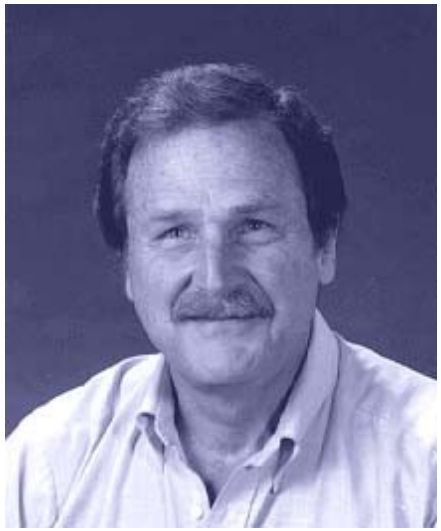
ADAPTIVE HUFFMAN



David A. Huffman
(9/8/1925 – 7/10/1999)

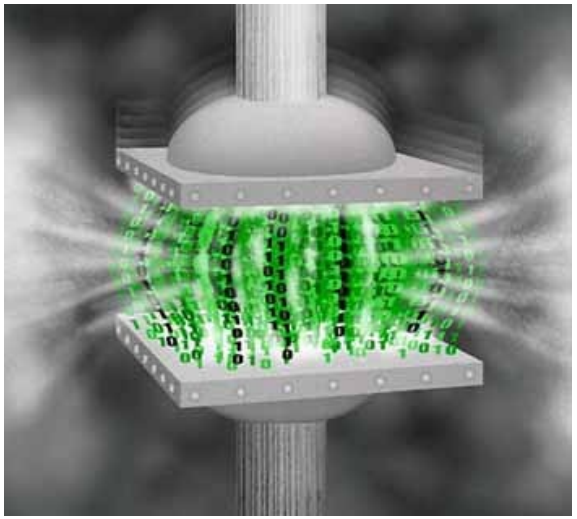
Topic 9: Adaptive Huffman

Nhóm thực hiện:

- Lữ Cao Tiến 0612444
- Nguyễn Khắc Tiệp 0612449
- Lê Phước Trung 0612461
- Lưu Đức Trí 0612484

Adaptive Huffman

- Giới thiệu:
  - ✓ Hạn chế của thuật toán Huffman tĩnh.
  - ✓ Ý tưởng.
  - ✓ Lịch sử hình thành.
  - ✓ Ưu điểm.
- Thuật toán tổng quát.
- Cây Huffman (động).
  - ✓ Tính chất anh em (Sibling property)
  - ✓ Hình thành và cập nhật cây.
  - ✓ Các vi phạm và cách giải quyết
- Thuật toán nén (Encoding)
- Thuật toán giải nén (Decoding)
- Demo minh họa.



Adaptive Huffman - Giới thiệu:

- **Giới thiệu:**
  - ✓ Hạn chế của thuật toán Huffman tĩnh.
  - ✓ Ý tưởng.
  - ✓ Lịch sử hình thành.
  - ✓ Ưu điểm.
- Thuật toán tổng quát.
- Cây Huffman (động).
  - ✓ Tính chất anh em (Sibling property)
  - ✓ Hình thành và cập nhật cây.
  - ✓ Các vi phạm và cách giải quyết
- Thuật toán nén (Encoding)
- Thuật toán giải nén (Decoding)
- Demo minh họa.

Adaptive Huffman - Giới thiệu (tt):

❖ Hạn chế của thuật toán Huffman tĩnh.

✓ Trong quá trình nén cần đến 2 lần duyệt File

→ Chi phí nén cao.

✓ Cần phải lưu trữ thông tin để giải nén

→ Làm tăng kích thước dữ liệu nén.

✓ Dữ liệu nén cần phải có sẵn

→ Không nén được dữ liệu phát sinh theo thời gian thực.

Adaptive Huffman - Giới thiệu (tt):

❖ Ý tưởng:

✓ Thuật toán này vẫn dựa trên ý tưởng của Huffman là sử dụng một vài bit (bit code) để biểu diễn một kí tự.

✓ Độ dài "mã bit" cho các kí tự không giống nhau:

- Kí tự xuất hiện nhiều lần → biểu diễn bằng mã ngắn.
- Kí tự xuất hiện ít → biểu diễn bằng mã dài.

✓ Tạo sẵn một cây "tối thiểu" ban đầu, dữ liệu nén sẽ được cập nhật dần vào cây.

Giới thiệu (tt):

- Lịch sử hình thành:
- ✓ Được đề xuất (độc lập) bởi Faller (1973) và Gallager (1978)
- ✓ Năm 1985 Knuth đưa ra một số cải tiến và hoàn chỉnh thuật toán. Vì vậy thuật toán này còn được gọi là thuật toán FGK.
- ✓ Năm 1987 Vitter trình bày các cải tiến liên quan tới việc tối ưu cây Huffman.

Giới thiệu (tt):

- Ưu điểm:
  - ✓ Không cần tính trước số lần xuất hiện của các kí tự.
  - ✓ Quá trình nén chỉ cần một lần duyệt file.
  - ✓ Không cần lưu thông tin phục vụ cho việc giải nén.
  - ✓ Nén “online” trên dữ liệu phát sinh theo thời gian thực.

Adaptive Huffman

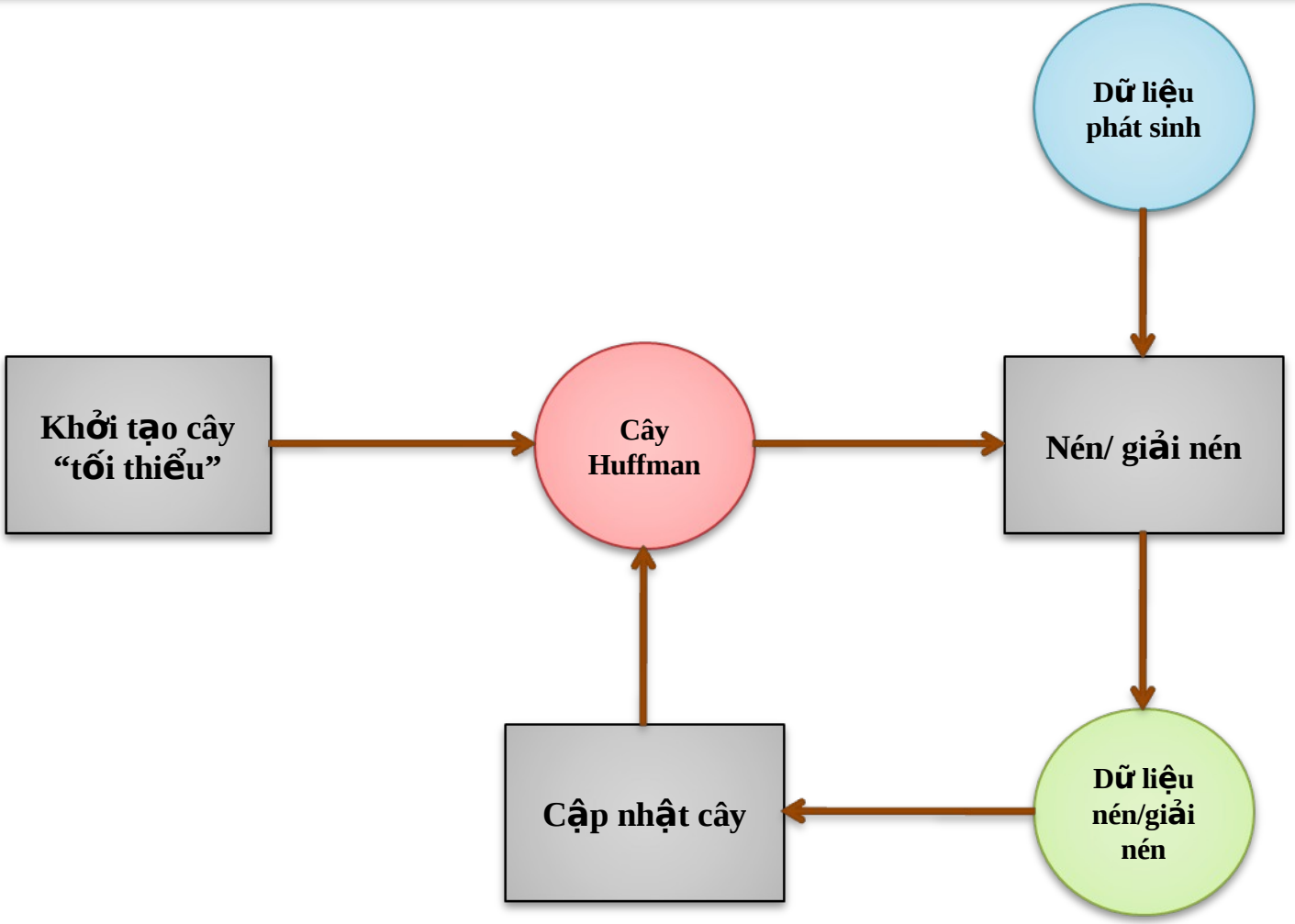
- Giới thiệu:
  - ✓ Hạn chế của thuật toán Huffman tĩnh.
  - ✓ Ý tưởng.
  - ✓ Lịch sử hình thành.
  - ✓ Ưu điểm.
- **Thuật toán tổng quát.**
- Cây Huffman (động).
  - ✓ Tính chất anh em (Sibling property)
  - ✓ Hình thành và cập nhật cây.
  - ✓ Các vi phạm và cách giải quyết
- Thuật toán nén (Encoding)
- Thuật toán giải nén (Decoding)

Adaptive Huffman - Thuật toán tổng quát:

- Static Huffman:
  - Cây Huffman được tạo thành từ bảng thống kê số lần xuất hiện của các kí tự.

- Adaptive Huffman:
  - Nén "online" → không có trước bản thống kê.
  - Phương pháp: khởi tạo cây "tối thiểu" ban đầu, cây sẽ được cập nhật dần dần dựa trên dữ liệu phát sinh trong quá trình nén hoặc giải nén.

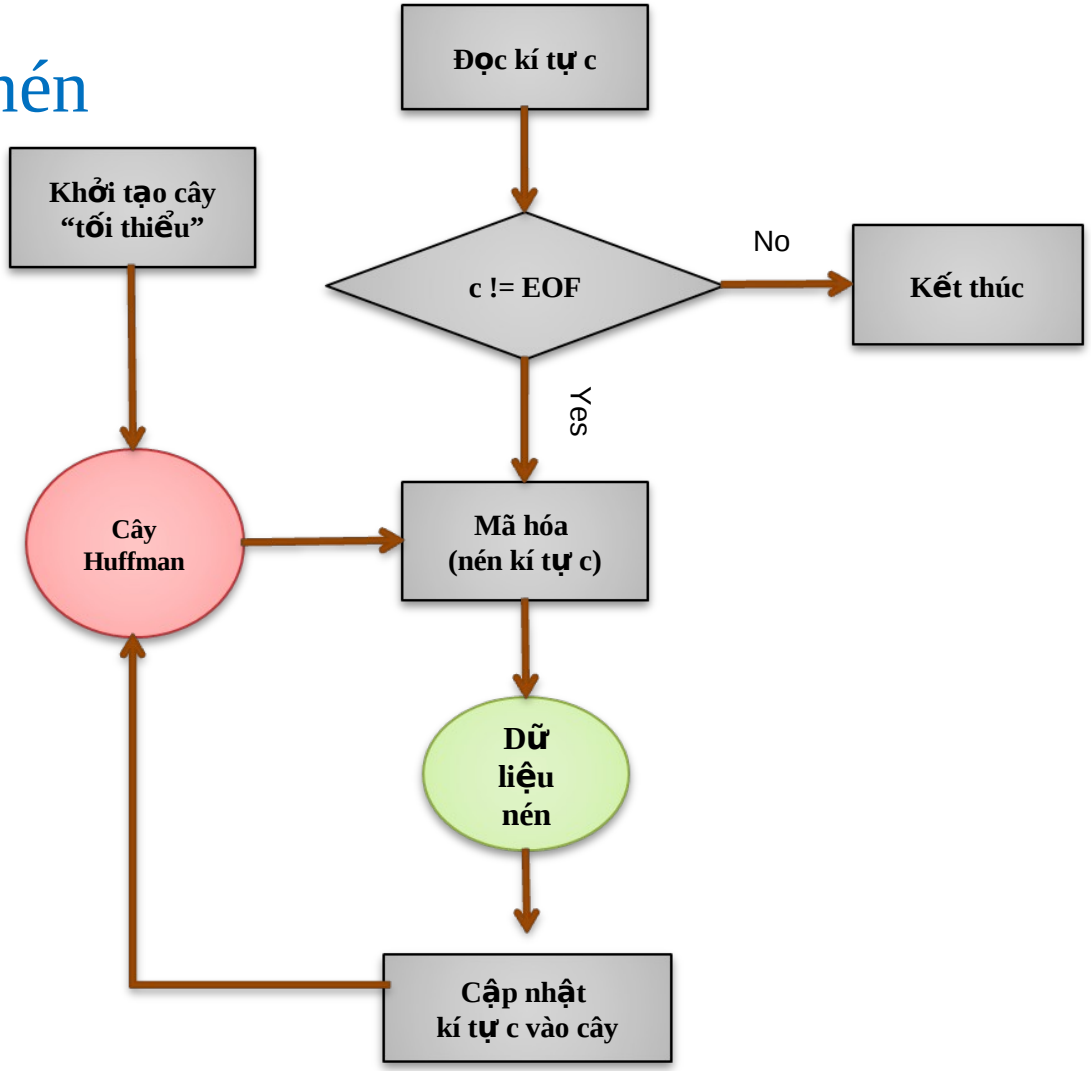
Adaptive Huffman - Thuật toán tổng quát:



