CÁC THUẬT TOÁN NÉN DỮ LIỆU

Bùi Tiến Lên

01/01/2017

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Giới thiệu

Mục đích của nén dữ liệu:

- Giảm kích thước dữ liệu
- *Tăng tính bảo mật*

cuu duong than cong . com

Giới thiệu (cont.)

Có hai dạng thuật nén

- Nén bảo toàn thông tin (*lossless compression*)
  - Thuật toán nén RLE
  - Thuật toán nén LZW
  - Thuật toán nén Huffman
- Nén không bảo toàn thông tin (*lossy compression*)
  - Thuật toán nén sử dụng biến đổi DFT
  - Thuật toán nén sử dụng biến đổi wavelet

Giới thiệu (cont.)

Định nghĩa 1

Hiệu suất nén: tỉ lệ kích thước giảm được sau khi áp dụng thuật toán nén

$$D = \frac{N - M}{N} 100 \quad (1)$$

- D: hiệu suất nén
- N: kích thước dữ liệu trước khi nén
- M: kích thước dữ liệu sau khi nén

Hiệu suất nén tùy thuộc vào:

- Phương pháp nén
- Đặc trưng của dữ liệu

Thuật toán nén RLE

- Thuật toán nén *Run Length Encoding* (RLE) mã hóa dữ liệu dựa trên sự lặp lại
- Một dãy các **ký tự** lặp lại liên tiếp được gọi là **đường chạy** (*run*)
- **Đường chạy** sẽ được nén bằng công thức sau

  `[số ký tự][ký tự]`
- Khi độ dài đường chạy lớn thì tỉ lệ nén sẽ tăng lên

Thuật toán nén RLE (cont.)

Ví dụ 1

Hãy nén chuỗi sau bằng RLE

`AAABBCCAAADE`

Sẽ được mã hóa thành

`3A2B2C3A1D1E`

Đánh giá thuật toán RLE

- Đơn giản, dễ cài đặt
- Dùng để nén các dữ liệu có nhiều đoạn lặp lại
- Thích hợp cho dữ liệu ảnh
- Hiệu suất nén không cao

Thuật toán nén LZW

Giới thiệu

- Được đề xuất bởi Ziv and Lempel và cải tiến bởi Welch [Lempel, 1978]
- Đây là một thuật toán nén dựa trên tần suất xuất hiện trong từ điển. Do đó nó còn được gọi là thuật toán nén từ điển
- Ảnh định dạng GIF sử dụng thuật toán nén này

Thuật toán nén dữ liệu

```
w ← null
while ( ĐỌC ký tự k )
    if ( wk có trong TỪ ĐIỂN )
        w = wk;
    else
        XUẤT mã c ← Code(w)
        THÊM wk vào TỪ ĐIỂN
        w ← k
XUẤT mã c ← Code(w)
```

Minh họa thuật toán nén dữ liệu

Xét một chuỗi ký tự “abracadabarabra” →

k	$w = \emptyset$	**output**
a	a	
b	b	0
r	r	1
a	a	4
c	c	0
a	a	2
d	d	0
a	a	3
b	ab	
a	a	5
r	r	0
a	ra	
b	b	7
r	br	
a	a	6
		0

c	**word**
0	a
1	b
2	c
3	d
4	r
5	ab
6	br
7	ra
8	ac
9	ca
10	ad
11	da
12	aba
13	ar
14	rab
15	bra

Minh họa thuật toán nén dữ liệu

Xét một chuỗi ký tự “abracadabarabra” $\rightarrow$

k	$w = \emptyset$	**output**
a	a	
b	b	0
r	r	1
a	a	4
c	c	0
a	a	2
d	d	0
a	a	3
b	ab	
a	a	5
r	r	0
a	ra	
b	b	7
r	br	
a	a	6
		0

c	**word**
0	a
1	b
2	c
3	d
4	r
5	ab
6	br
7	ra
8	ac
9	ca
10	ad
11	da
12	aba
13	ar
14	rab
15	bra

Minh họa thuật toán nén dữ liệu

Xét một chuỗi ký tự "abracadabarabra" $\rightarrow$ 0

k	$w = \emptyset$	**output**
a	a	
b	b	0
r	r	1
a	a	4
c	c	0
a	a	2
d	d	0
a	a	3
b	ab	
a	a	5
r	r	0
a	ra	
b	b	7
r	br	
a	a	6
		0

c	**word**
0	a
1	b
2	c
3	d
4	r
5	ab
6	br
7	ra
8	ac
9	ca
10	ad
11	da
12	aba
13	ar
14	rab
15	bra

Minh họa thuật toán nén dữ liệu

Xét một chuỗi ký tự "abracadabarabra" $\rightarrow$ 0 1

k	$w = \emptyset$	**output**
a	a	
b	b	0
r	r	1
a	a	4
c	c	0
a	a	2
d	d	0
a	a	3
b	ab	
a	a	5
r	r	0
a	ra	
b	b	7
r	br	
a	a	6
		0

c	**word**
0	a
1	b
2	c
3	d
4	r
5	ab
6	br
7	ra
8	ac
9	ca
10	ad
11	da
12	aba
13	ar
14	rab
15	bra

Minh họa thuật toán nén dữ liệu

Xét một chuỗi ký tự "abracadabarabra" $\rightarrow$ 0 1 4

k	$w = \emptyset$	**output**
a	a	
b	b	0
r	r	1
a	a	4
c	c	0
a	a	2
d	d	0
a	a	3
b	ab	
a	a	5
r	r	0
a	ra	
b	b	7
r	br	
a	a	6
		0

c	**word**
0	a
1	b
2	c
3	d
4	r
5	ab
6	br
7	ra
8	ac
9	ca
10	ad
11	da
12	aba
13	ar
14	rab
15	bra

Minh họa thuật toán nén dữ liệu

Xét một chuỗi ký tự "abracadabarabra" $\rightarrow$ 0 1 4 0

k	$w = \emptyset$	**output**
a	a	
b	b	0
r	r	1
a	a	4
c	c	0
a	a	2
d	d	0
a	a	3
b	ab	
a	a	5
r	r	0
a	ra	
b	b	7
r	br	
a	a	6
		0

c	**word**
0	a
1	b
2	c
3	d
4	r
5	ab
6	br
7	ra
8	ac
9	ca
10	ad
11	da
12	aba
13	ar
14	rab
15	bra

Minh họa thuật toán nén dữ liệu

Xét một chuỗi ký tự "abracadabarabra" $\rightarrow$ 0 1 4 0 2

k	$w = \emptyset$	**output**
a	a	
b	b	0
r	r	1
a	a	4
c	c	0
a	a	2
d	d	0
a	a	3
b	ab	
a	a	5
r	r	0
a	ra	
b	b	7
r	br	
a	a	6
		0

c	**word**
0	a
1	b
2	c
3	d
4	r
5	ab
6	br
7	ra
8	ac
9	ca
10	ad
11	da
12	aba
13	ar
14	rab
15	bra

Minh họa thuật toán nén dữ liệu

Xét một chuỗi ký tự "abracadabarabra" $\rightarrow$ 0 1 4 0 2 0

k	$w = \emptyset$	**output**
a	a	
b	b	0
r	r	1
a	a	4
c	c	0
a	a	2
d	d	0
a	a	3
b	ab	
a	a	5
r	r	0
a	ra	
b	b	7
r	br	
a	a	6
		0

c	**word**
0	a
1	b
2	c
3	d
4	r
5	ab
6	br
7	ra
8	ac
9	ca
10	ad
11	da
12	aba
13	ar
14	rab
15	bra

Minh họa thuật toán nén dữ liệu

Xét một chuỗi ký tự "abracadabarabra" $\rightarrow$ 0 1 4 0 2 0 3

k	$w = \emptyset$	**output**
a	a	
b	b	0
r	r	1
a	a	4
c	c	0
a	a	2
d	d	0
a	a	3
b	ab	
a	a	5
r	r	0
a	ra	
b	b	7
r	br	
a	a	6
		0

c	**word**
0	a
1	b
2	c
3	d
4	r
5	ab
6	br
7	ra
8	ac
9	ca
10	ad
11	da
12	aba
13	ar
14	rab
15	bra

Minh họa thuật toán nén dữ liệu

Xét một chuỗi ký tự "abracadabarabra" $\rightarrow$ 0 1 4 0 2 0 3

k	$w = \emptyset$	**output**
a	a	
b	b	0
r	r	1
a	a	4
c	c	0
a	a	2
d	d	0
a	a	3
b	ab	
a	a	5
r	r	0
a	ra	
b	b	7
r	br	
a	a	6
		0

c	**word**
0	a
1	b
2	c
3	d
4	r
5	ab
6	br
7	ra
8	ac
9	ca
10	ad
11	da
12	aba
13	ar
14	rab
15	bra

Minh họa thuật toán nén dữ liệu

Xét một chuỗi ký tự "abracadabarabra" $\rightarrow$ 0 1 4 0 2 0 3 5

k	$w = \emptyset$	**output**
a	a	
b	b	0
r	r	1
a	a	4
c	c	0
a	a	2
d	d	0
a	a	3
b	ab	
a	a	5
r	r	0
a	ra	
b	b	7
r	br	
a	a	6
		0

c	**word**
0	a
1	b
2	c
3	d
4	r
5	ab
6	br
7	ra
8	ac
9	ca
10	ad
11	da
12	aba
13	ar
14	rab
15	bra

