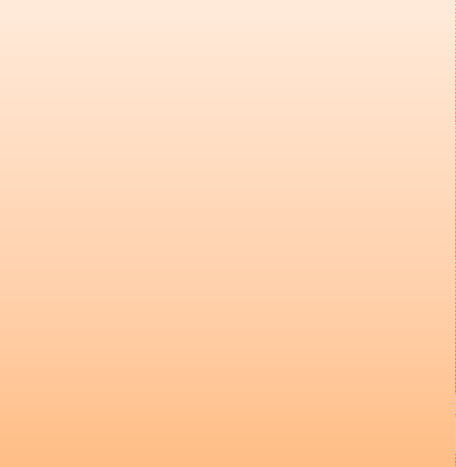
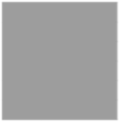
chứa một kí tự và trọng số (tần suất của ký tự đó). (N = số ký tự)

B2: Lặp lại thao tác sau cho đến khi chỉ còn 1 cây duy nhất:

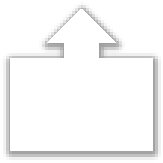
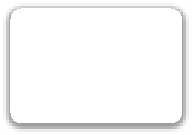
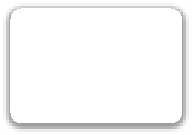
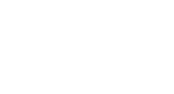
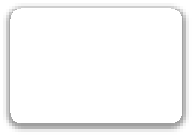
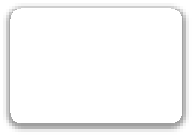
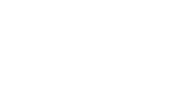
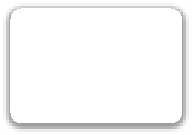
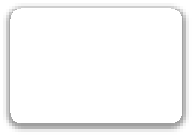
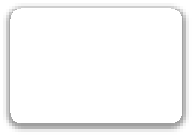
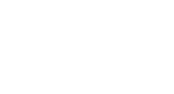
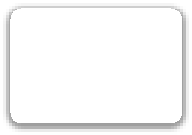
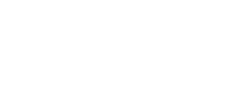
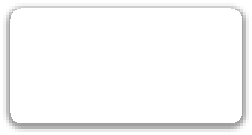
+ Ghép 2 cây con có trọng số gốc nhỏ nhất thành 1 nút cha, có tổng ký tự, tổng trọng số trọng số của 2 nút con.

+ Xóa các cây đã duyệt.

+ Điều chỉnh lại cây nếu vi phạm tính chất.



**Bảng mã Huffman**



ABCDE | 47

0

1

AC | 19

BDE | 28

0

1

0

1

A |9

C | 10

DE | 13

B | 15

0

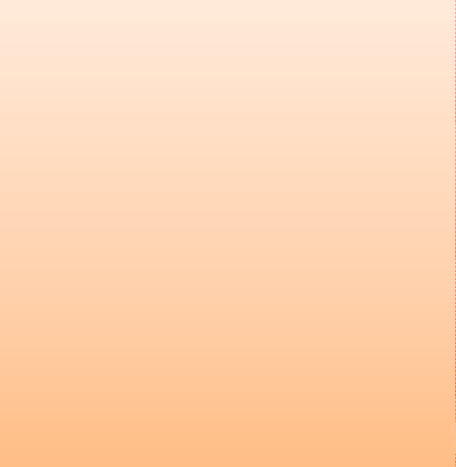
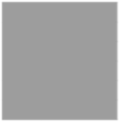
1

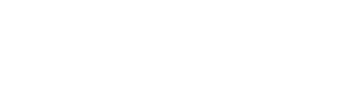
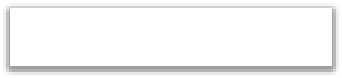
**00**

D | 6

E | 7

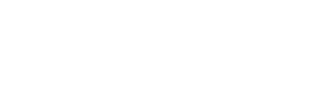
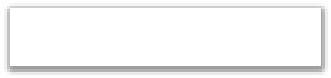
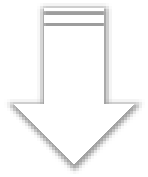
|  |  |
| --- | --- |
| **Ký tự Mã Huffman** | |
| A | 00 |
| B | 11 |
| C | 01 |
| D | 100 |
| E | 101 |