Sự phối hợp giữa việc dùng cây (cho thuật toán nén/giải nén) và cập nhật cây

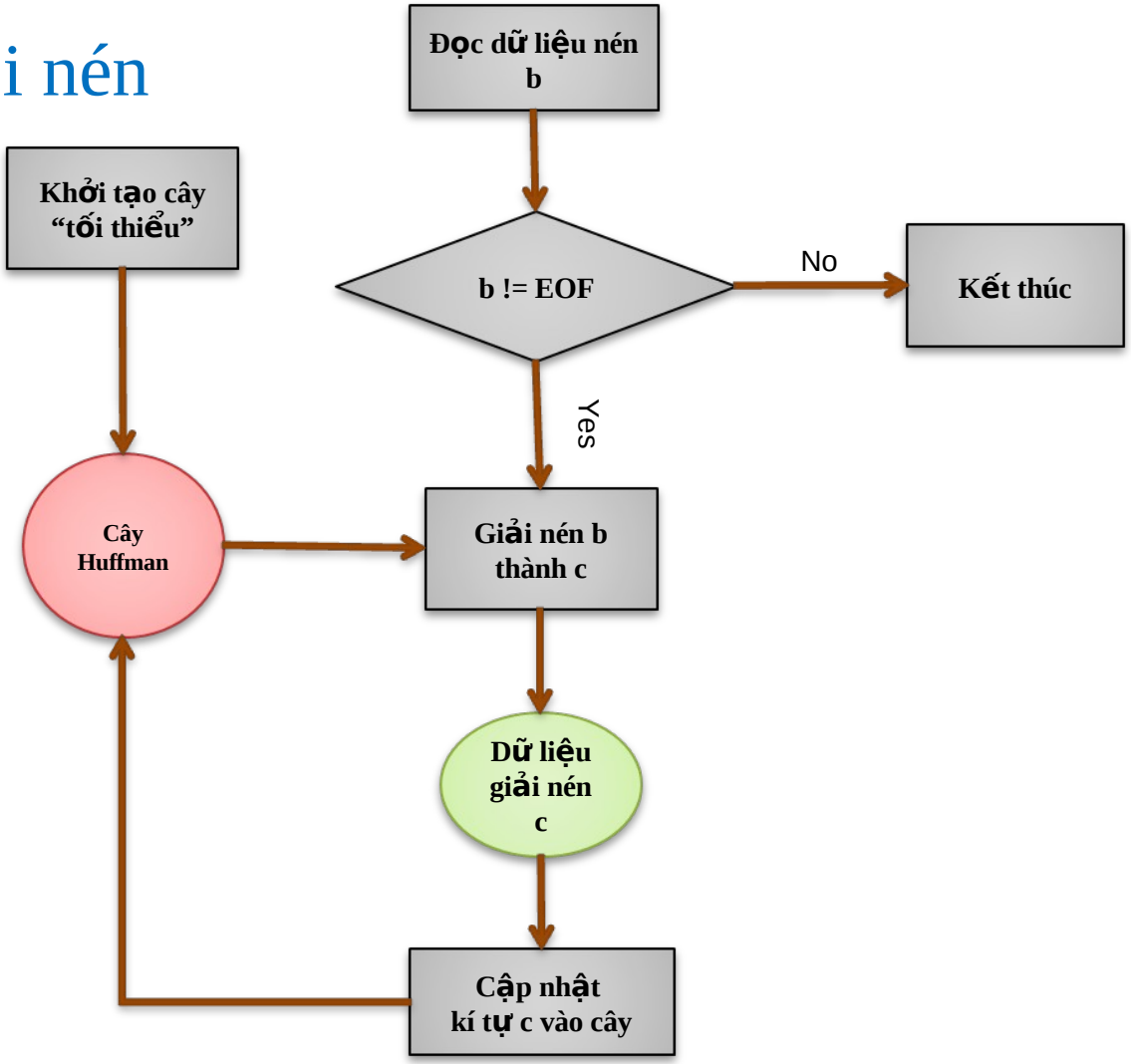
Adaptive Huffman - Thuật toán tổng quát:

Thuật toán nén



Adaptive Huffman - Thuật toán tổng quát:

Thuật toán giải nén



Adaptive Huffman

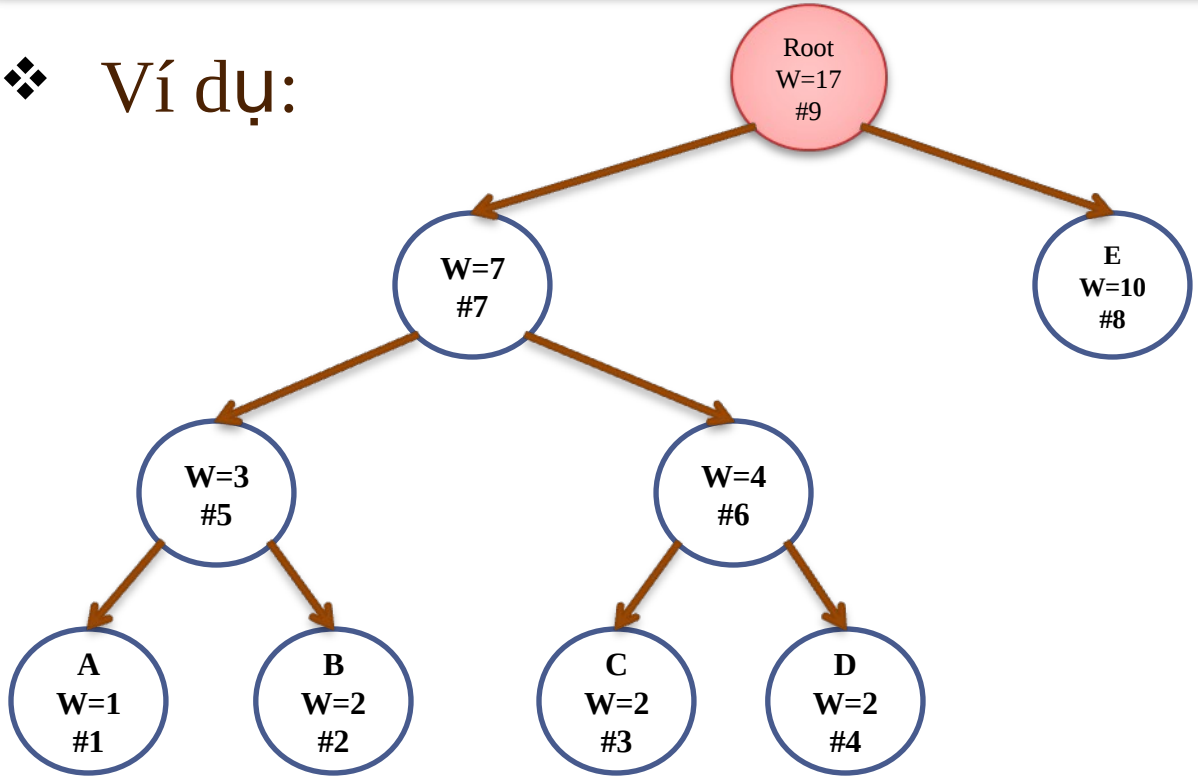
- Giới thiệu:
  - ✓ Hạn chế của thuật toán Huffman tĩnh.
  - ✓ Ý tưởng.
  - ✓ Lịch sử hình thành.
  - ✓ Ưu điểm.
- Thuật toán tổng quát.
- **Cây Huffman (động).**
  - ✓ Tính chất anh em (Sibling property)
  - ✓ Hình thành và cập nhật cây.
  - ✓ Các vi phạm và cách giải quyết
- Thuật toán nén (Encoding)
- Thuật toán giải nén (Decoding)
- Demo minh họa.

Adaptive Huffman - Cây Huffman (động):

- Một cây nhị phân có n node lá được gọi là cây Huffman nếu thỏa:

Adaptive Huffman - Cây Huffman (động):

- Ví dụ:

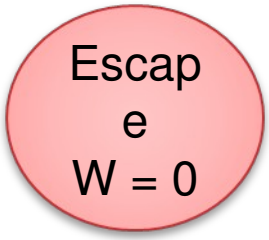


Thứ tự	#1	#2	#3	#4	#5	#6	#7	#8	#9
W_i	1	2	2	2	3	6	7	10	17
Giá trị	A	B	C	D				E	Root

Adaptive Huffman - Cây Huffman (động):

❖ Cách thức tạo cây:

b1: Khởi tạo cây "tối thiểu", chỉ có node Escape (node 0)



b2: Cập nhật từng kí tự vào trong cây theo qui tắc:

- Nếu kí tự chưa có trong cây ⇨ thêm mới node lá
- Nếu kí tự đã có trong cây ⇨ tăng trọng số node lên 1
- Cập nhật trọng số của các node liên quan trong cây

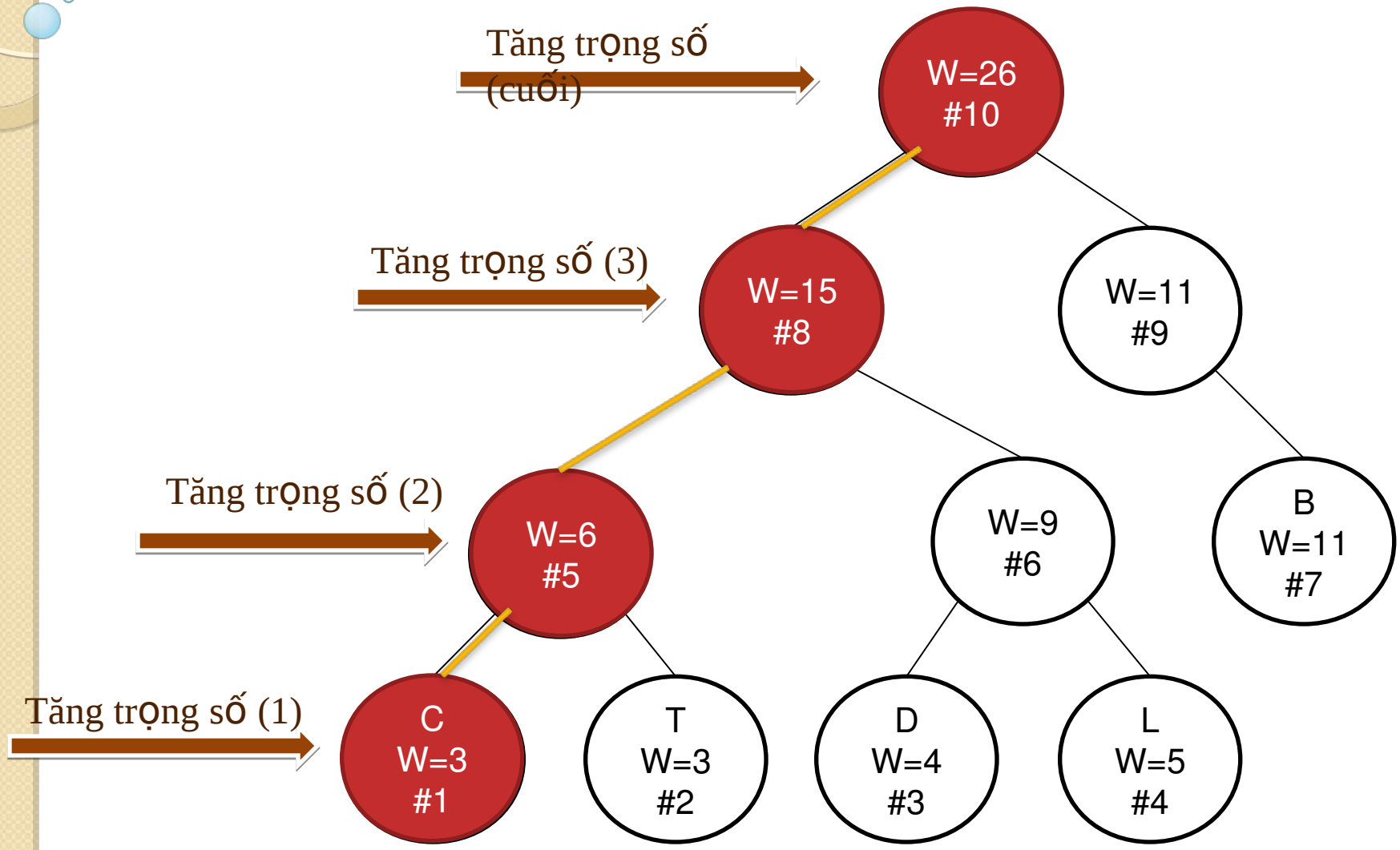
Adaptive Huffman - Cây Huffman (động):

❖Thuật toán cập nhật trọng số:

✓ Tăng trọng số của node lá lên 1.