Minh họa thuật toán nén dữ liệu

Xét một chuỗi ký tự "abracadabarabra" $\rightarrow$ 0 1 4 0 2 0 3 5 0

k	$w = \emptyset$	**output**
a	a	
b	b	0
r	r	1
a	a	4
c	c	0
a	a	2
d	d	0
a	a	3
b	ab	
a	a	5
r	r	0
a	ra	
b	b	7
r	br	
a	a	6
		0

c	**word**
0	a
1	b
2	c
3	d
4	r
5	ab
6	br
7	ra
8	ac
9	ca
10	ad
11	da
12	aba
13	ar
14	rab
15	bra

Minh họa thuật toán nén dữ liệu

Xét một chuỗi ký tự "abracadabarabra" $\rightarrow$ 0 1 4 0 2 0 3 5 0

k	$w = \emptyset$	**output**
a	a	
b	b	0
r	r	1
a	a	4
c	c	0
a	a	2
d	d	0
a	a	3
b	ab	
a	a	5
r	r	0
a	ra	
b	b	7
r	br	
a	a	6
		0

c	**word**
0	a
1	b
2	c
3	d
4	r
5	ab
6	br
7	ra
8	ac
9	ca
10	ad
11	da
12	aba
13	ar
14	rab
15	bra

Minh họa thuật toán nén dữ liệu

Xét một chuỗi ký tự "abracadabarabra" $\rightarrow$ 0 1 4 0 2 0 3 5 0 7

k	$w = \emptyset$	**output**
a	a	
b	b	0
r	r	1
a	a	4
c	c	0
a	a	2
d	d	0
a	a	3
b	ab	
a	a	5
r	r	0
a	ra	
b	b	7
r	br	
a	a	6
		0

c	**word**
0	a
1	b
2	c
3	d
4	r
5	ab
6	br
7	ra
8	ac
9	ca
10	ad
11	da
12	aba
13	ar
14	rab
15	bra

Minh họa thuật toán nén dữ liệu

Xét một chuỗi ký tự "abracadabarabra" $\rightarrow$ 0 1 4 0 2 0 3 5 0 7

k	$w = \emptyset$	**output**
a	a	
b	b	0
r	r	1
a	a	4
c	c	0
a	a	2
d	d	0
a	a	3
b	ab	
a	a	5
r	r	0
a	ra	
b	b	7
r	br	
a	a	6
		0

c	**word**
0	a
1	b
2	c
3	d
4	r
5	ab
6	br
7	ra
8	ac
9	ca
10	ad
11	da
12	aba
13	ar
14	rab
15	bra

Minh họa thuật toán nén dữ liệu

Xét một chuỗi ký tự "abracadabarabra" $\rightarrow$ 0 1 4 0 2 0 3 5 0 7 6

k	$w = \emptyset$	**output**
a	a	
b	b	0
r	r	1
a	a	4
c	c	0
a	a	2
d	d	0
a	a	3
b	ab	
a	a	5
r	r	0
a	ra	
b	b	7
r	br	
a	a	6
		0

c	**word**
0	a
1	b
2	c
3	d
4	r
5	ab
6	br
7	ra
8	ac
9	ca
10	ad
11	da
12	aba
13	ar
14	rab
15	bra

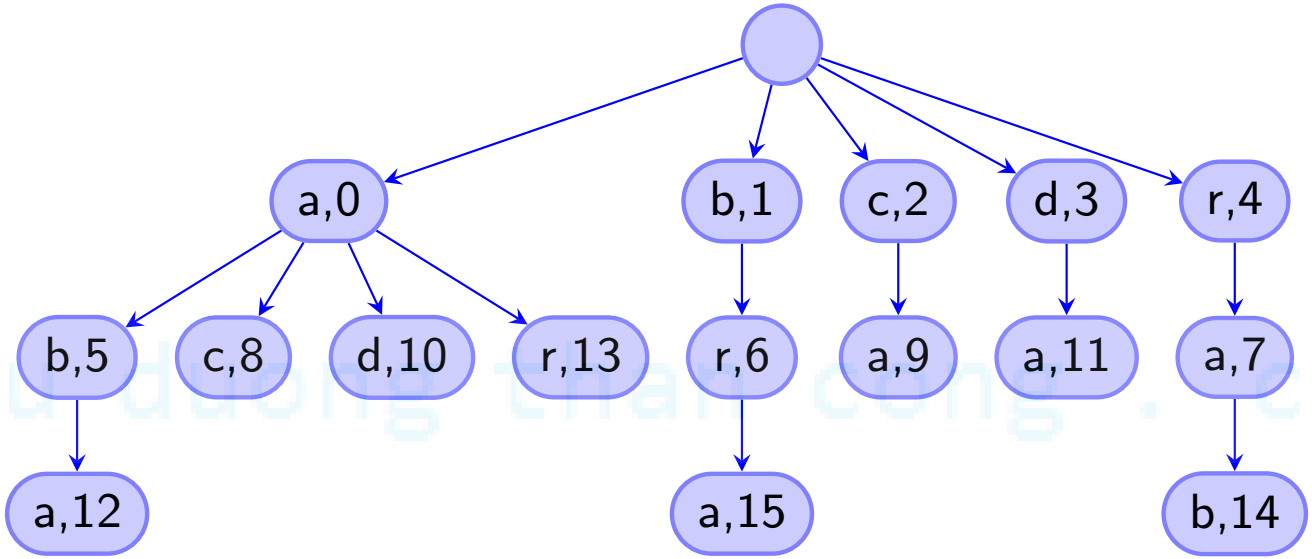
Minh họa thuật toán nén dữ liệu

Xét một chuỗi ký tự "abracadabarabra" $\rightarrow$ 0 1 4 0 2 0 3 5 0 7 6 0

k	$w = \emptyset$	**output**
a	a	
b	b	0
r	r	1
a	a	4
c	c	0
a	a	2
d	d	0
a	a	3
b	ab	
a	a	5
r	r	0
a	ra	
b	b	7
r	br	
a	a	6
		0

c	**word**
0	a
1	b
2	c
3	d
4	r
5	ab
6	br
7	ra
8	ac
9	ca
10	ad
11	da
12	aba
13	ar
14	rab
15	bra

Cây từ điển LZW



Hình 1: Cây từ điển LZW

Thuật toán giải nén dữ liệu

Trong thuật toán giải nén không cần phải cung cấp hết toàn bộ từ điển mà có thể sử dụng lại kỹ thuật của thuật toán nén dữ liệu để xây dựng lại từ điển

```
ĐỌC mã c
w ← WORD(c)
XUẤT từ w
while ( ĐỌC mã c )
    if ( mã c có trong TỪ ĐIỂN )
        tmp ← w
        w ← WORD(c)
        THÊM tmp + w_0 vào TỪ ĐIỂN
    else
        THÊM w + w_0 vào TỪ ĐIỂN
        w ← WORD(c)
    XUẤT từ w
```

Minh họa thuật toán giải nén dữ liệu

Cho chuỗi mã hóa nén dữ liệu "0 1 2 4 3 6" $\rightarrow$

c	w	**output**
0	a	a
1	b	b
2	ab	ab
4	aba	aba
3	ba	ba
6	bab	bab

c	**word**
0	a
1	b
2	ab
3	ba
4	aba
5	abab
6	bab

Minh họa thuật toán giải nén dữ liệu

Cho chuỗi mã hóa nén dữ liệu "0 1 2 4 3 6" $\rightarrow$ a

c	w	**output**
0	a	a
1	b	b
2	ab	ab
4	aba	aba
3	ba	ba
6	bab	bab

c	**word**
0	a
1	b
2	ab
3	ba
4	aba
5	abab
6	bab

Minh họa thuật toán giải nén dữ liệu

Cho chuỗi mã hóa nén dữ liệu "0 1 2 4 3 6" $\rightarrow$ ab

c	w	**output**
0	a	a
1	b	b
2	ab	ab
4	aba	aba
3	ba	ba
6	bab	bab

c	**word**
0	a
1	b
2	ab
3	ba
4	aba
5	abab
6	bab

Minh họa thuật toán giải nén dữ liệu

Cho chuỗi mã hóa nén dữ liệu "0 1 2 4 3 6" $\rightarrow$ abab

c	w	**output**
0	a	a
1	b	b
2	ab	ab
4	aba	aba
3	ba	ba
6	bab	bab

c	**word**
0	a
1	b
2	ab
3	ba
4	aba
5	abab
6	bab

Minh họa thuật toán giải nén dữ liệu

Cho chuỗi mã hóa nén dữ liệu "0 1 2 4 3 6" $\rightarrow$ abababa

c	w	**output**
0	a	a
1	b	b
2	ab	ab
4	aba	aba
3	ba	ba
6	bab	bab

c	**word**
0	a
1	b
2	ab
3	ba
4	aba
5	abab
6	bab

Minh họa thuật toán giải nén dữ liệu

Cho chuỗi mã hóa nén dữ liệu "0 1 2 4 3 6" $\rightarrow$ abababababa

c	w	**output**
0	a	a
1	b	b
2	ab	ab
4	aba	aba
3	ba	ba
6	bab	bab

c	**word**
0	a
1	b
2	ab
3	ba
4	aba
5	abab
6	bab

Minh họa thuật toán giải nén dữ liệu

Cho chuỗi mã hóa nén dữ liệu "0 1 2 4 3 6" $\rightarrow$ abababababab

c	w	**output**
0	a	a
1	b	b
2	ab	ab
4	aba	aba
3	ba	ba
6	bab	bab

c	**word**
0	a
1	b
2	ab
3	ba
4	aba
5	abab
6	bab

Đánh giá LZW

Đánh giá thuật toán LZW

- ?
- ?

