# Giải thuật nén Huffman

- Nén tĩnh (Static Huffman)
- Nén động (Adaptive Huffman)

# Nén tĩnh (Static Huffman)

# Giới thiệu

- **Mã hóa Huffman** (David A. Huffman)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.

Trong bản mã ASCII, mỗi ký tự được biểu diễn bằng chuỗi 8 bit.

Ý tưởng

- Giảm số bit để biểu diễn 1 ký tự
- Dùng chuỗi bit ngắn hơn để biểu diễn ký tự xuất hiện nhiều
- Sử dụng mã tiền tố để phân cách các ký tự

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

| Ký tự | Mã bit |
|---|---|
| A | 000 |
| B | 001 |
| C | 010 |
| D | 011 |
| E | 100 |

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

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

| Ký tự | Mã tiền tố |
|---|---|
| A | 00 |
| B | 11 |
| C | 01 |
| D | 100 |
| E | 101 |

# 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 là đường dẫn từ nút gốc đến nút lá đó.

- Sang trái ta được bit 0
- Sang phải ta được bit 1

Có độ dài biến đổi (tối ưu bảng mã).

- Các ký tự có tần suất lớn có độ dài ngắn.
- Các ký tự có tần suất nhỏ có độ dài dài hơn.

## Thuật toán nén tĩnh (Static Huffman)

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

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

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

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

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

## Chuỗi ký tự cần nén

F = “ABABBCBBDEEEABABBAEEDDCCABBBCDEEDCBCCCCDBBBCAAA”

N = 47

## Bảng tần suất xuất hiện

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

# 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:

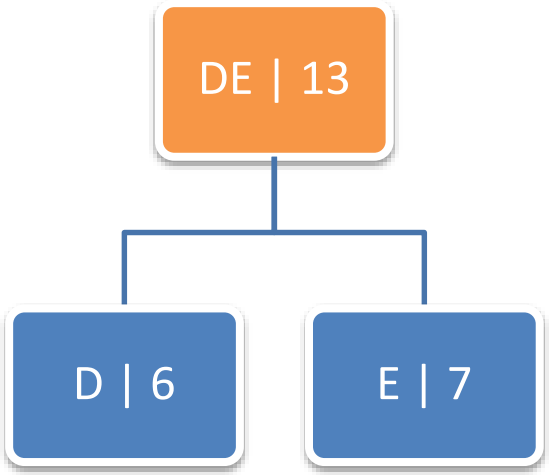
+ Ghép 2 cây con có trọng số gốc nhỏ nhất thành 1 nút cha, có tổng ký tự, tổng trọng số trọng số của 2 nút con.

+ Xóa các cây đã duyệt.

+ Điều chỉnh lại cây nếu vi phạm tính chất.