✓ Đi từ node lá đến node gốc tăng trọng số của các node lên 1. Kiểm tra tính chất anh em và hiệu chỉnh lại cây nếu có vi phạm.

Adaptive Huffman - Cây Huffman (động):

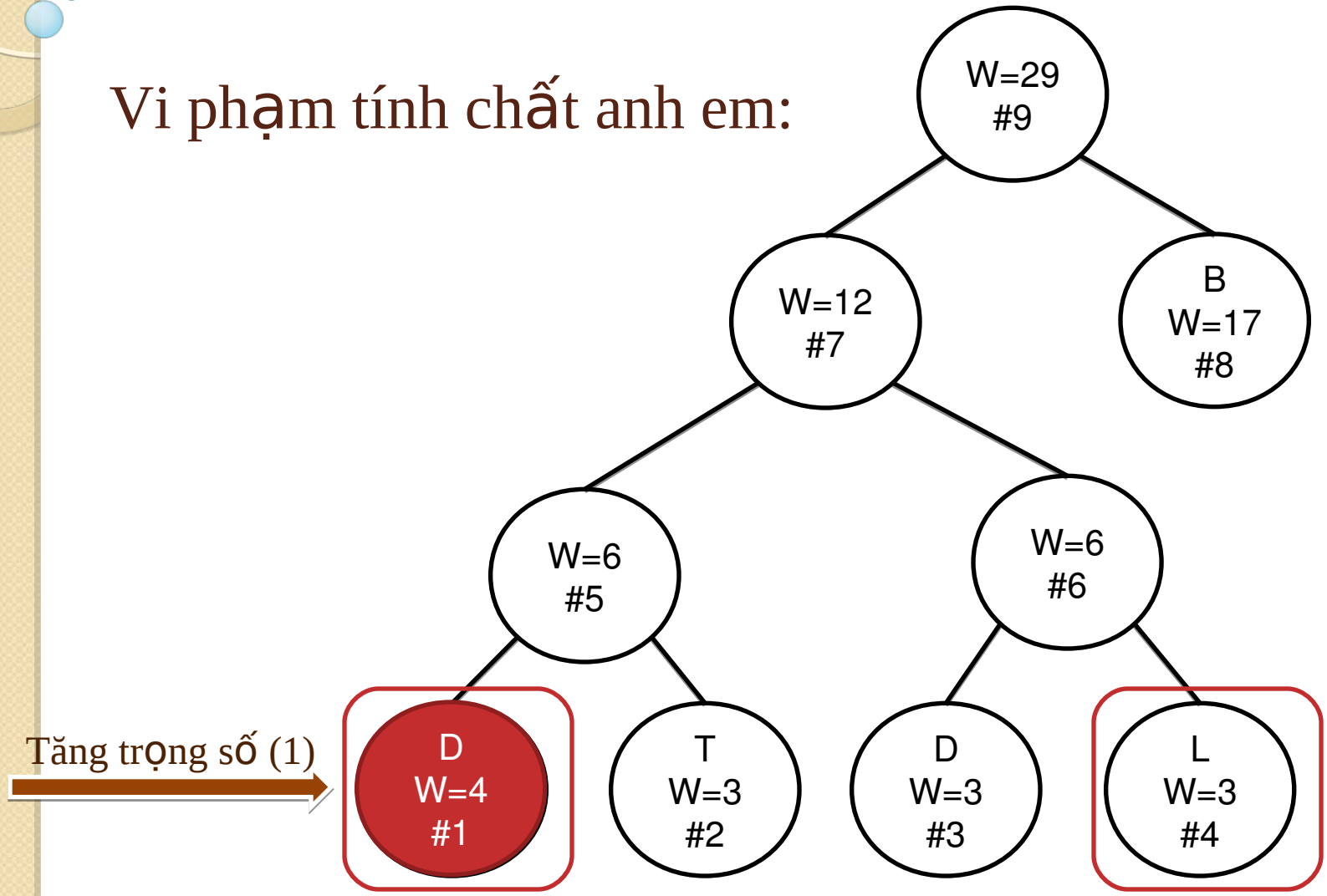


Adaptive Huffman - Cây Huffman (động):

- Khi thêm một node mới hoặc tăng trọng số:
  - ✓ Vi phạm tính chất anh em.
  - ✓ Tràn số.

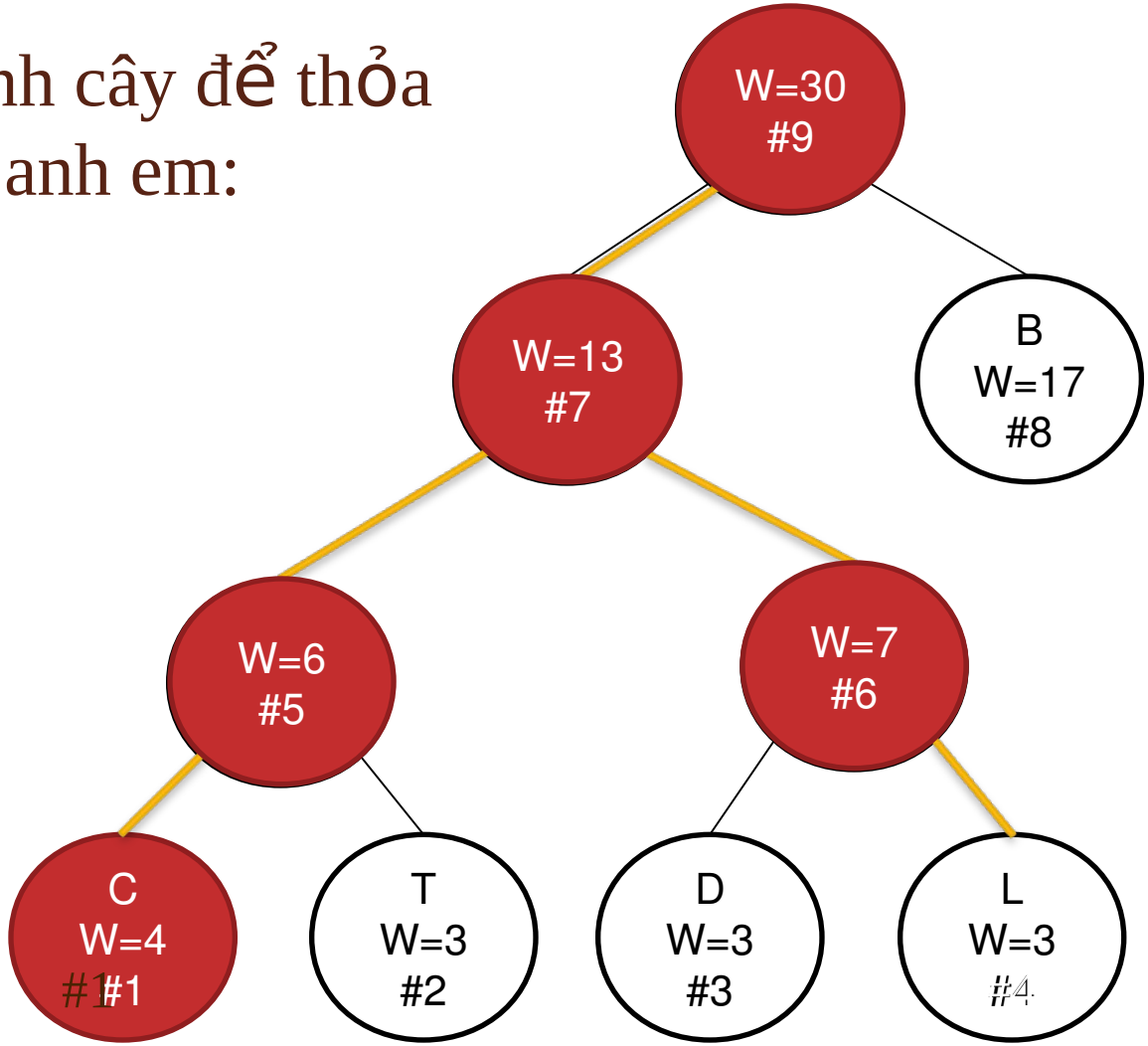
Adaptive Huffman - Cây Huffman (động):

Vi phạm tính chất anh em:



Adaptive Huffman - Cây Huffman (động):

Hiệu chỉnh cây để thỏa tính chất anh em:



Adaptive Huffman - Cây Huffman (động):

- Thuật toán xác định node vi phạm:
  - Gọi x là node hiện hành.
  - So sánh x với các node tiếp theo sau (theo thứ tự từ trái sang phải, từ dưới lên trên).
  - Nếu tồn tại y sao cho y.Weight < x.Weight thì x là node vi phạm.
- Thuật toán hiệu chỉnh cây thỏa tính chất anh em:
  - Gọi x là node vi phạm.
  - Tìm node y xa nhất, có trọng số cao nhất thỏa y.Weight < x.Weight
  - Hoán đổi x và node y trên cây.
  - Cập nhật lại các node cha tương ứng.
  - Lặp lại b1 cho đến khi không còn node vi phạm.