Thuật toán nén Huffman

Ý tưởng

Sử dụng tần suất xuất hiện của dữ liệu để nén dữ liệu. Những dữ liệu nào có tần suất cao thì dùng ít bits còn những dữ diệu nào có tần suất thấp thì dùng nhiều bits

Các phương pháp mã hóa dữ liệu

Có rất nhiều phương pháp để mã hóa dữ liệu

- Phương pháp sử dụng một dãy bit có chiều dài cố định (8 bits) để mã hóa ký tự: ASCII
- Phương pháp sử dụng một dãy bit có chiều dài thay đổi để mã hóa ký tự: Morse, Huffman

Ví dụ bảng morse

Bảng 1: Bảng Morse

A	·−	F	··−·	K	−·−	P	·−−·	U	··−
B	−···	G	−−·	L	·−··	Q	−−·−	V	···−
C	−·−·	H	····	M	−−	R	·−·	W	·−−
D	−··	I	··	N	−·	S	···	X	−··−
E	·	J	·−−−	O	−−−	T	−	Y	−·−−
Z	−−··								

Bảng thông kê số lần xuất hiện

Bảng thông kê số lần xuất hiện của ký tự trong dữ liệu được minh họa qua ví dụ dưới đây

- Cho một chuỗi ký tự "ADDAABBCCBAAABBCCCBBBCDAADDEEAA"
- Bảng thống kê số lần xuất hiện của ký tự trong chuỗi

Bảng 2: Bảng tần suất

Ký tự	Tần suất
A	10
B	8
C	6
D	5
E	2

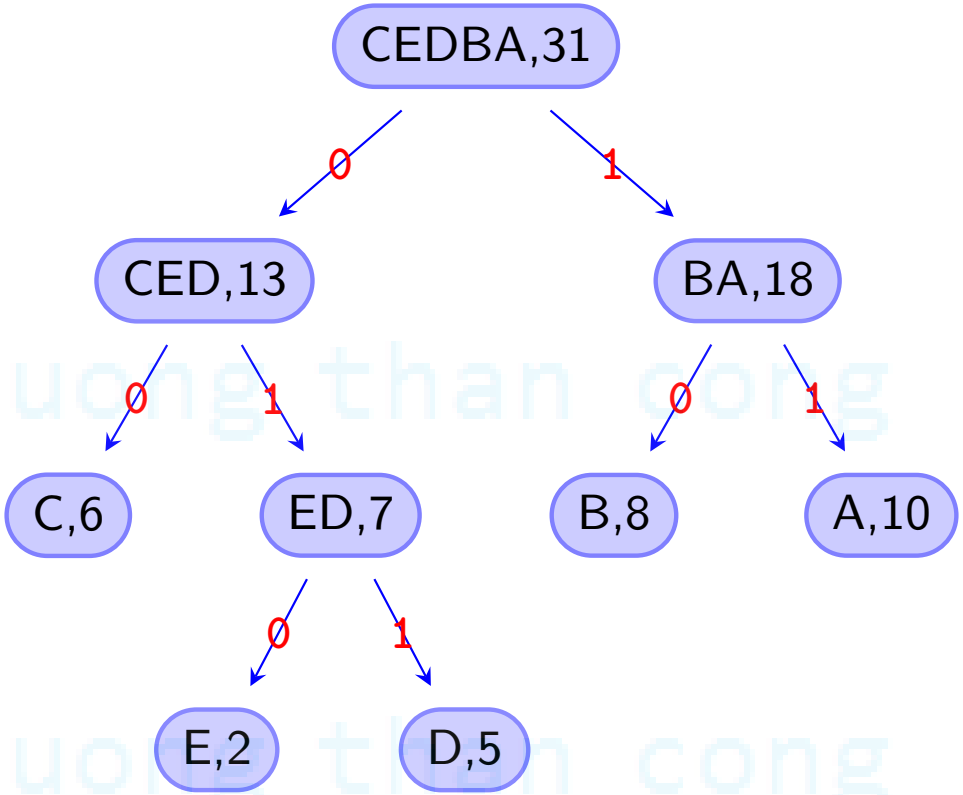
Cây Huffman

Định nghĩa 2

Cây Huffman [Huffman, 1952] là một cây nhị phân đầy đủ

- Về ký tự
  - Nút lá chứa một ký tự
  - Nút cha sẽ chứa các ký tự của những nút con
  - Nút con trái sẽ có thứ tự từ điển trước nút con phải
- Về trọng số: mỗi nút sẽ được gán một trọng số
  - Nút lá có trọng số bằng số lần xuất hiện của ký tự trong dữ liệu
  - Nút cha có trọng số bằng tổng trọng số của các nút con
  - Nút con trái có giá trị trọng số nhỏ hơn hoặc bằng trọng số của nút con trái phải
- Mỗi cung sẽ được gán một giá trị: cung trái là 0 và cung phải là 1

Minh họa cây Huffman



Hình 2: Cây Huffman

Một số tính chất của cây Huffman

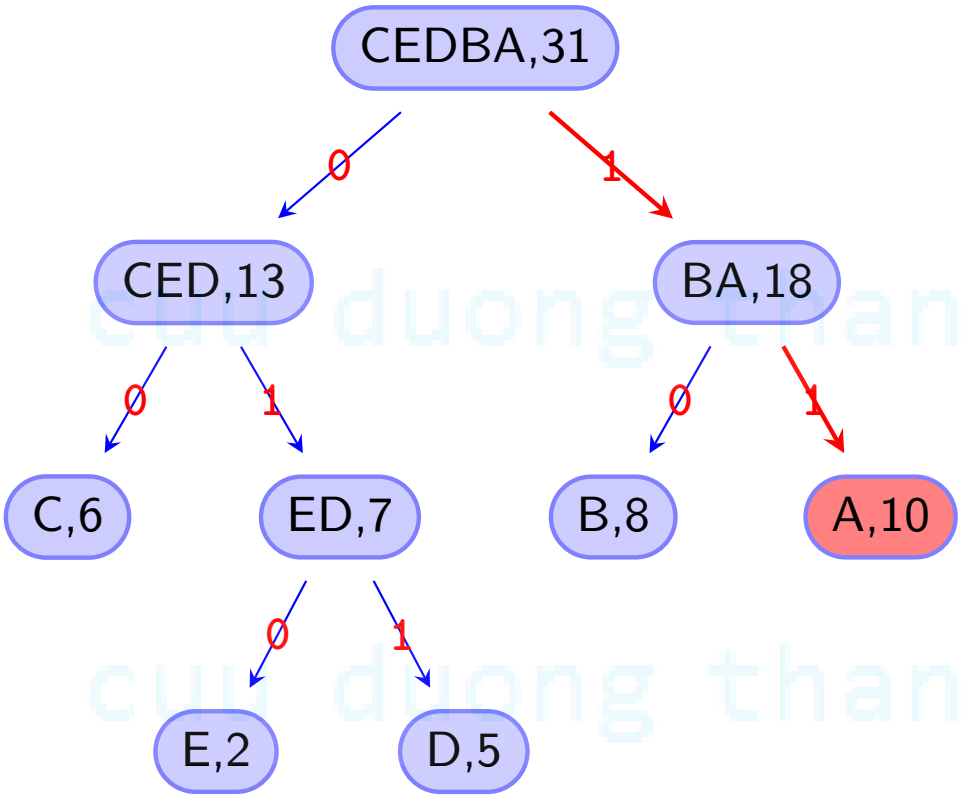
Định lý 1

- *Cây Huffman là cây nhị phân đầy đủ*
- *Các nút có tần suất cao nằm gần gốc*
- *Các nút có tần suất thấp nằm xa gốc*
- *Tổng số nút của cây là* $2n-1$ *với n là số ký tự*

Thuật toán nén Huffman

- Bước 1: Duyệt dữ liệu để lập bảng thống kê số lần xuất hiện của mỗi ký tự
- Bước 2: Tạo cây Huffman từ bảng thống kê
- Bước 3: Tạo bảng mã cho các ký tự
- Bước 4: Duyệt dữ liệu để thay thế các ký tự bằng mã bit tương ứng
- Bước 5: Lưu lại thông tin của cây Huffman dùng để giải nén

Phát sinh bảng mã bit cho cây Huffman



Hình 3: Cây Huffman

Bảng 3: Bảng mã bit

Ký tự	Mã bit
A	11
B	10
C	00
D	011
E	010

Thuật toán tạo cây Huffman

Từ bảng thông kê tần suất xuất hiện của các ký tự ta khởi tạo cây T gồm có n nút là các ký tự với các trọng số của chúng

- Bước 1: Chọn hai phần tử có trọng số thấp nhất x và y
- Bước 2: Tạo một phần tử z từ x và y sao cho trọng số của z là tổng của x và y và chuỗi của z là tổng của x và y
- Bước 3: Loại bỏ x và y khỏi bảng thống kê
- Bước 4: Thêm z vào bảng thống kê và thêm nút z vào cây T với x và y là nút con trái và phải
- Lặp lại các bước trên cho đến khi chỉ còn một phần tử

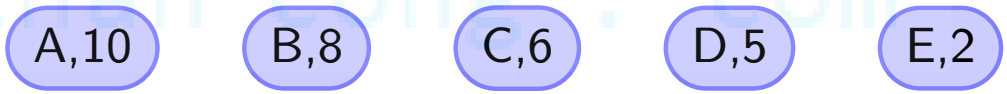
Minh họa thuật toán tạo cây Huffman

Ví dụ 2

Hãy xây dựng cây Huffman cho chuỗi ký tự "ADDAABBCCBAAABBCCCBBBCDAADDEEAA"

Tính bảng bảng tần suất cho các ký tự

Chuỗi ký tự	Tần suất
A	10
B	8
C	6
D	5
E	2



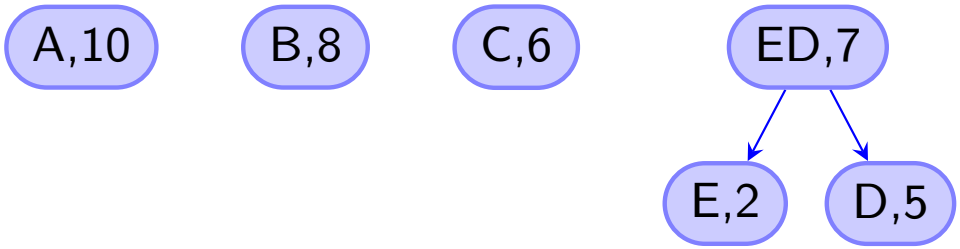
Hình 4: Khởi tạo cây Huffman

Minh họa thuật toán tạo cây Huffman (cont.)