# Xây dựng cây Huffman

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

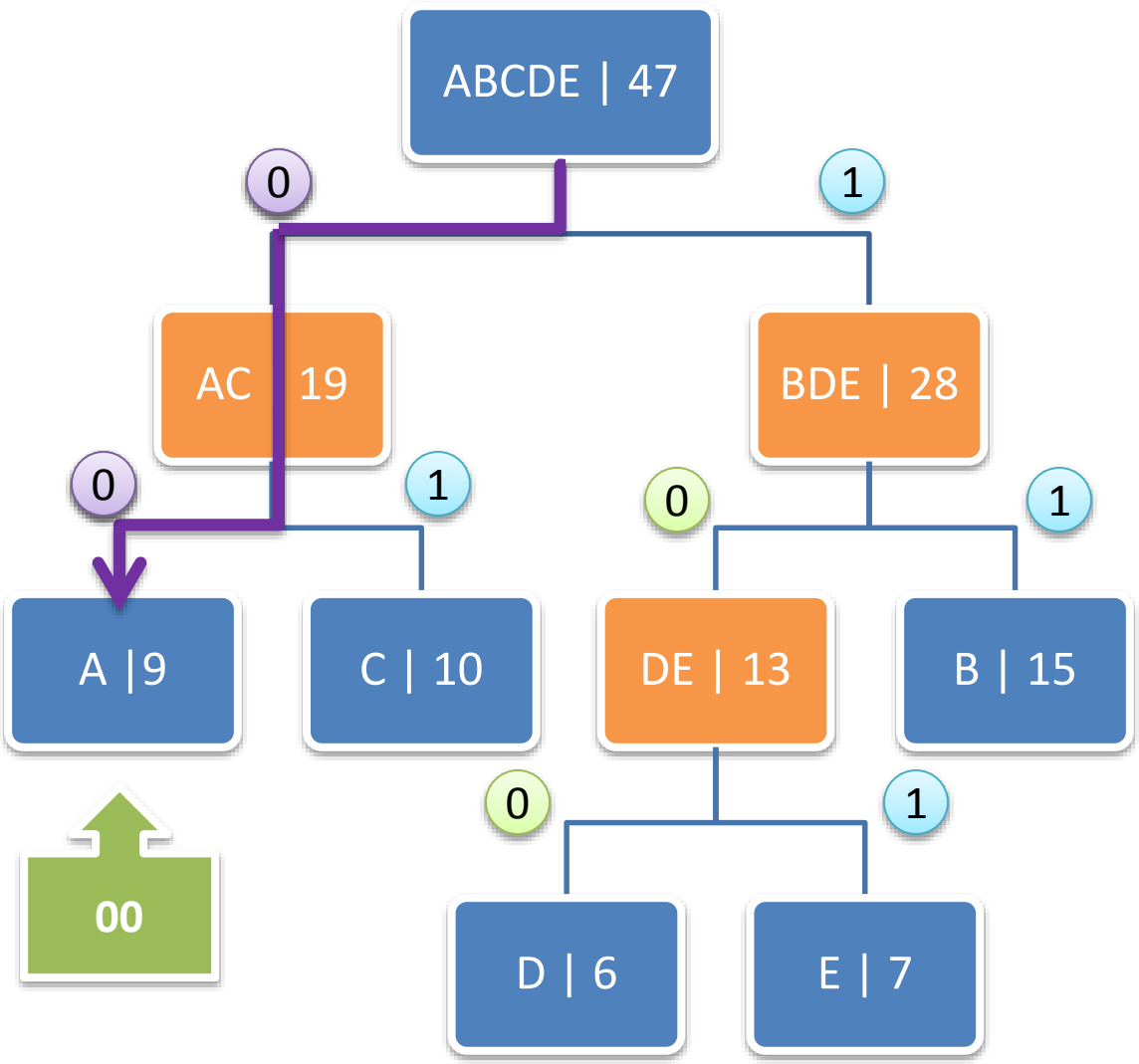



| Ký tự | Tần suất |
|---|---|
| B | 15 |
| DE | 13 |
| **C** | **10** |
| **A** | **9** |

| Ký tự | Tần suất |
|---|---|
| AC | 19 |
| **B** | **15** |
| **DE** | **13** |

| Ký tự | Tần suất |
|---|---|
| BDE | 28 |
| **AC** | **19** |

| Ký tự | Tần suất |
|---|---|
| ABCDE | 47 |




**Bảng mã Huffman**

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

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

Chuỗi ký tự cần nén

F = “ABABBCBBDEEEABABBAEEDDCCABBBCDEEDCBCCCCDBBBCAAA”

Chuỗi đã được nén

$F_{Output}$ =
“00110011110111111001011011010011001111001011011001000101001111110110010110110001110101010110011111101000000”