Adaptive Huffman - Cây Huffman (động):

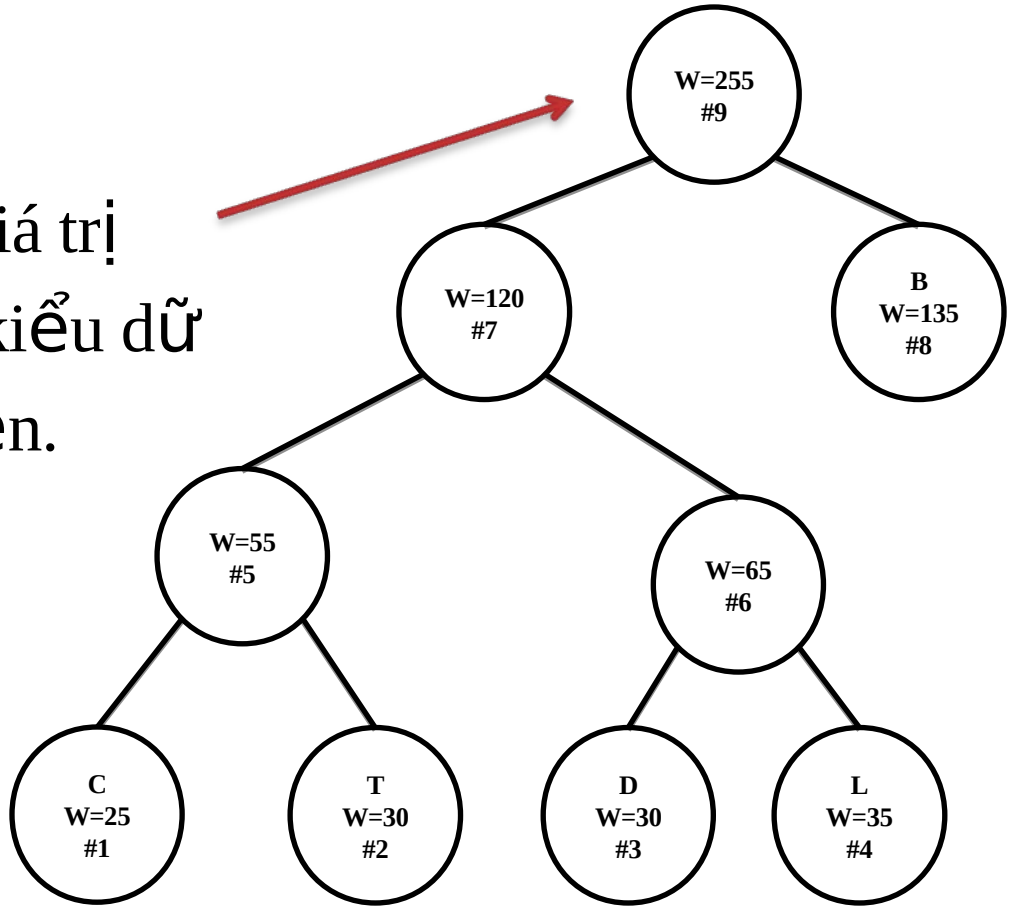
- Vấn đề tràn số:
  - Quá trình cập nhật cây → làm tăng trọng số các node.
  - Trọng số node gốc tăng nhanh → có thể vượt quá khả năng lưu trữ của kiểu dữ liệu.
    - Kiểu int → giá trị max = 2^{15}- 1
    - Kiểu unsigned int → giá trị max = 2^{16} – 1
    - Kiểu long → giá trị max = 2^{31} - 1

Adaptive Huffman - Cây Huffman (động):

- Ví dụ về tràn số:

Node gốc đang có giá trị trọng số tối đa mà kiểu dữ liệu có thể biểu diễn.

Hiện tượng tràn số sẽ xuất hiện khi ta tăng trọng số của bất kì node lá nào



Adaptive Huffman - Cây Huffman (động):

❖ Thuật toán xử lí tràn số:

➢ Khi cập nhật trọng số, kiểm ta trọng số node gốc.

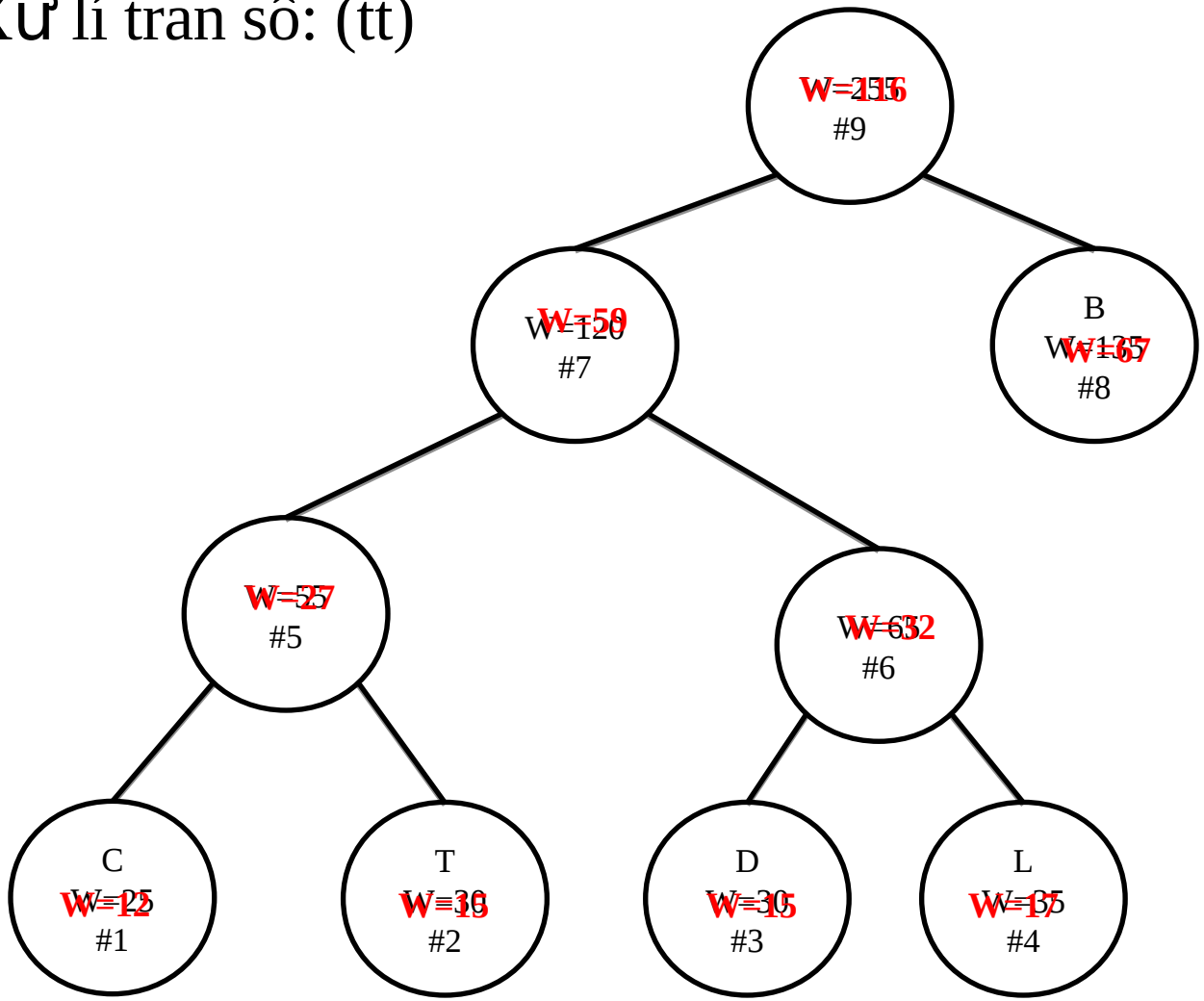
➢ Nếu trọng số node gốc > MAX_VALUE:

- Giảm trọng số các node lá trong cây (chia cho 2).
- Cập nhật lai trọng số cac node nhánh.
- Kiểm tra tính chất anh em và hiệu chỉnh lại cây.

(Do phép chia cho 2 làm mất phần dư của số nguyên)

Adaptive Huffman - Cây Huffman (động):

- Xử lí tràn số: (tt)



Adaptive Huffman

- Giới thiệu:
  - ✓ Hạn chế của thuật toán Huffman tĩnh.
  - ✓ Ý tưởng.
  - ✓ Lịch sử hình thành.
  - ✓ Ưu điểm.
- Thuật toán tổng quát.
- Cây Huffman (động).
  - ✓ Tính chất anh em (Sibling property)
  - ✓ Hình thành và cập nhật cây.
  - ✓ Các vi phạm và cách giải quyết
- **Thuật toán nén (Encoding)**
- Thuật toán giải nén (Decoding)
- Demo minh họa.

Adaptive Huffman

- Thuật toán nén:

```
initialize_model();
while(c != EOF)
{
      c = getchar(inputfile);
      encode(c, outputfile);
      update_model(c);
}
```

Adaptive Huffman

- Thuật toán mã hóa kí tự c:

```
encode (c, outputfile)
```

- Nếu c chưa có trong cây
  - Duyệt cây T tìm mã bit của Escape ghi lên outputfile.
  - Ghi tiếp 8 bit mã ASCII của c lên outputfile.
  - Thêm node c vào cây và cập nhật lại cây.
- Nếu c đã có trong cây:
  - Duyệt cây T tìm mã bit của c và ghi lên outputfile
  - Tăng trọng số của node c lên 1 và cập nhật lại cây

Adaptive Huffman

- Giới thiệu:
  - ✓ Hạn chế của thuật toán Huffman tĩnh.
  - ✓ Ý tưởng.
  - ✓ Lịch sử hình thành.
  - ✓ Ưu điểm.
- Thuật toán tổng quát.
- Cây Huffman (động).
  - ✓ Tính chất anh em (Sibling property)
  - ✓ Hình thành và cập nhật cây.
  - ✓ Các vi phạm và cách giải quyết
- Thuật toán nén (Encoding)
- **Thuật toán giải nén (Decoding)**
- Demo minh họa.

Adaptive Huffman

- Thuật toán giải nén:

```
inputfile: dữ liệu ở dạng nén
outputfile: dữ liệu giải nén

initialize_model();
while((c = decode(inputfile)) != EOF)
{
    putchar(c, outputfile);
    update_model(c);
}
```

Adaptive Huffman

- Thuật toán giải mã kí tự c:

```
decode(inputfile)
b = getchar(inputfile);
```

- Bắt đầu từ vị trí hiện tại trên inputfile.
- Lấy từng bit của b, duyệt trên cây (b==0: trái, b==1: phải).
  - Nếu đi đến node lá x, return x.char
  - Nếu đi đến node Escape:
    - c = lấy 8 bit tiếp theo từ inputfile
    - Thêm c vào cây, cập nhật lại cây.
    - return c

DEMO MINH HỌA

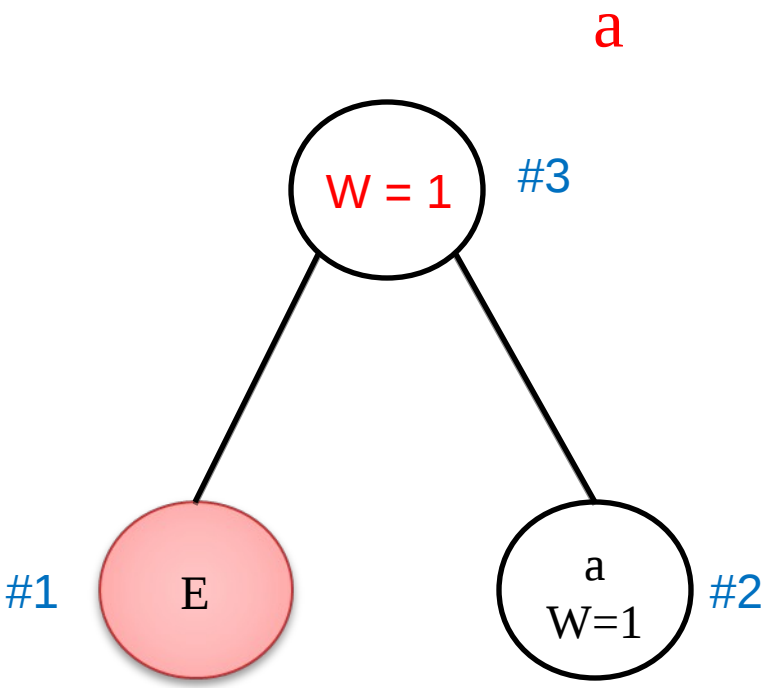
Adaptive Huffman – Ví dụ

Giả sử có dữ liệu như sau:

f = “abacbdc”

Adaptive Huffman – Ví dụ (tt)

B1: Khởi tạo cây và đưa kí tự “a” vào cây:



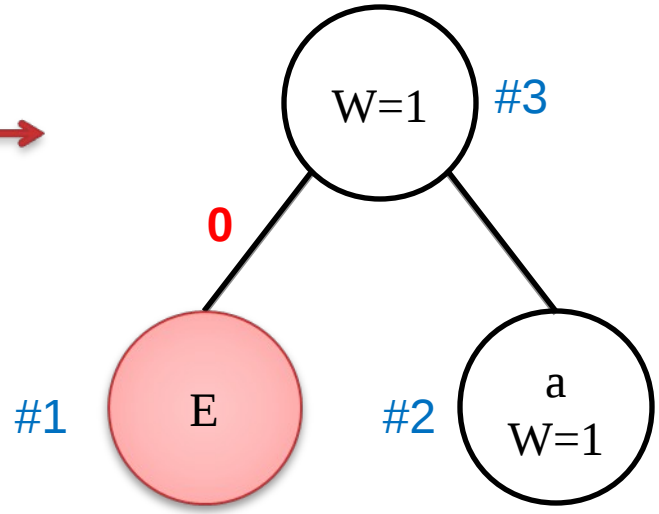
input = 01100001

Adaptive Huffman – Ví dụ (tt)