Chuỗi ký tự	Tần suất
A	10
B	8
C	6
D	5
E	2

Loại bỏ D, E và thêm ED rồi sắp xếp lại bảng tần suất

Chuỗi ký tự	Tần suất
A	10
B	8
ED	7
C	6



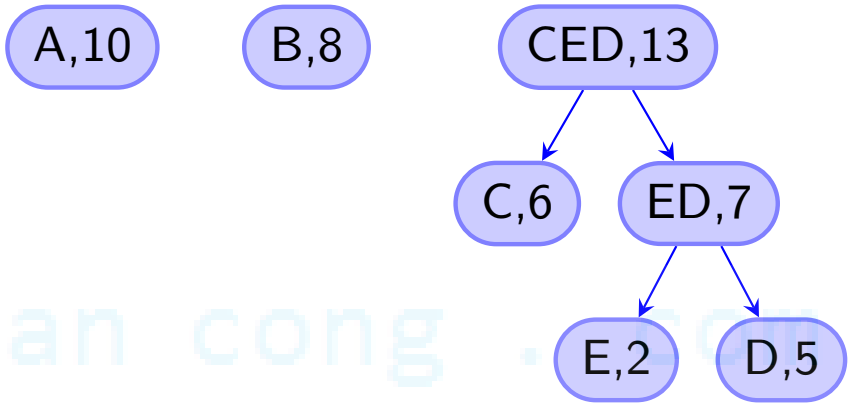
Hình 5: Thêm nút ED cho cây Huffman

Minh họa thuật toán tạo cây Huffman (cont.)

Chuỗi ký tự	Tần suất
A	10
B	8
ED	7
C	6

Loại bỏ C, ED và thêm CED rồi sắp xếp lại bảng tần suất

Chuỗi ký tự	Tần suất
CED	13
A	10
B	8



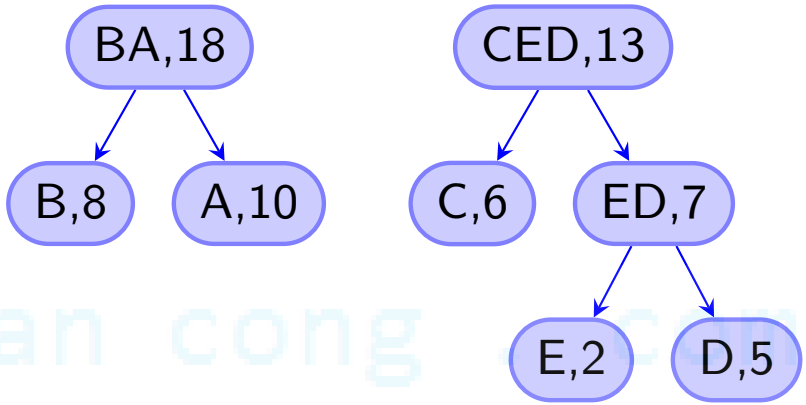
Hình 6: Thêm nút CED cho cây Huffman

Minh họa thuật toán tạo cây Huffman (cont.)

Chuỗi ký tự	Tần suất
CED	13
A	10
B	8

Loại bỏ A, B và thêm BA rồi sắp xếp lại bảng tần suất

Chuỗi ký tự	Tần suất
BA	18
CED	13



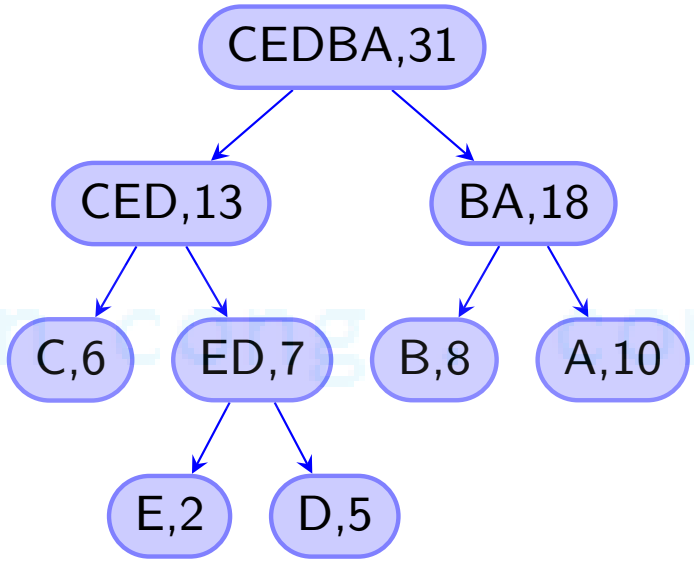
Hình 7: Thêm nút BA cho cây Huffman

Minh họa thuật toán tạo cây Huffman (cont.)

Chuỗi ký tự	Tần suất
BA	18
CED	13

Loại bỏ BA, CED và thêm CEDBA vào bảng tần suất

Chuỗi ký tự	Tần suất
CEDBA	31



Hình 8: Thêm nút CEDBA cho cây Huffman

Một số điểm hạn chế của thuật toán nén Huffman

Hạn chế

- Duyệt dữ liệu hai lần (thống kế và mã hóa) $\Rightarrow$ chi phí cao
- Phải lưu trữ cây Huffman $\Rightarrow$ tăng kích thước dữ liệu nén
- Dữ liệu cần nén phải có sẵn đầy đủ $\Rightarrow$ không nén được trên dữ liệu phát sinh theo thời gian thực

Adaptive Huffman

Lịch sử

- Được đề xuất bởi Faller (1973) và Gallager (1978)
- Knuth (1985) đưa ra một số cải tiến và hoàn chỉnh thuật toán và thuật toán còn có tên "thuật toán FGK"
- Vitter [Vitter, 1987] trình bày các cải tiến liên quan đến việc tối ưu cây Huffman

Adaptive Huffman (cont.)

Ưu điểm

- Không cần tính trước số lần xuất hiện của các ký tự
- Quá trình nén chỉ cần 1 lần duyệt dữ liệu
- Không cần lưu thông tin phục vụ cho việc giải nén
- Nén "on-line" dựa trên dữ liệu phát sinh theo thời gian thực

Adaptive Huffman (cont.)

Định nghĩa 3

Cây Adaptive Huffman (*Adaptive Huffman Tree*) là

- Cây nhị phân đầy đủ
- Mỗi nút có trọng số không âm
- Mỗi nút lá chứa một ký tự và trọng số của nó chính là số lần xuất hiện của ký tự tính đến thời điểm đang xét
- Nút không phải là nút lá trọng số của nó bằng tổng trọng số các nút con của nó
- Có một nút đặc biệt chứa ký hiệu NYT (**N**ot **Y**et **T**ransmitted Symbol) luôn có trọng số bằng 0 (tương ứng với các ký tự chưa xuất hiện trên cây đến thời điểm đang xét)
- Mỗi cung được gán một giá trị: cung trái 0 và cung phải là 1

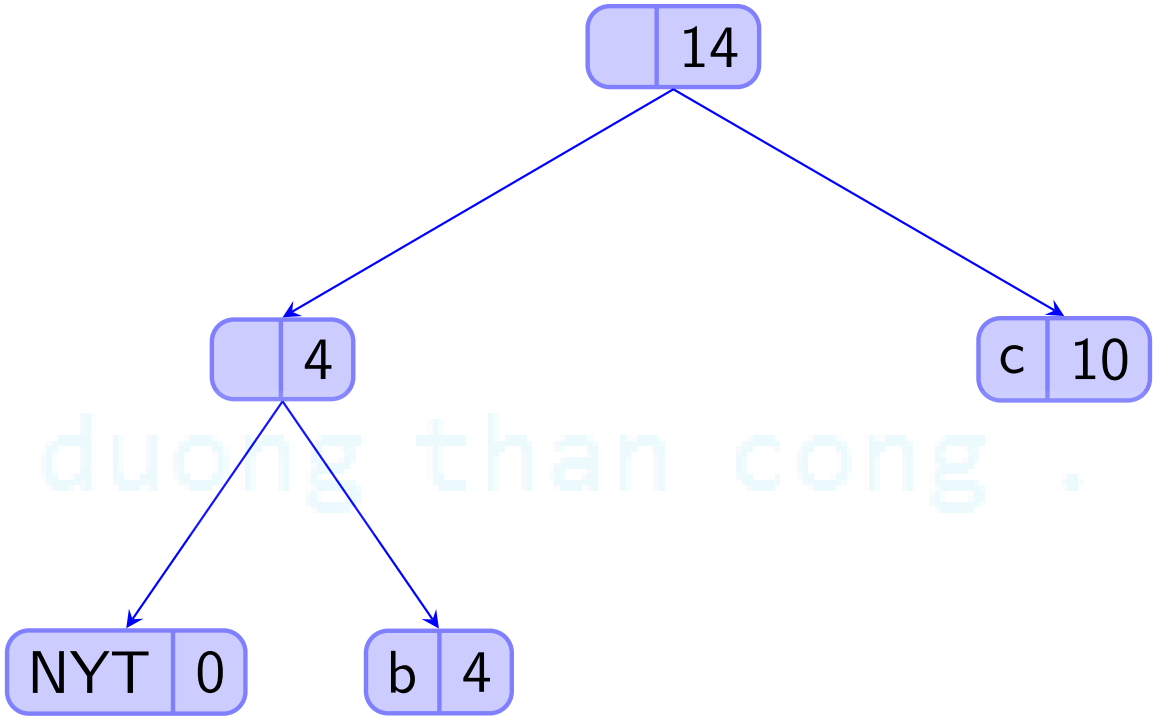
Adaptive Huffman (cont.)