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 %

## Thuật toán giải nén

**B1**: Xây dựng lại cây Huffman từ thông tin giải mã đã lưu.

**B2:** Duyệt file, đọc lần lượt từng bit trong file nén và duyệt cây.

**B3**: Xuất ký tự tương ứng khi duyệt hết nút lá.

**B4**: Thực hiện B2, B3 cho đến khi duyệt hết file.

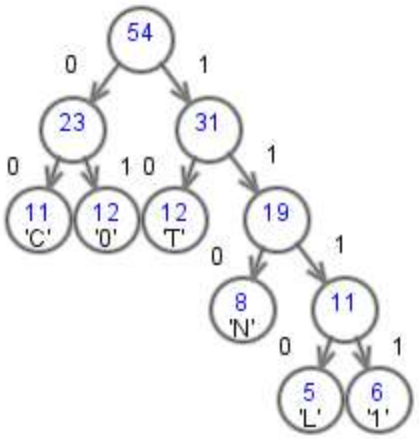
**B5**: Xuất file đã giải nén.

## Bài tập: Nén chuỗi sau bằng giải thuật nén tĩnh – Static Huffman

F = “CNTT10110CLCCNNTTT10000CCCCLLLLCCCTTTT11000NTNNN000TNT”

N = 54

| Ký tự | Tần suất |
|---|---|
| 0 | 12 |
| 1 | 6 |
| C | 11 |
| L | 5 |
| N | 8 |
| T | 12 |




| Ký tự | Mã Huffman |
|---|---|
| 0 | 01 |
| 1 | 1111 |
| C | 00 |
| L | 1110 |
| N | 110 |
| T | 10 |

Kết quả

$F_{Output}$ = “00110101011110111111111010011100000110110101010111101010101000000001110111011101110000001010101011111111010101110101101101100101011011010”

# 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)

**Ưu điểm**

- Khắc phục nhược điểm của Static Huffman.
- Đầu đọc vừa duyệt, vừa cập nhật cây Huffman, vừa xuất kết quả ra file nén theo thời gian thực.
- (Ngược lại).

# Cây Huffman

Tính chất anh em:

Trọng số của nút bên trái phải nhỏ hơn nút bên phải, nhỏ hơn nút cha

Nút NYT (not yet transmitted) có trọng số luôn = 0, dùng để nhận biết ký tự đã xuất hiện trong cây hay chưa.

Trọng số nút cha bằng tổng trọng số 2 nút con.

# Thuật toán nén động (Adaptive Huffman)

**B1**: Duyệt tuần tự từng ký tự có trong file nhập.

TH1: Nếu ký tự chưa tồn tại:

+ Chuỗi bit: đường dẫn đến NYT + Mã bit của ký tự.

+ Chèn nút mới (Ký tự | trọng số = 1) vào NYT. Đánh lại số thứ tự.

TH2: Nếu ký tự đã tồn tại:

+ Chuỗi bit: đường dẫn đến ký tự đó.

+ Tăng trọng số của ký tự đó. (+1)

**B2**: + Tăng trọng số của các nút cha. (+1)

+ Nếu vi phạm tính anh em → điều chỉnh cho đến khi hết vi phạm.