B2: Thêm kí tự “b” vào trong cây

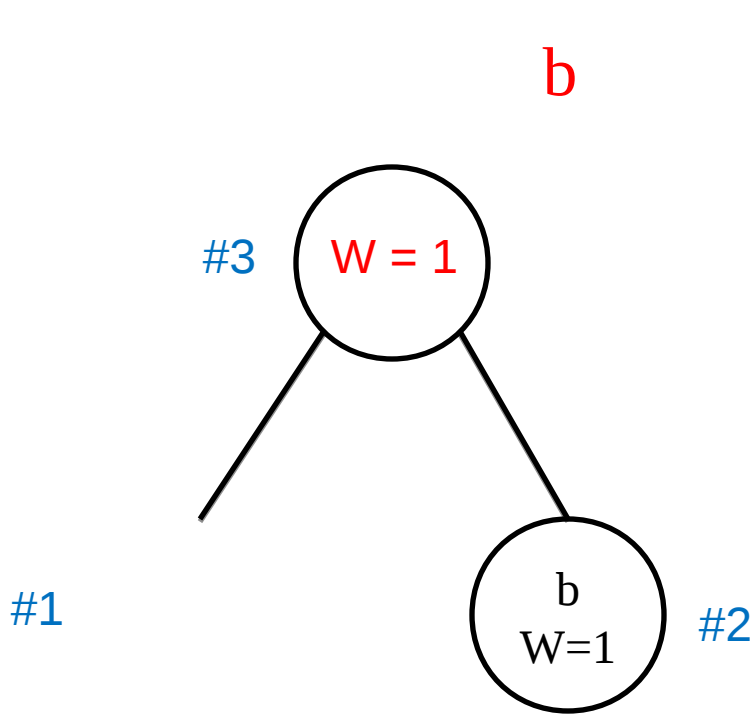
“b” chưa có trong cây →

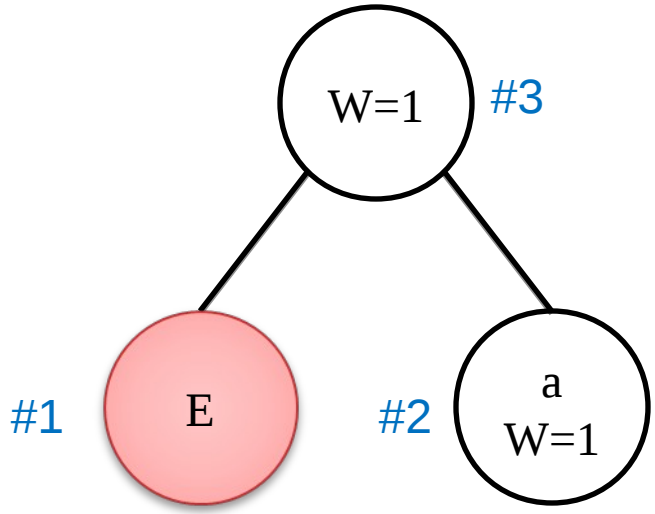
Ta thêm “b” vào thành node lá trên cây



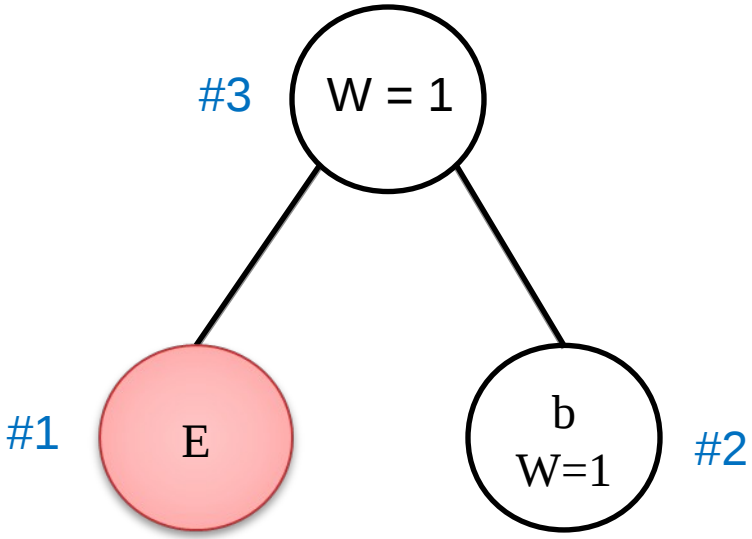
input = 011000010 01100010

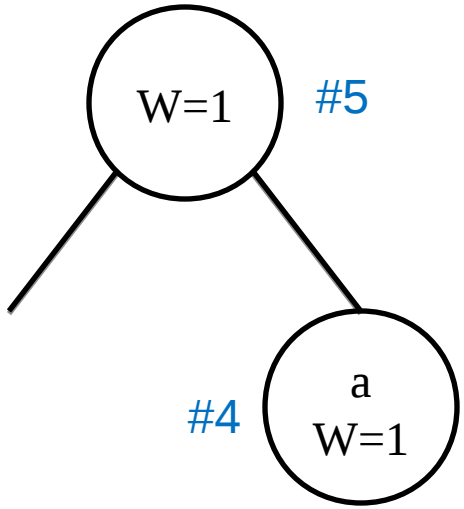
Adaptive Huffman – Ví dụ (tt)





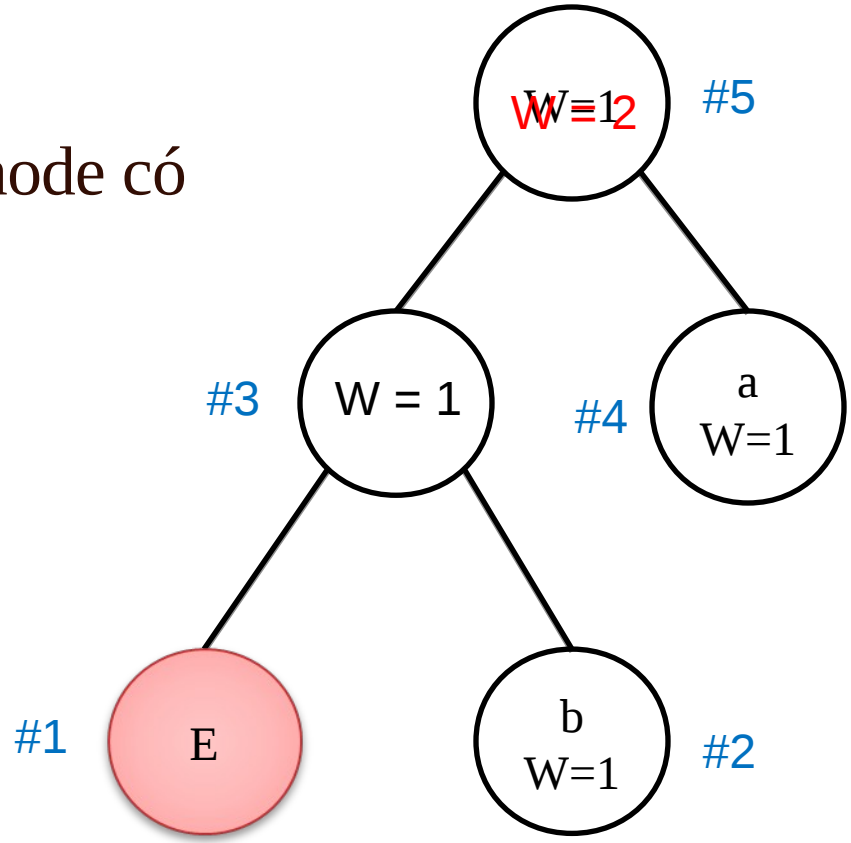
Adaptive Huffman – Ví dụ (tt)





Adaptive Huffman – Ví dụ (tt)

Cập nhật trọng số của các node có liên quan



Adaptive Huffman – Ví dụ (tt)

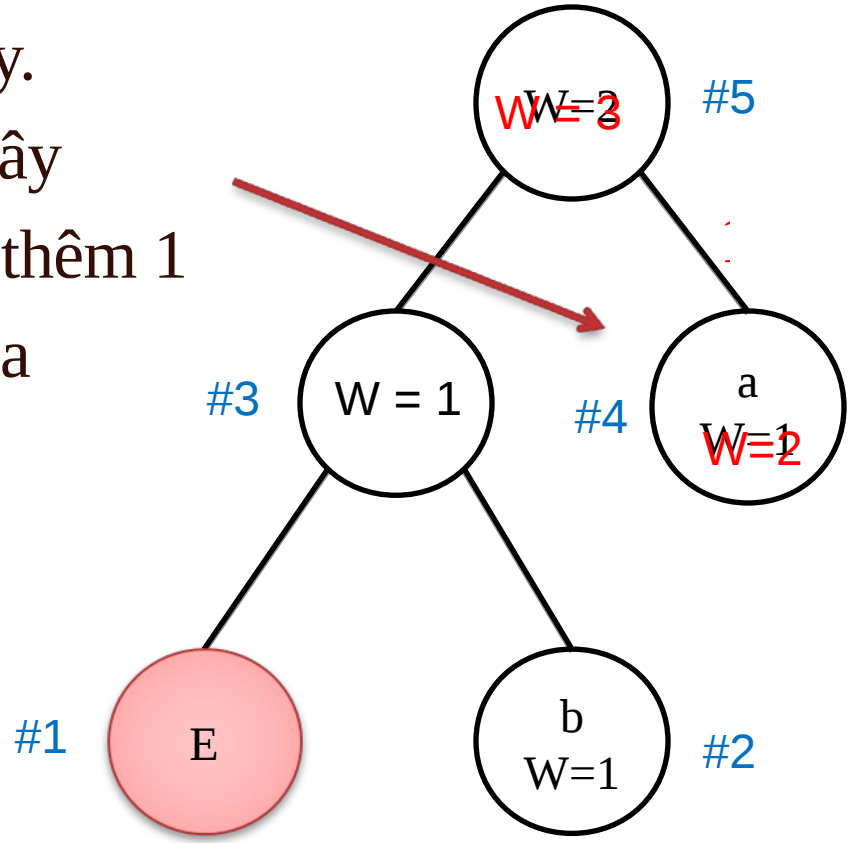
a

B3: thêm kí tự “a” vào cây.

“a” đã tồn tại trong cây

Tăng trọng số của node a thêm 1

Cập nhật lại trọng số của các node liên quan



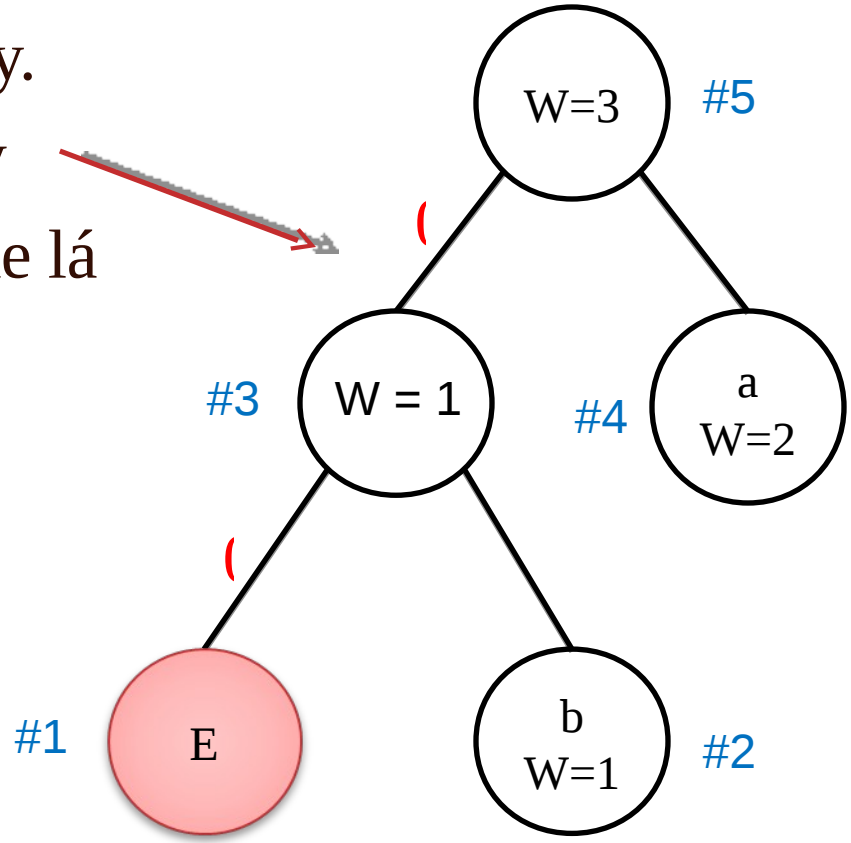
input = 011000010011000101

Adaptive Huffman – Ví dụ (tt)

B4: thêm kí tự “c” vào cây.