Định nghĩa 4

Cây Adaptive Huffman phải có **tính chất anh em** (*sibling property*)

- Các nút không phải nút gốc phải có nút anh em (vì là cây nhị phân đầy đủ)
- Trọng số tăng dần từ dưới $\rightarrow$ trên, và từ trái $\rightarrow$ phải

Adaptive Huffman (cont.)



Hình 9: Cây Adaptive Huffman, mỗi nút gồm hai phần: phần thứ nhất chứa ký tự đối với nút lá, phần thứ hai chứa trọng số

Thuật toán nén

- Bước 1: Khởi tạo cây Adaptive Huffman
- Bước 2: Lặp nếu con dữ liệu
  - Bước 2.1: Đọc ký tự **c**
  - Bước 2.2: Mã hóa ký tự **c**
  - Bước 2.3: Cập nhật cây Adaptive Huffman

Thuật toán nén (cont.)

Bước 1: Khởi tạo cây Adaptive Huffman

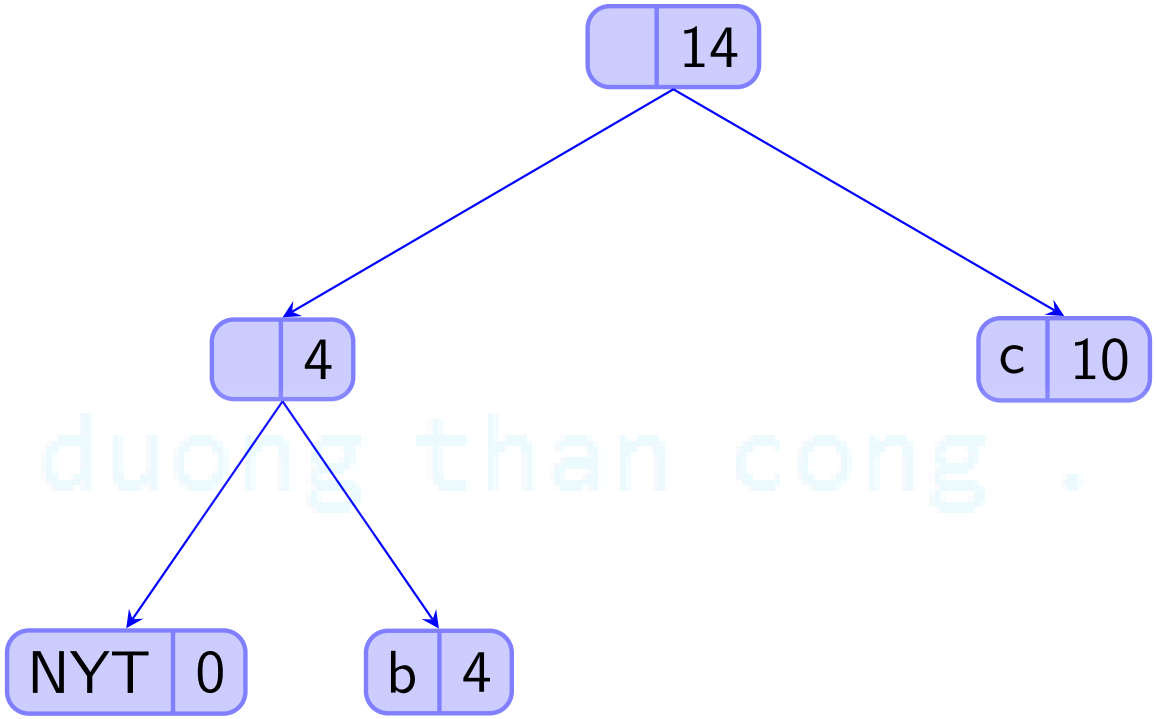
Hình 10: Cây Adaptive Huffman khởi tạo

Thuật toán nén (cont.)

Bước 2.2: Mã hóa ký tự **c**. Kiểm tra ký tự **c** có trong cây Adaptive Huffman hay chưa?

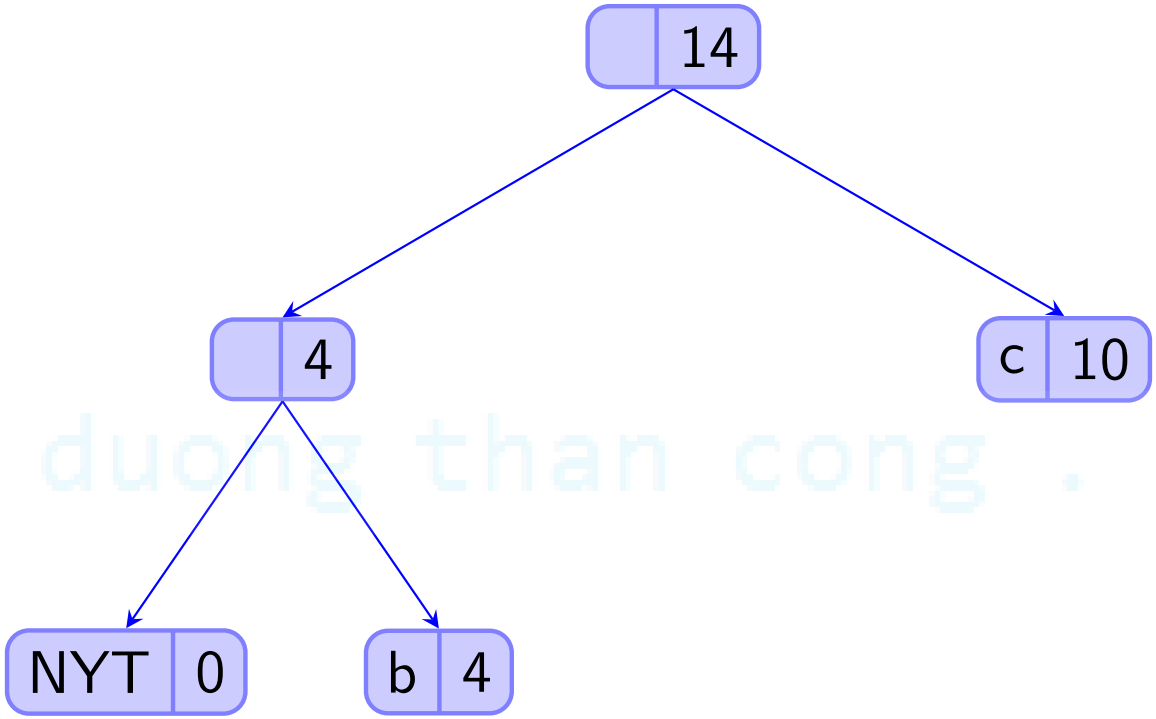
- Nếu có: Phát sinh mã bit cho **c** (giống như cây Huffman thông thường)
- Nếu chưa có: Phát sinh mã bit bao gồm "mã bit của nút NYT + *mã ASCII 8 bit của ký tự* ***c***" (?)

Thuật toán nén (cont.)



Hình 11: Mã hóa ký tự đã có trong cây: mã của **b** là **01**, mã của **c** là **1**

Thuật toán nén (cont.)



Hình 12: Mã hóa ký tự chưa có trong cây: mã của d là : **00***01100100*

Thuật toán nén (cont.)

Bước 2.3: Cập nhật cây

- Nếu **c** có trong cây? Gọi **p** là nút lá chứa **c**
- Nếu **c** chưa có trong cây? tách nút NYT thành hai nút
  - nút NYT (mới) trọng số 0
  - và nút lá **p** chứa **c** có trọng số khởi tạo là 0

Lặp **p** $\neq$ NULL

1. Nếu **p** lá nút gốc tăng trọng số lên 1
2. Hoán đổi **p** với **nút xa nhất** (tính từ nút **p** theo hướng trái $\rightarrow$ phải và dưới $\rightarrow$ trên) có **cùng trọng số** (không tính nút cha của **p**)
3. Tăng trọng số của **p** lên 1
4. Di chuyển **p** lên nút cha của nó

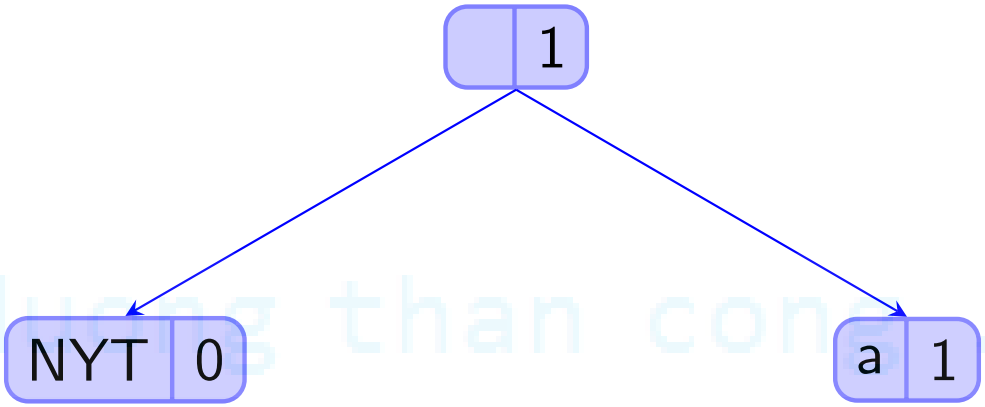
Thuật toán nén (cont.)