**B3**: Lưu chuỗi bit vào file xuất. Lặp lại B1, B2 đến khi duyệt hết file.

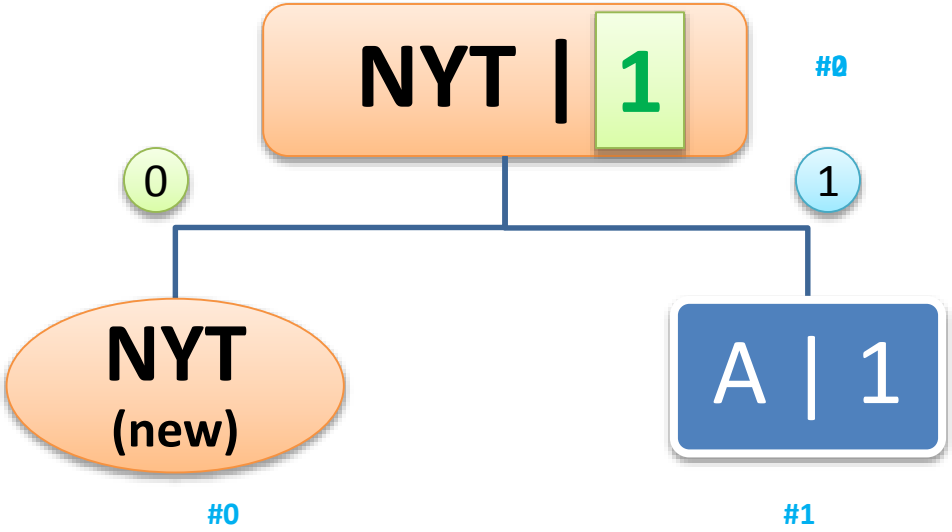
# Thuật toán điều chỉnh

+ 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.

TH1: Ký tự chưa tồn tại

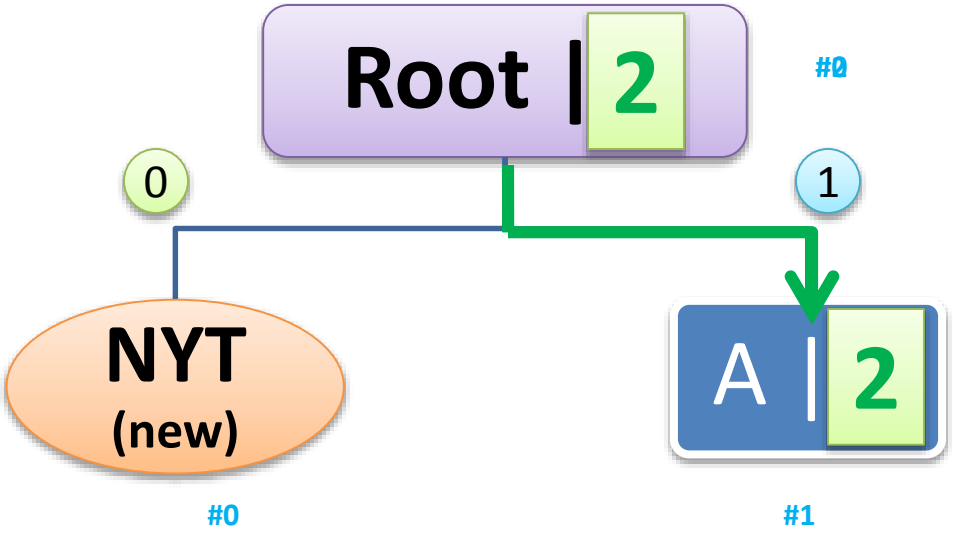
F = "AABBB"