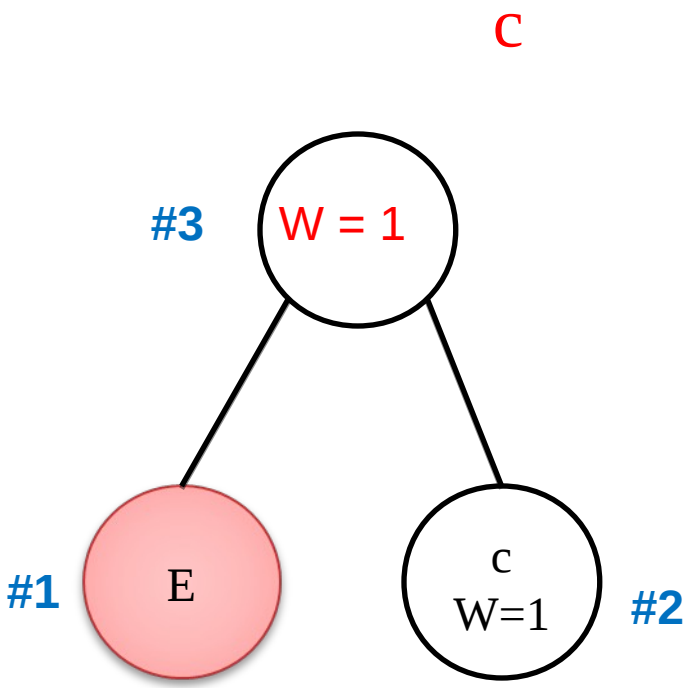
“c” chưa có trong cây

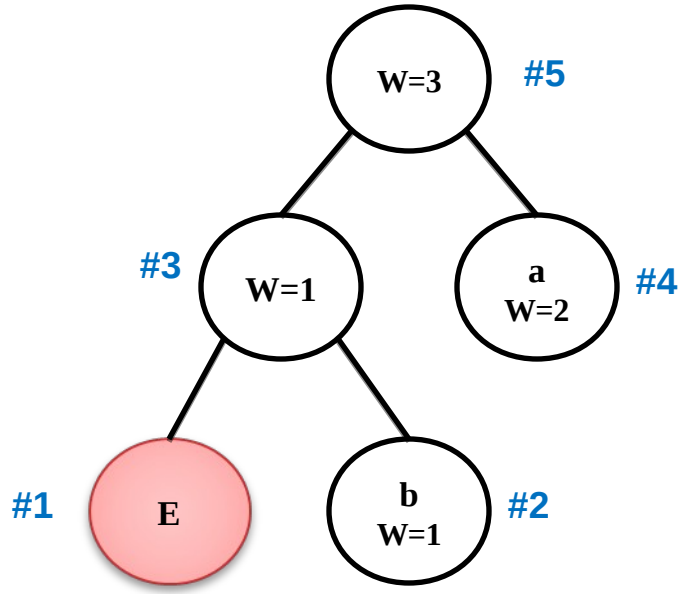
Ta thêm “c” vào thành node lá trên cây



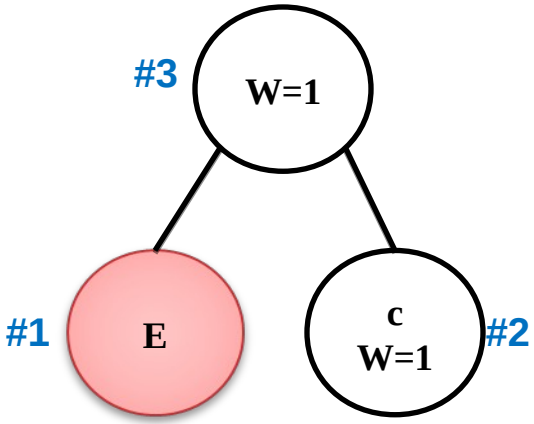
input = 01100001001100010100 01100010

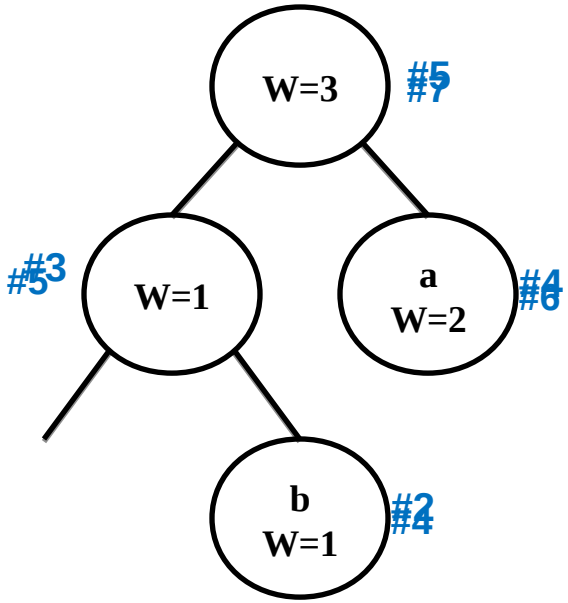
Adaptive Huffman – Ví dụ (tt)





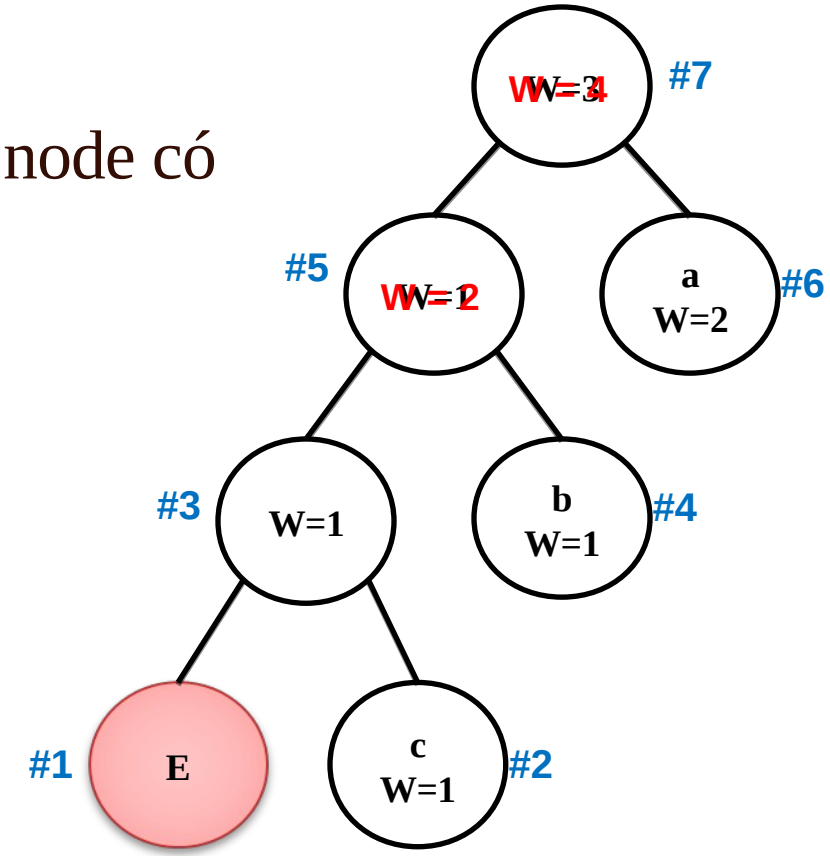
Adaptive Huffman – Ví dụ (tt)





Adaptive Huffman – Ví dụ (tt)

Cập nhật trọng số của các node có liên quan



Adaptive Huffman – Ví dụ (tt)

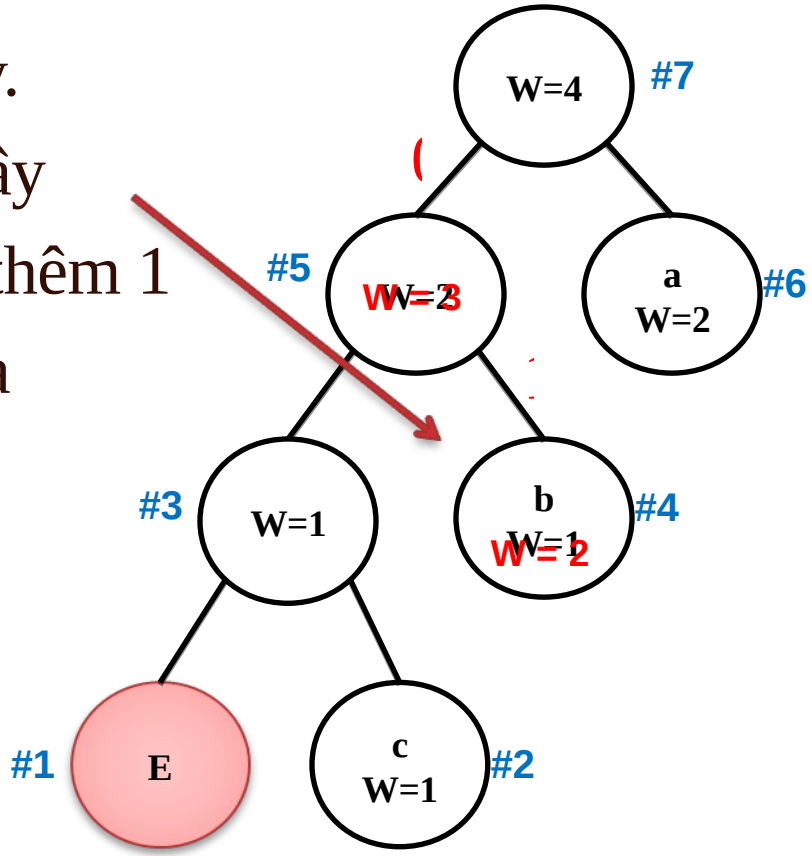
b

B5: thêm kí tự “b” vào cây.

“b” đã tồn tại trong cây

Tăng trọng số của node b thêm 1

Cập nhật lại trọng số của các node liên quan



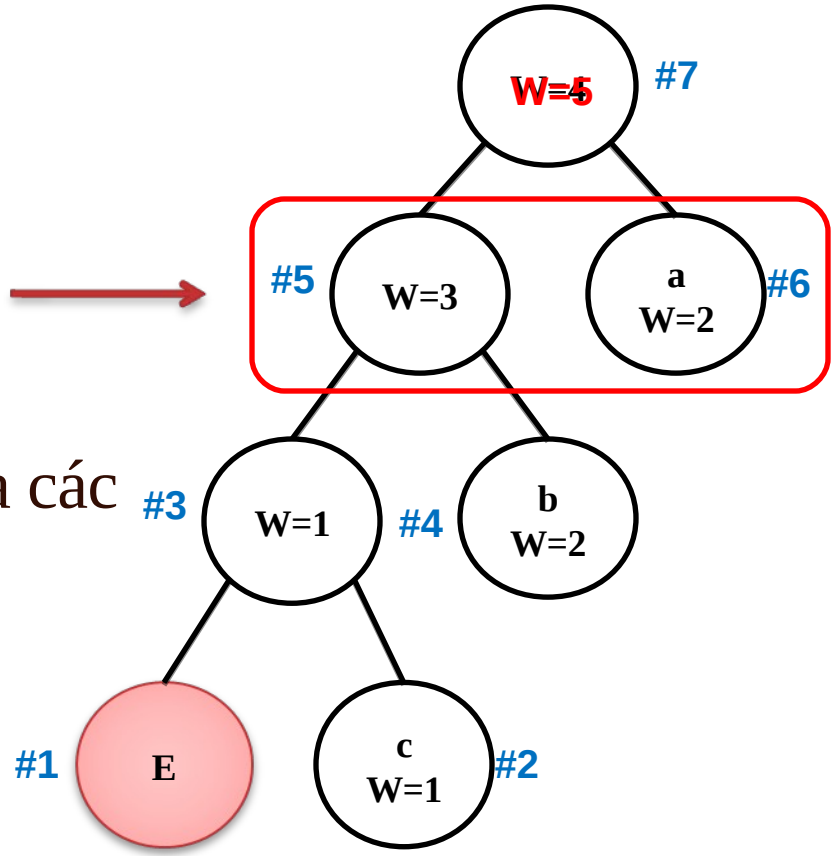
input = 01100001001100010100011000101

Adaptive Huffman – Ví dụ (tt)

Vi phạm tính chất anh em

Ta phải xoay lại cây

Cập nhật lại trọng số của các Node liên quan.