Cho cây Adaptive ban đầu hãy cập nhật cây với chuỗi ký tự “aad”

Hình 13: Cây Adaptive Huffman khởi đầu

Thuật toán nén (cont.)

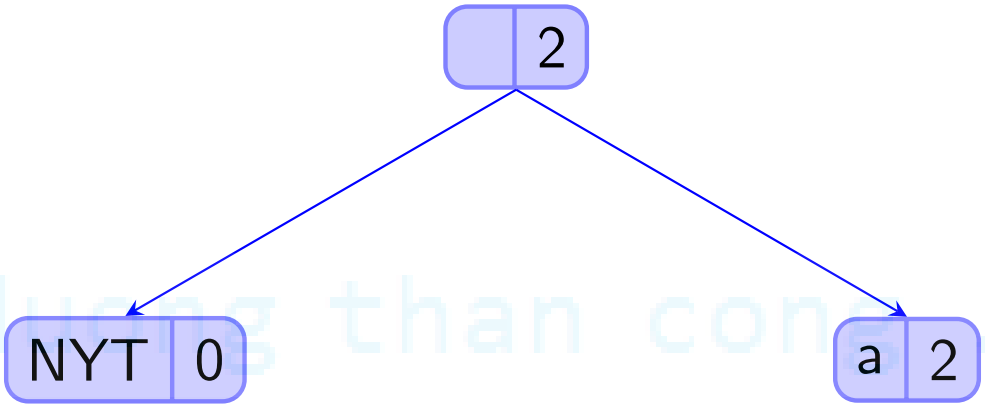
Đọc dữ liệu "**a**ad"



Hình 14: Cập nhật cây sau khi thêm **a**

Thuật toán nén (cont.)

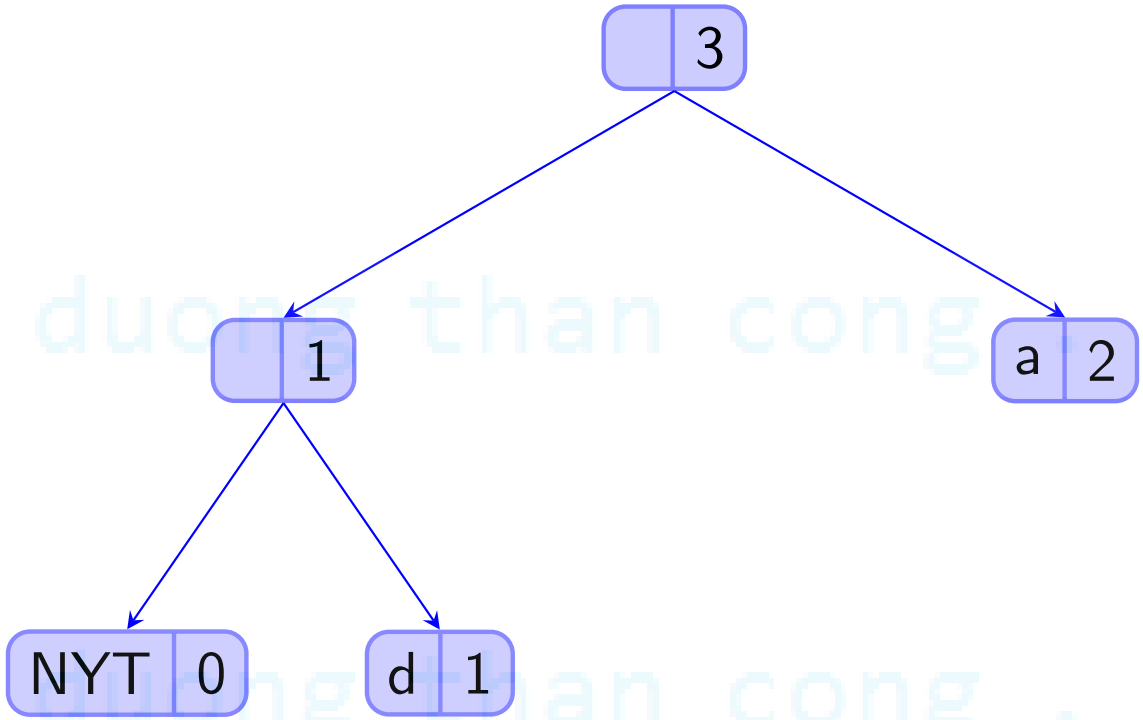
Đọc dữ liệu "a**a**d"



Hình 15: Cập nhật cây sau khi thêm **a**

Thuật toán nén (cont.)

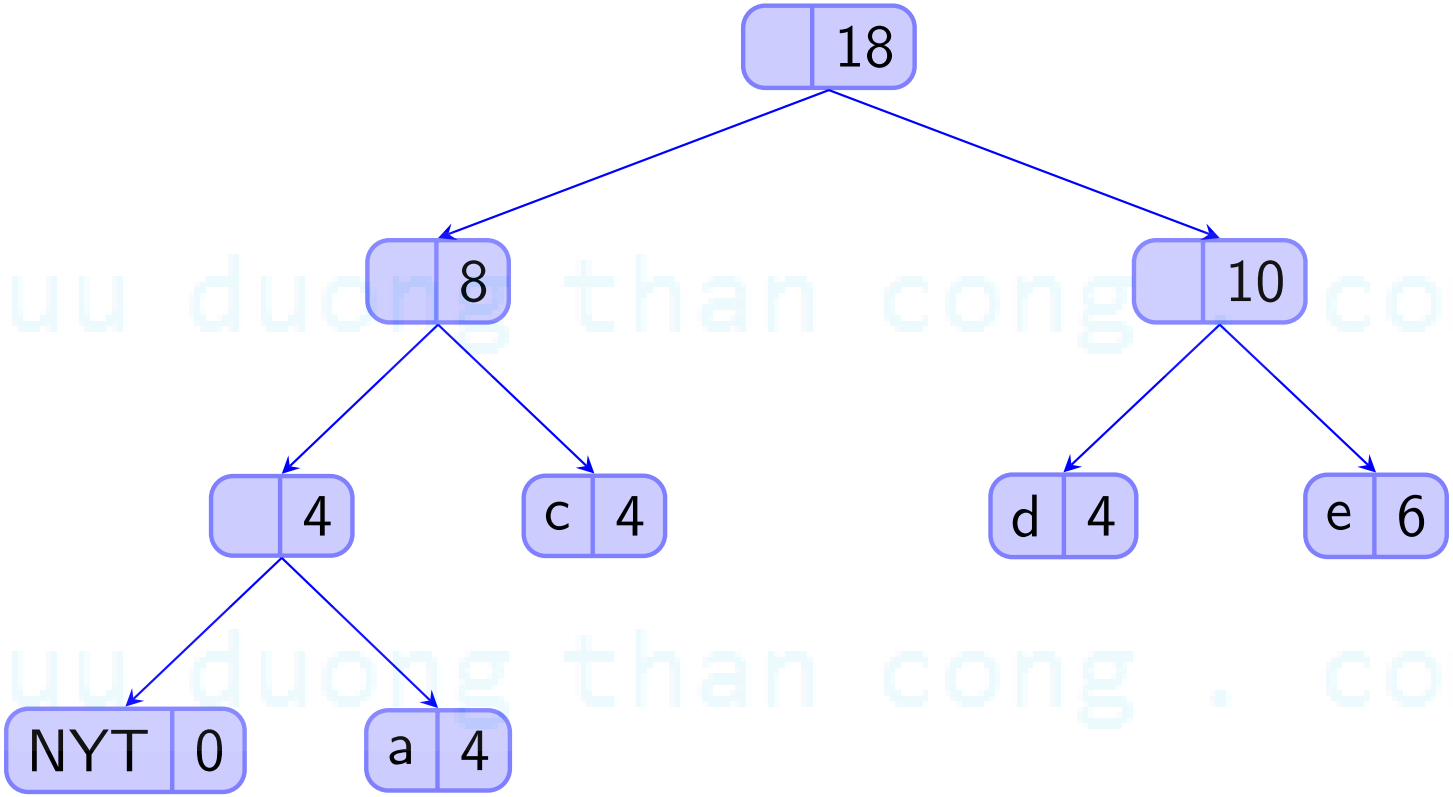
Đọc dữ liệu "aa**d**"