$F_{Output}$ = 01000011

TH1: Ký tự đã tồn tại

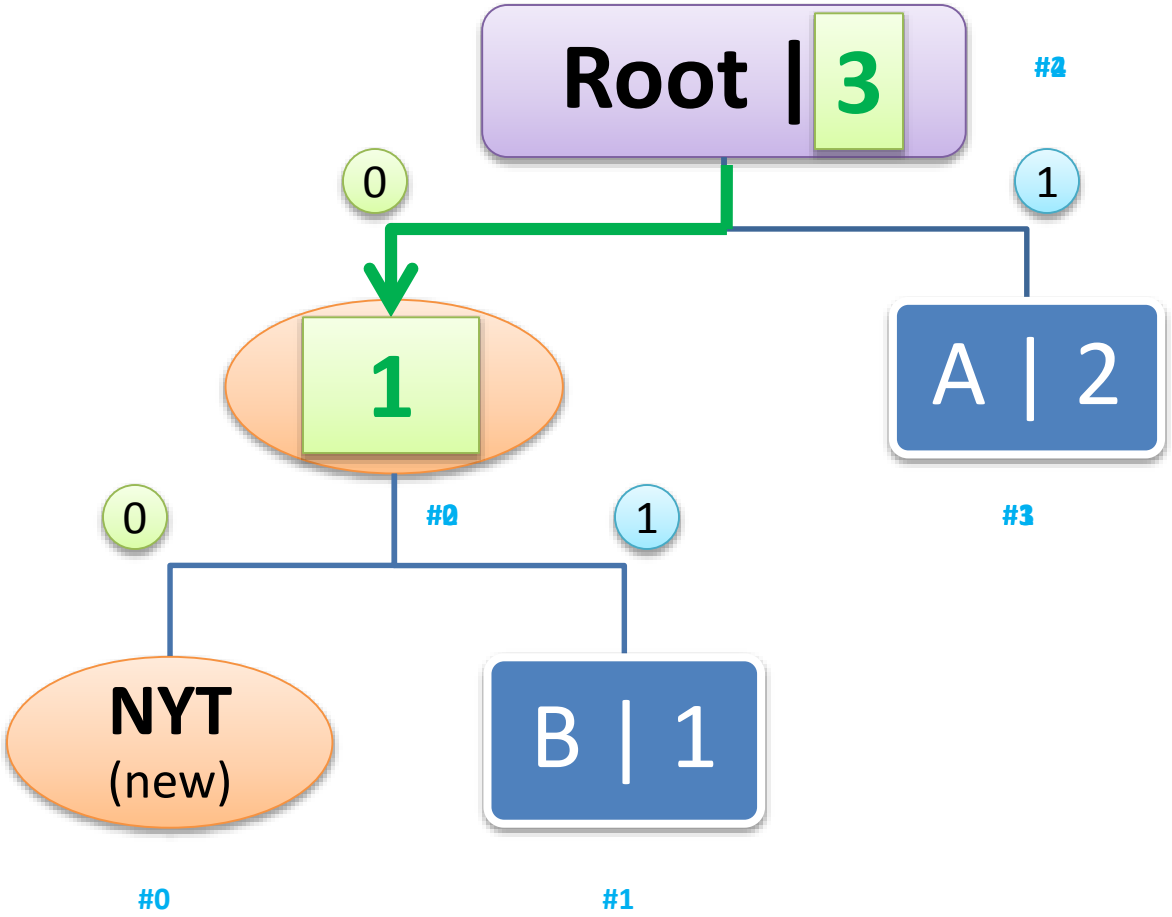
F = "AABBB"