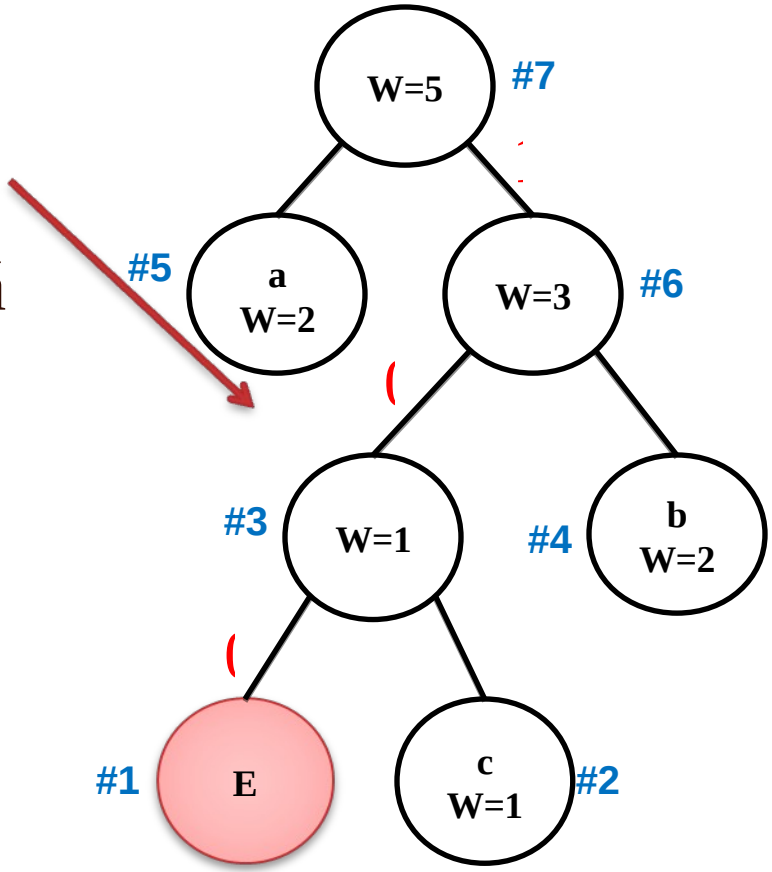


Adaptive Huffman – Ví dụ (tt)

B6: thêm kí tự “d” vào cây.

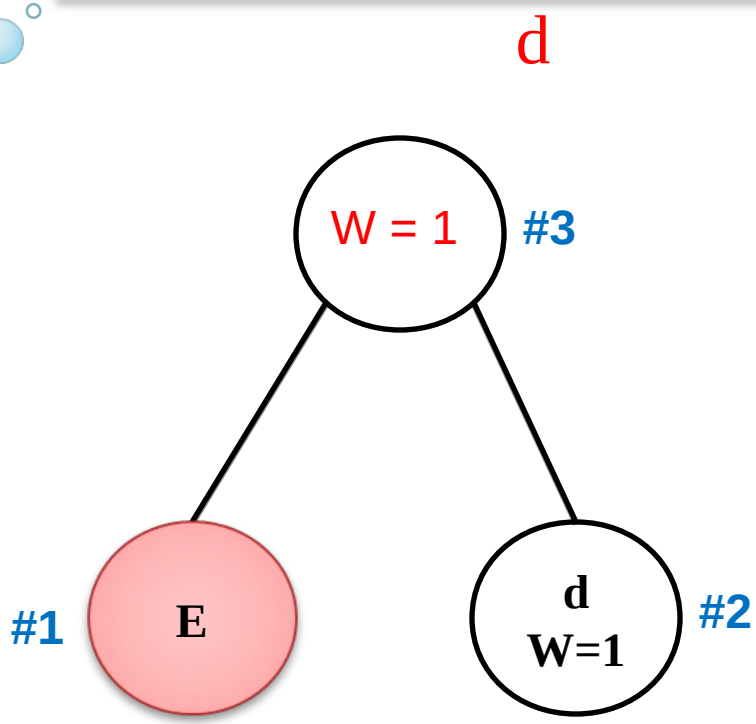
“d” chưa có trong cây

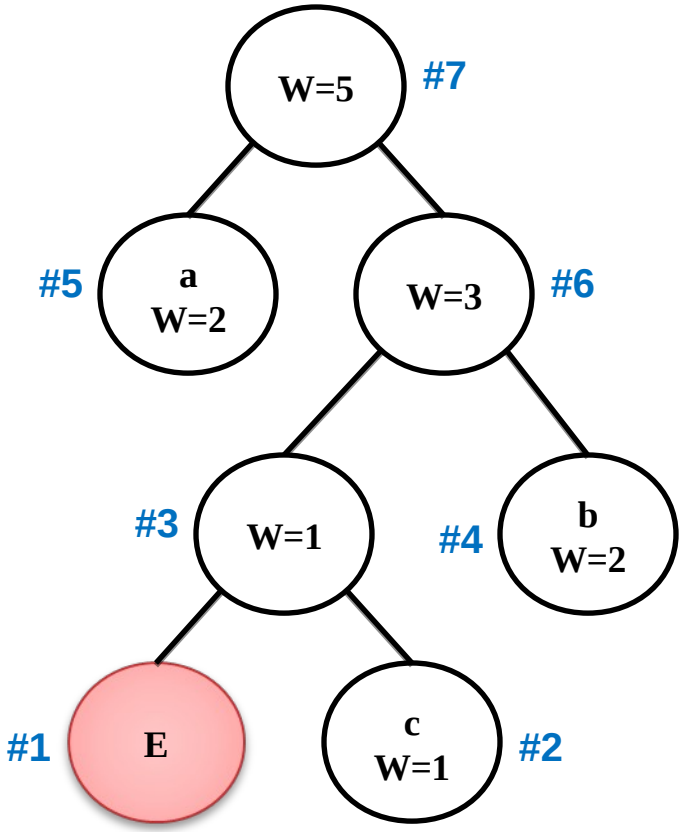
Ta thêm “d” vào thành node lá trên cây



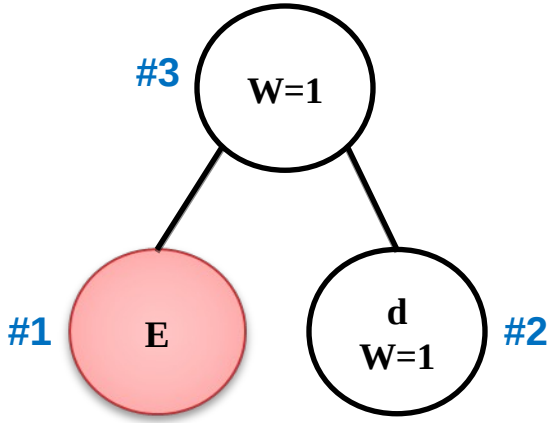
input = 01100001001100010100011000100110001100100

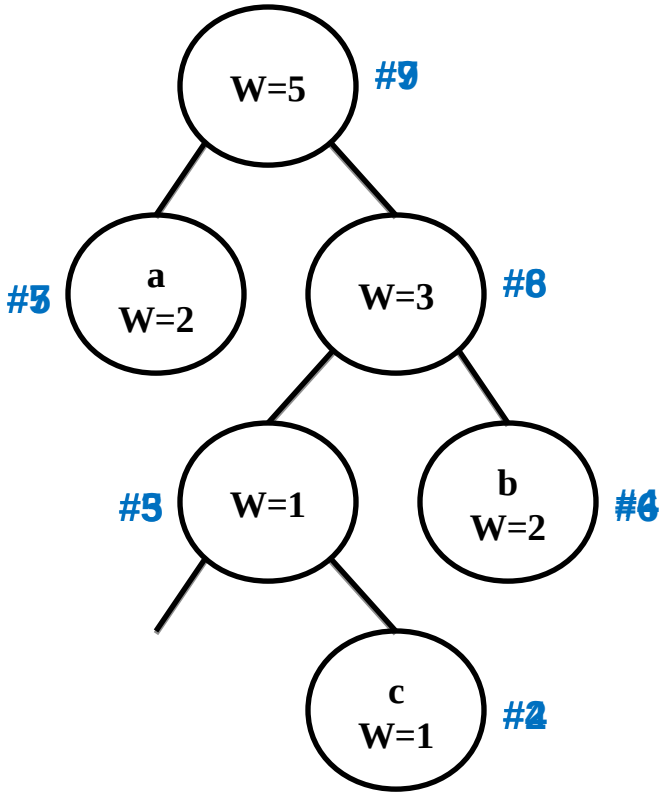
Adaptive Huffman – Ví dụ (tt)





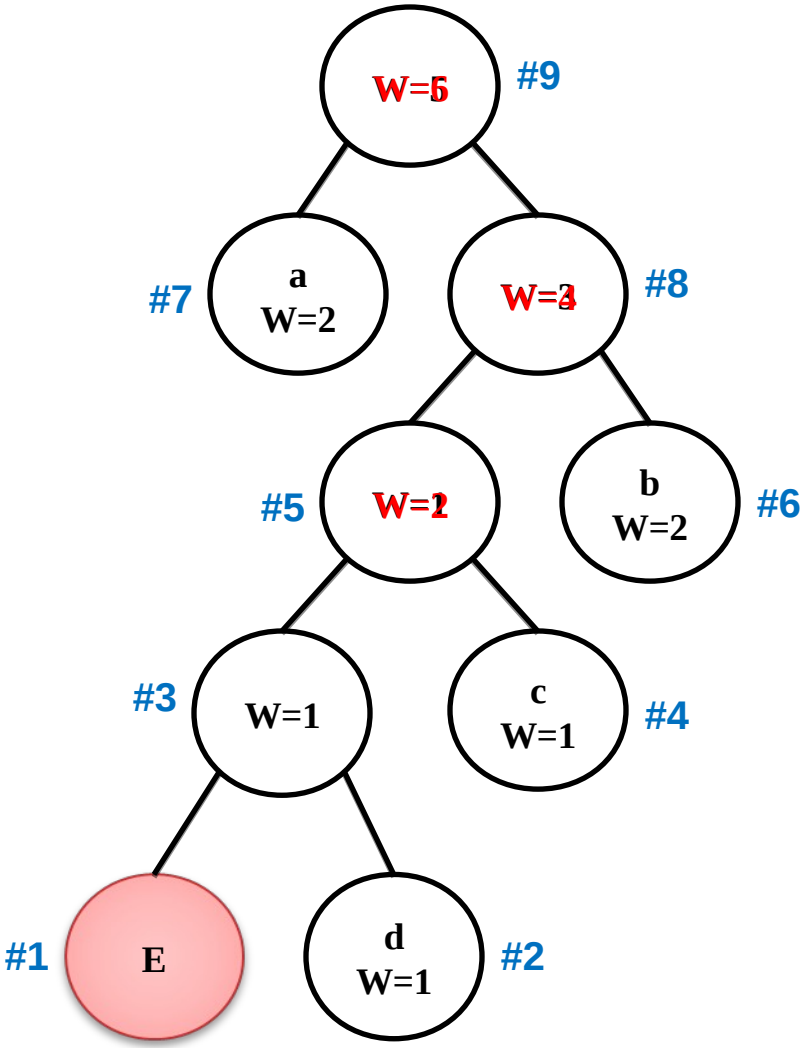
Adaptive Huffman – Ví dụ (tt)





Adaptive Huffman – Ví dụ (tt)