F = “ABABBCBBDEEEABABBAEEDDCCABBBCDEEDCBCCCCDBBBCAAA”



Chuỗi ký tự cần nén

|  |  |
| --- | --- |
| **Ký tự Mã Huffman** | |
| A | 00 |
| B | 11 |
| C | 01 |
| D | 100 |
| E | 101 |



Chuỗi đã được nén

FOutput = “001100111101111110010110110100110011110010110110010001

01001111110110010110110001110101010110011111101000000”

Tiết kiệm: 8\*47 - (2\*9 + 2\*15 + 2\*10 + 3\*6 + 3\*7) = 376 - 107 = 269 bit Tỷ lệ nén: (1 - 107/376)\*100 = 72.54 %

Ưu – Nhược điểm

Ưu điểm:

* + - * + Hệ số nén tương đối cao
        + Phương pháp thực hiện tương đối đơn giản.
        + Đòi hỏi ít bộ nhớ.

Nhược điểm:

* + - * + Mất 2 lần duyệt file khi nén.
        + Phải lưu trữ thông tin giải mã vào file nén.
        + Phải xây dựng lại cây Huffman khi giải nén.

Nén động (Adaptive Huffman)

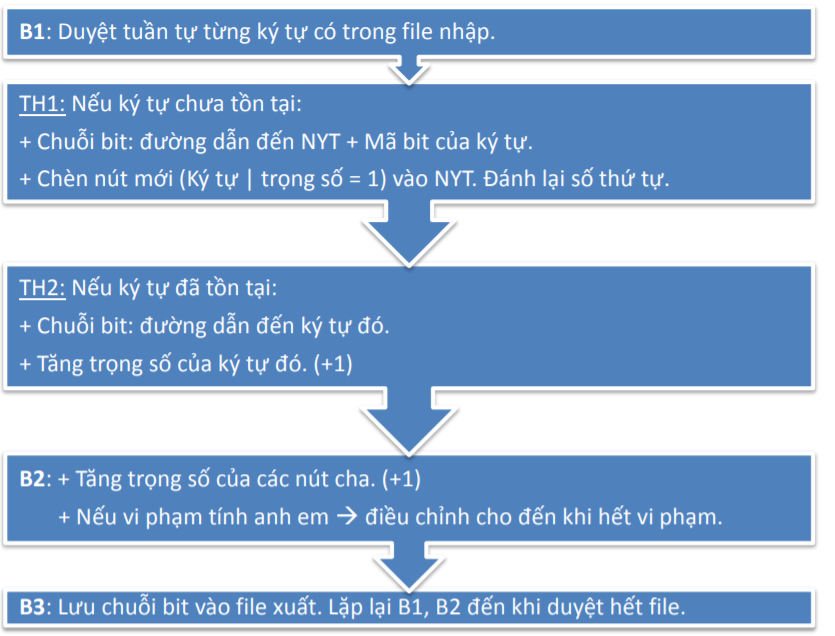
* Khắc phục được nhược điểm của static Huffman.
* Vừa duyệt, vừa cập nhật cây Huffman và xuất kết quả theo thời gian thực

Cây Huffman

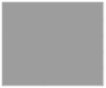
* + - Tính chất anh em:
      * Trọng số nút bên trái bé hơn trọng số nút bên phải, bé hơn nút cha
    - Nút NYT (not yet transmitted)
      * Luôn có trọng số = 0.

.

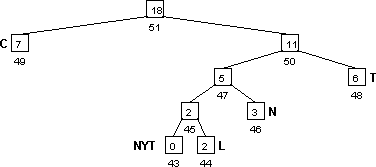
Thuật toán nén động

+ Chuỗi bit: đường dẫn đ ến N YT + Mã bit của ký tự.

* Thuật toán điều chỉnh:
  + - * Nếu trọng số nút hiện hành > nút lân cận từ phải sang trái, từ dưới lên trên -> Vi phạm
      * Tìm nút xa nhất có trọng số cao nhất < trọng số nút vi phạm -> Hoán đổi vị trí.

**Ví dụ**: F = “CNTTCLCCNTTCLCCNTT”

|  |  |
| --- | --- |
| **Ký tự Mã bit** | |
| C | 01000011 |
| L | 01001100 |
| N | 01001110 |
| T | 01010100 |



**FOutput = “C 0N 00T 101 101 100L 11 0 101 10 10 0 1001 0 0 101 10 10”**