$F_{Output}$ = 01000011 1

TH1: Ký tự chưa tồn tại

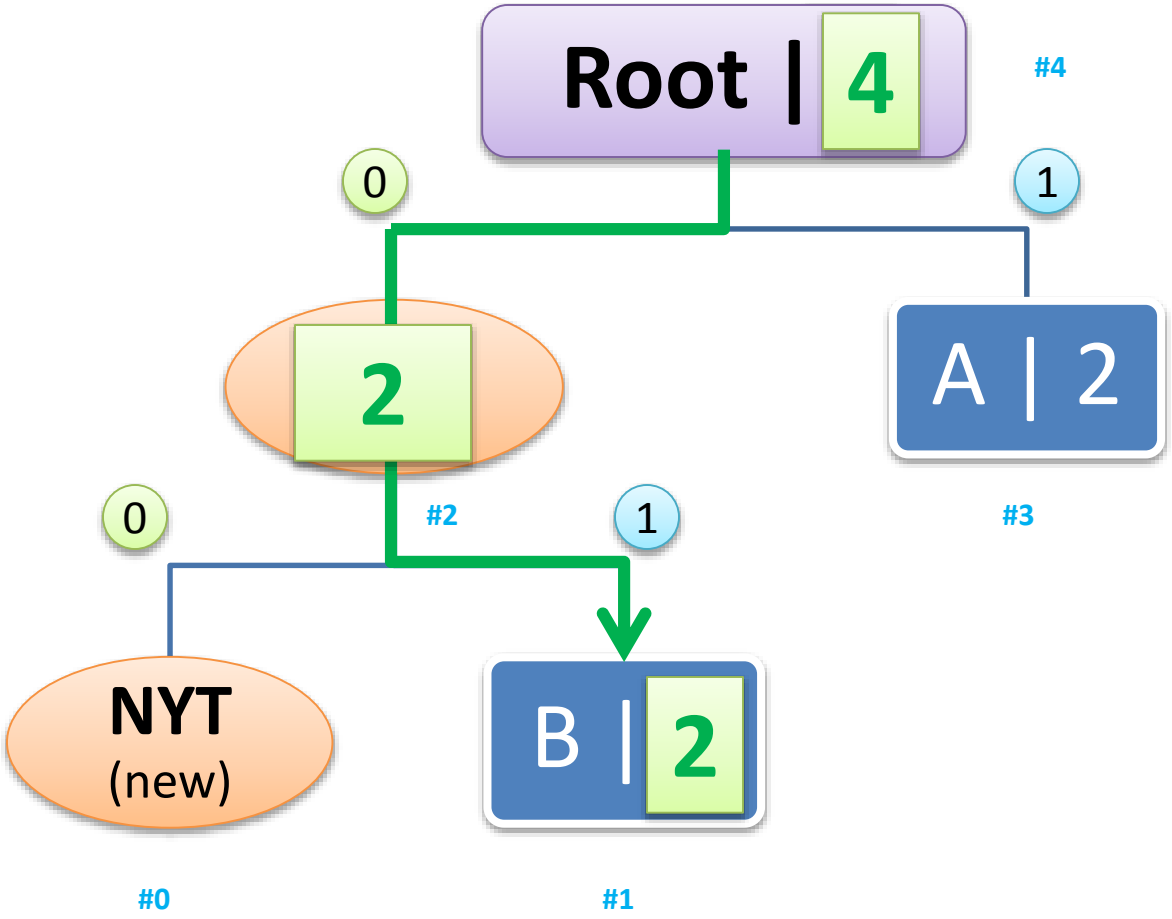
F = "AABBB"