Hình 16: Cập nhật cây sau khi thêm **d**

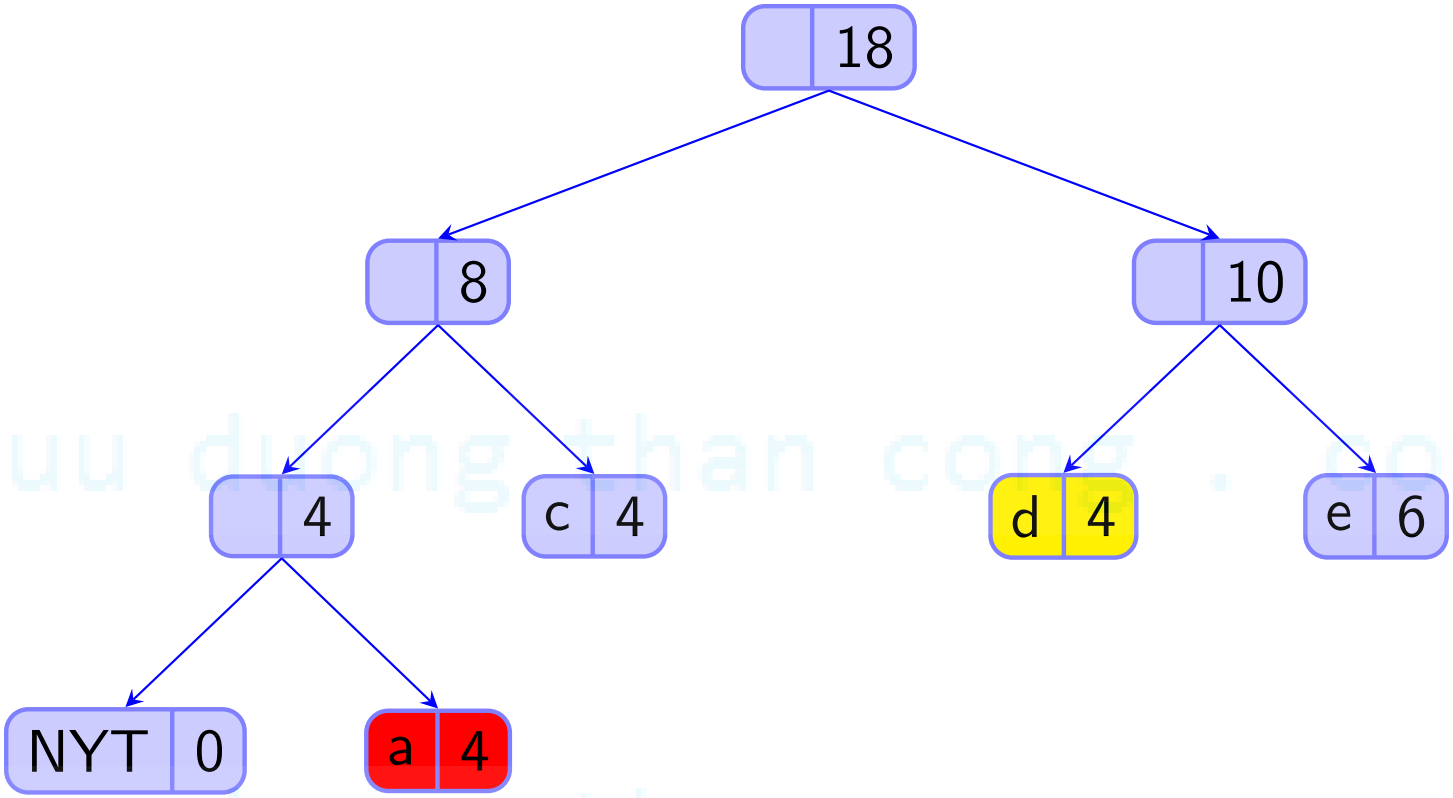
Minh họa cập nhật trọng số

Cho một cây Adaptive Huffman. Và thêm **a** vào cây



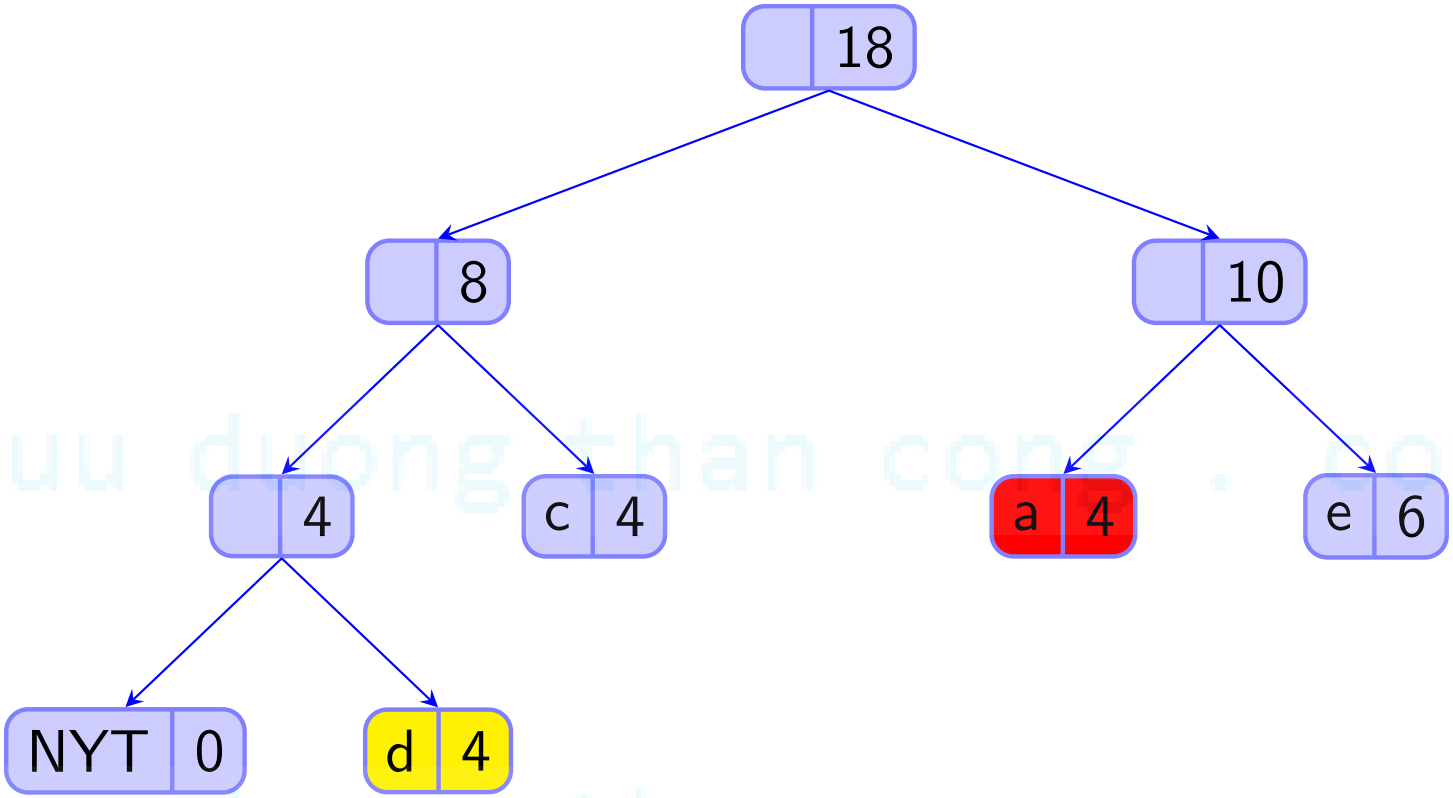
Hình 17: Cây Adaptive Huffman

Minh họa cập nhật trọng số (cont.)



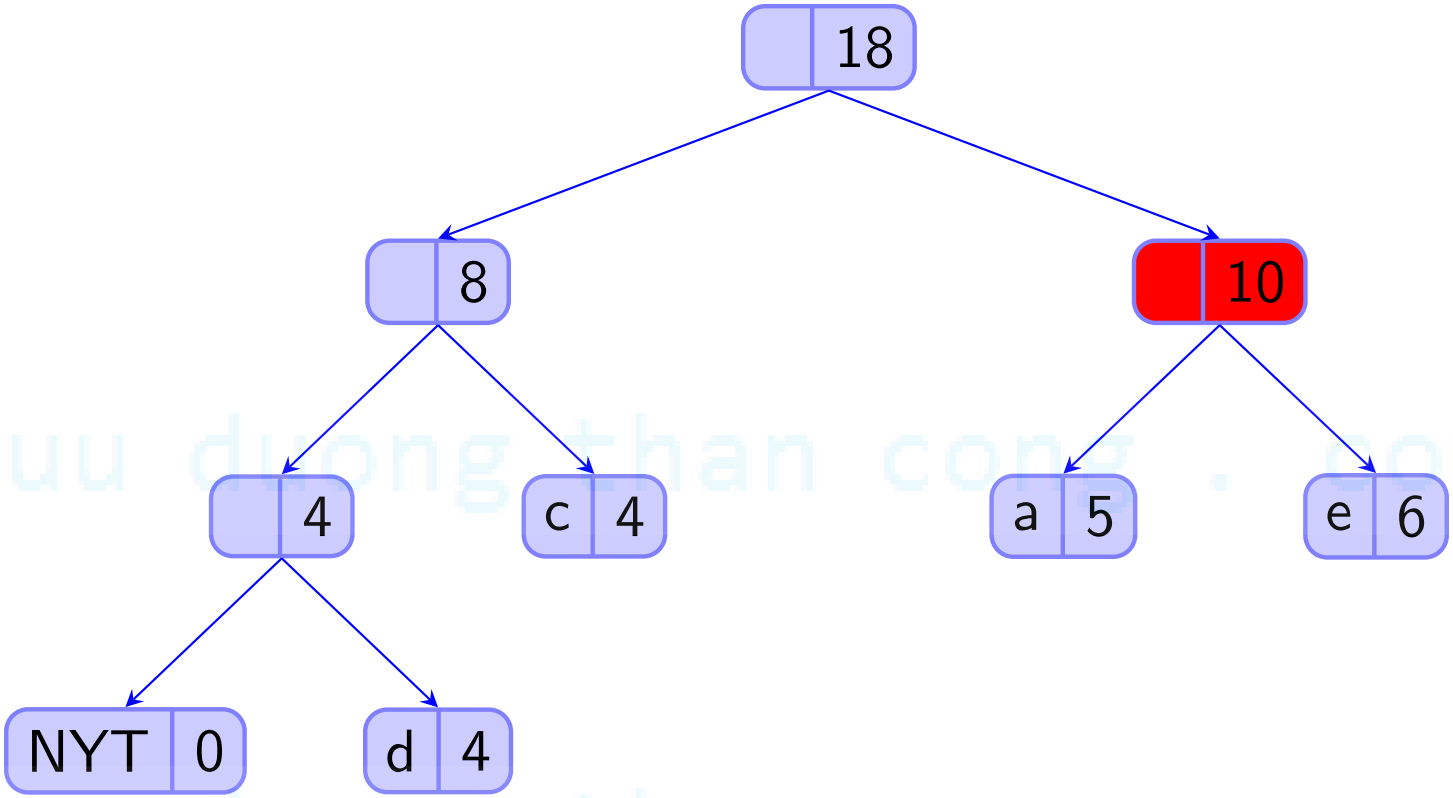
Hình 18: Tìm nút xa nhất có trọng số bằng với a và chuẩn bị hoán đổi

Minh họa cập nhật trọng số (cont.)