$F_{Output}$ = 0100001110010000 10

TH2: Ký tự đã tồn tại

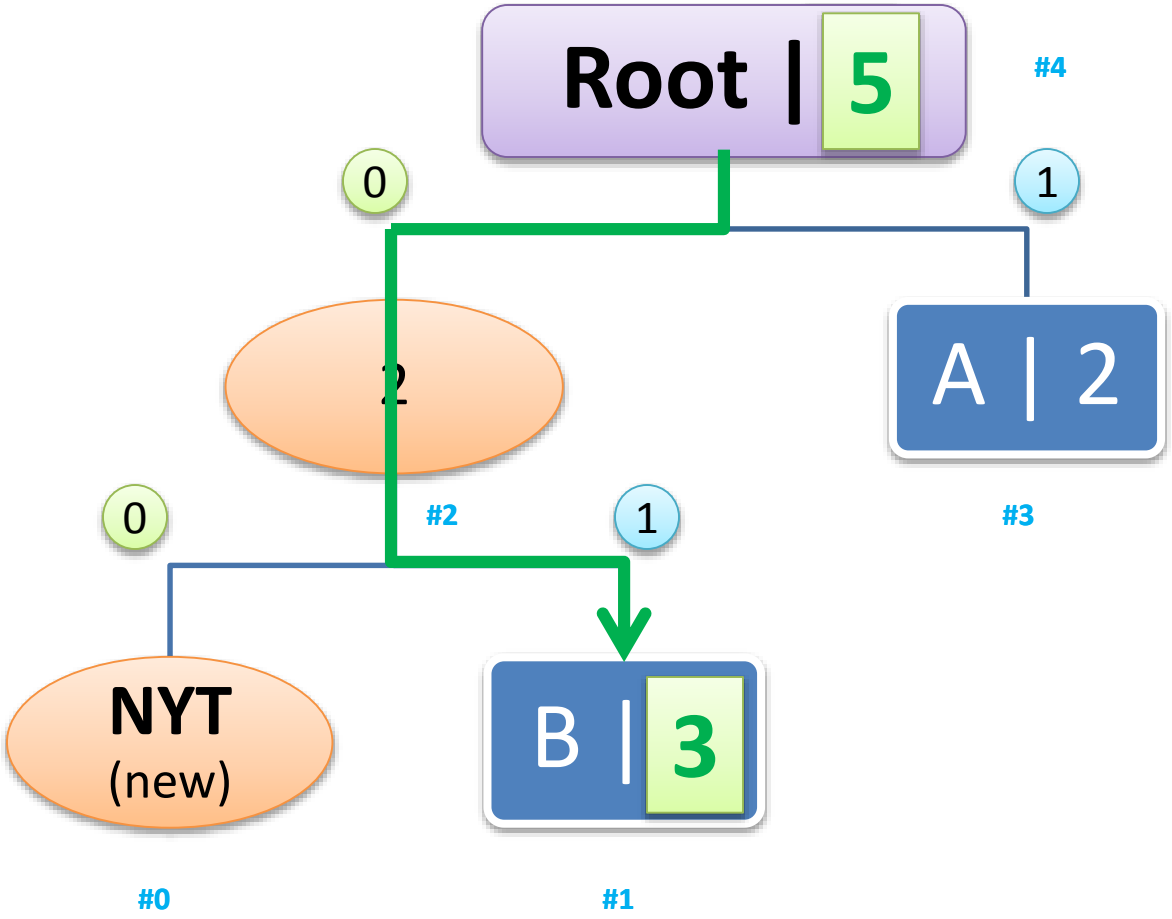
F = "AABBB"




$F_{Output}$ = 01000011**10**01000010 **01**

TH2: Ký tự đã tồn tại (vi phạm)

F = "AABBB"




$F_{Output}$ = 0100001110010000100101

## Thuật toán giải nén

**B1:** Duyệt file, đọc lần lượt từng bit trong file nén và duyệt cây.

**B2**: Xuất ký tự tương ứng khi duyệt hết nút lá.

**B3:** Nếu gặp nút NYT, đọc 8 bit tiếp theo. Xuất ký tự tương ứng. Cập nhật ký tự vừa xuất vào cây.

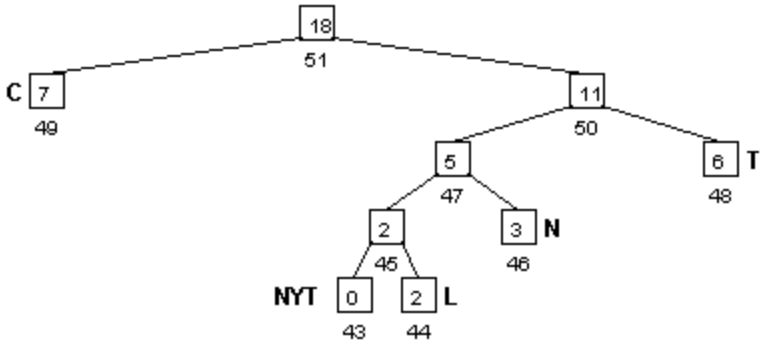
**B4**: Thực hiện B1, B2, B3 cho đến khi duyệt hết file.

**B5**: Xuất file đã giải nén.

## Bài tập: Nén chuỗi sau bằng giải thuật nén động – Adaptive Huffman

F = “CNTTCLCCNTTCLCCNTT”

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




**$F_{Output}$ = “C 0N 00T 101 101 100L 11 0 101 10 10 0 1001 0 0 101 10 10”**

# Thanks for your listening !

# Thực hiện

Nguyễn Văn Hòa

Cái Ngọc Tịnh Tiến