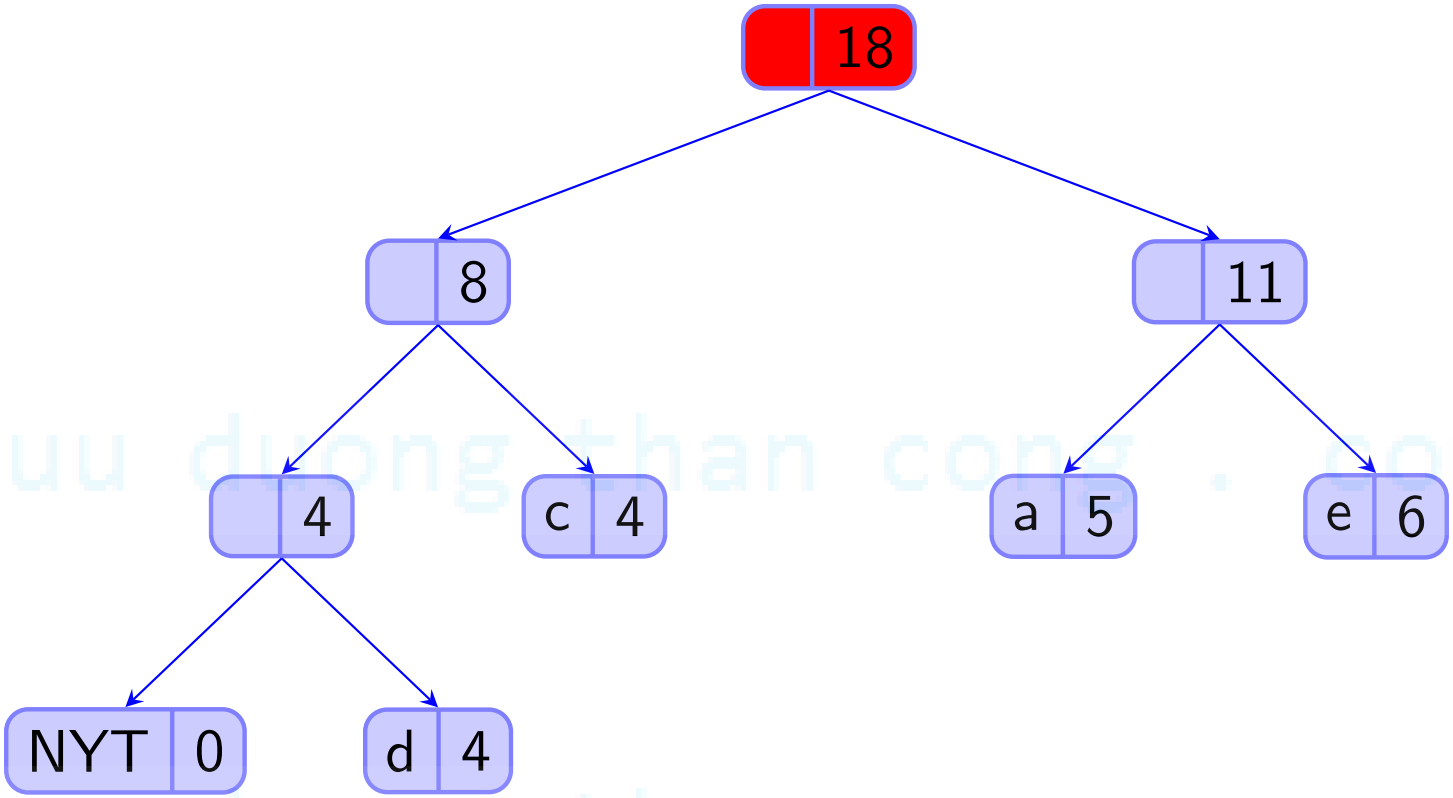
Hình 19: Hoán đổi

Minh họa cập nhật trọng số (cont.)



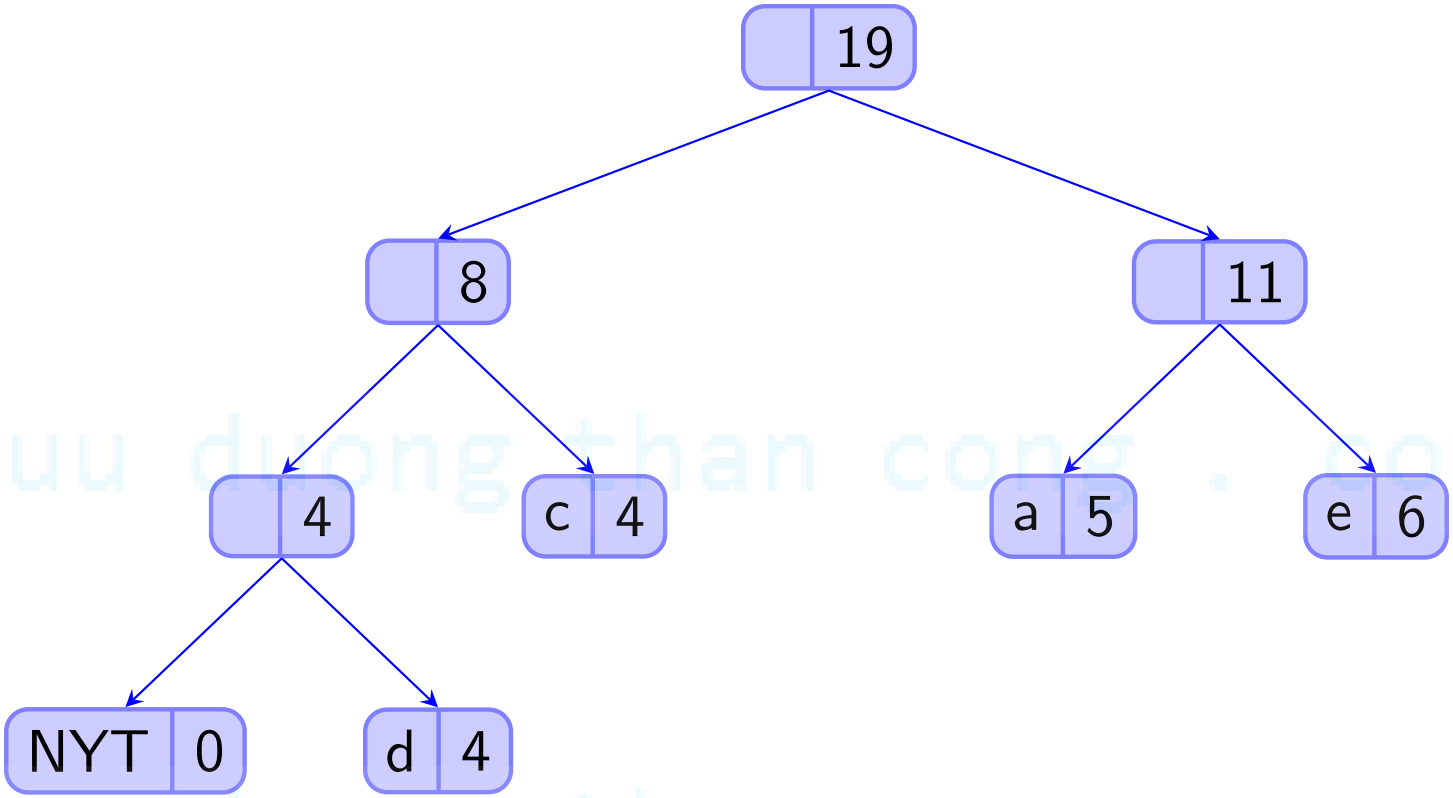
Hình 20: Cập nhật trọng số và di chuyển lên nút cha

Minh họa cập nhật trọng số (cont.)



Hình 21: Cập nhật trọng số và di chuyển lên nút cha

Minh họa cập nhật trọng số (cont.)



Hình 22: Cập nhật trọng số và kết thúc

Thuật toán giải nén

- Bước 1: Khởi tạo cây Adaptive Huffman
- Bước 2: Lặp nếu còn dữ liệu
  - Bước 2.1: Đọc ký tự **c**
  - Bước 2.2: Mã hóa ký tự **c**
  - Bước 2.3: Cập nhật cây Adaptive Huffman

Đánh giá cây Huffman và Adaptive Huffman

Đánh giá thuật toán cây Huffman và Adaptive Huffman

- ?
- ?

Tài liệu tham khảo

Huffman, D. A. (1952).
A method for the construction of minimum-redundancy codes.
Proceedings of the IRE, 40(9):1098–1101.

Lempel, A. (1978).
Compression of individual sequences via variable-rate coding.
IEEE Transactions on Information Theory, 24(5):530–536.

Vitter, J. S. (1987).
Design and analysis of dynamic huffman codes.
Journal of the ACM (JACM), 34(4):825–845.