Cập nhật lại trọng số của các
node có liên quan

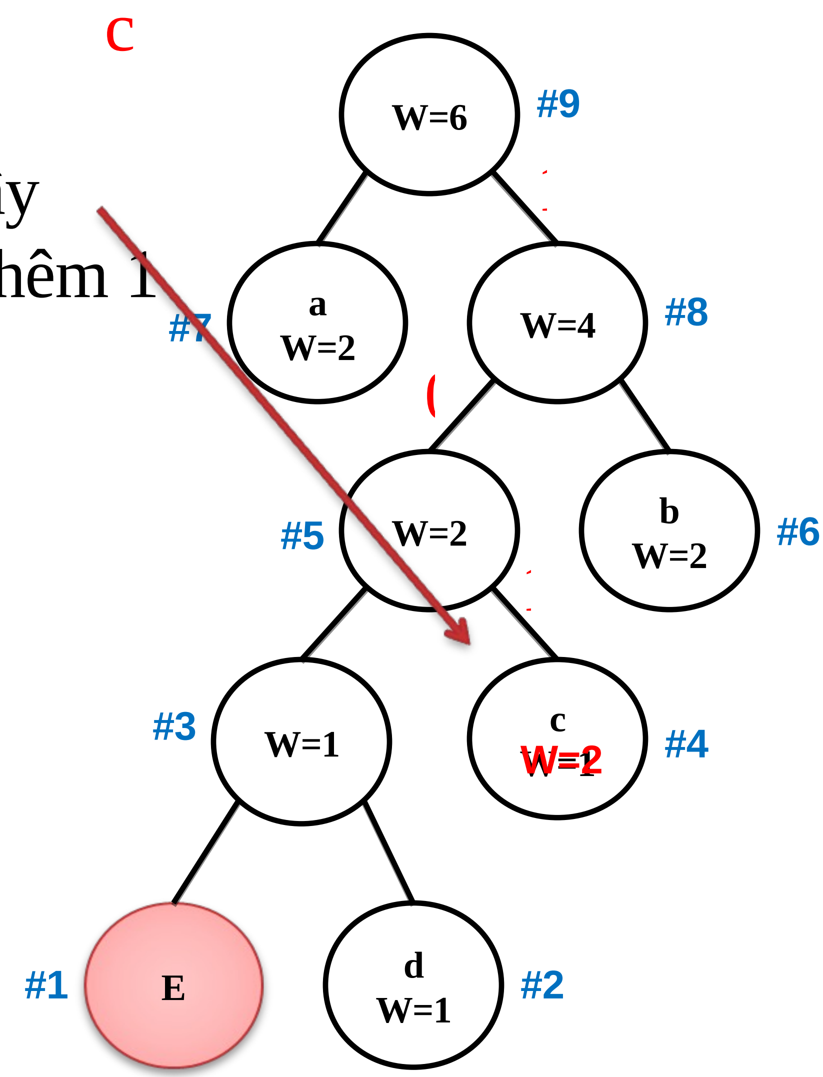


Adaptive Huffman – Ví dụ (tt)

B7: thêm kí tự “c” vào cây.

“c” đã tồn tại trong cây

Tăng trọng số của node c thêm 1



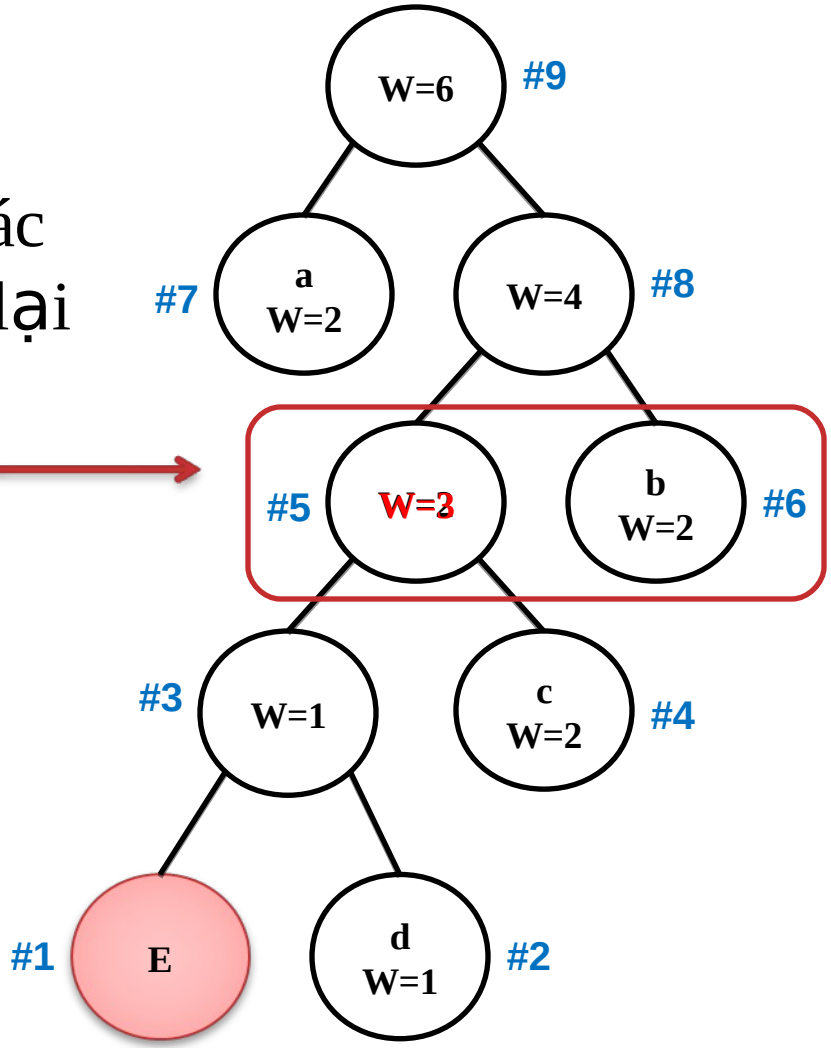
input = 0110000100110001010001100010011000110010010101

Adaptive Huffman – Ví dụ (tt)

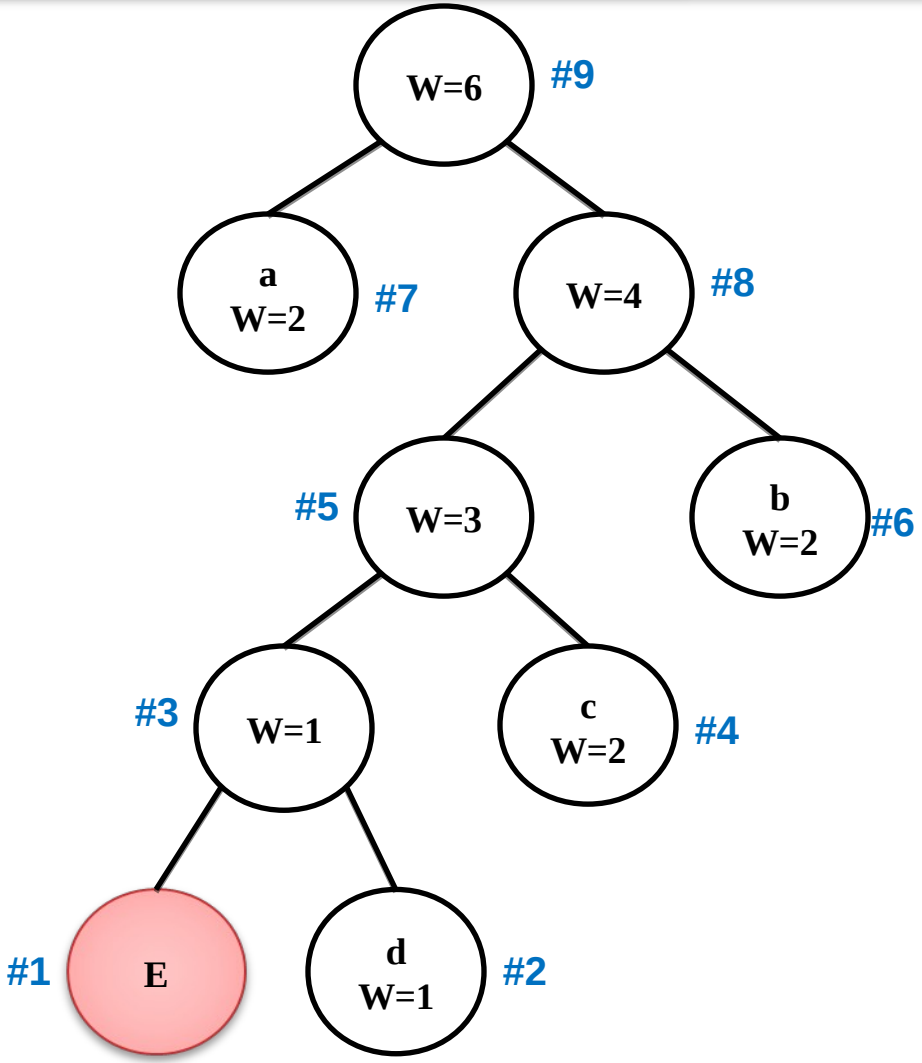
Cập nhật lại trọng số của các node liên quan và hiệu chỉnh lại cây.

Vi phạm tính chất anh em

Ta phải xoay lại cây



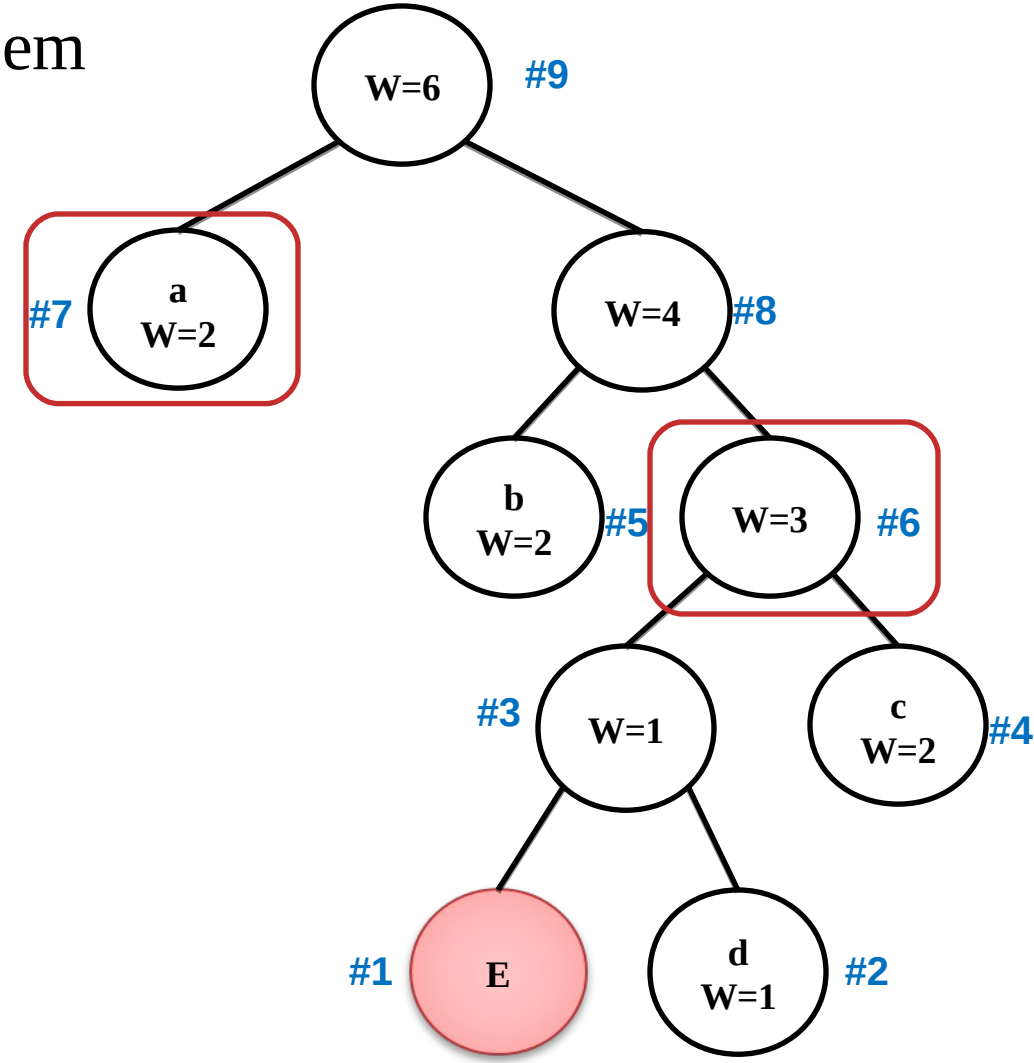
Adaptive Huffman – Ví dụ (tt)



Adaptive Huffman – Ví dụ (tt)

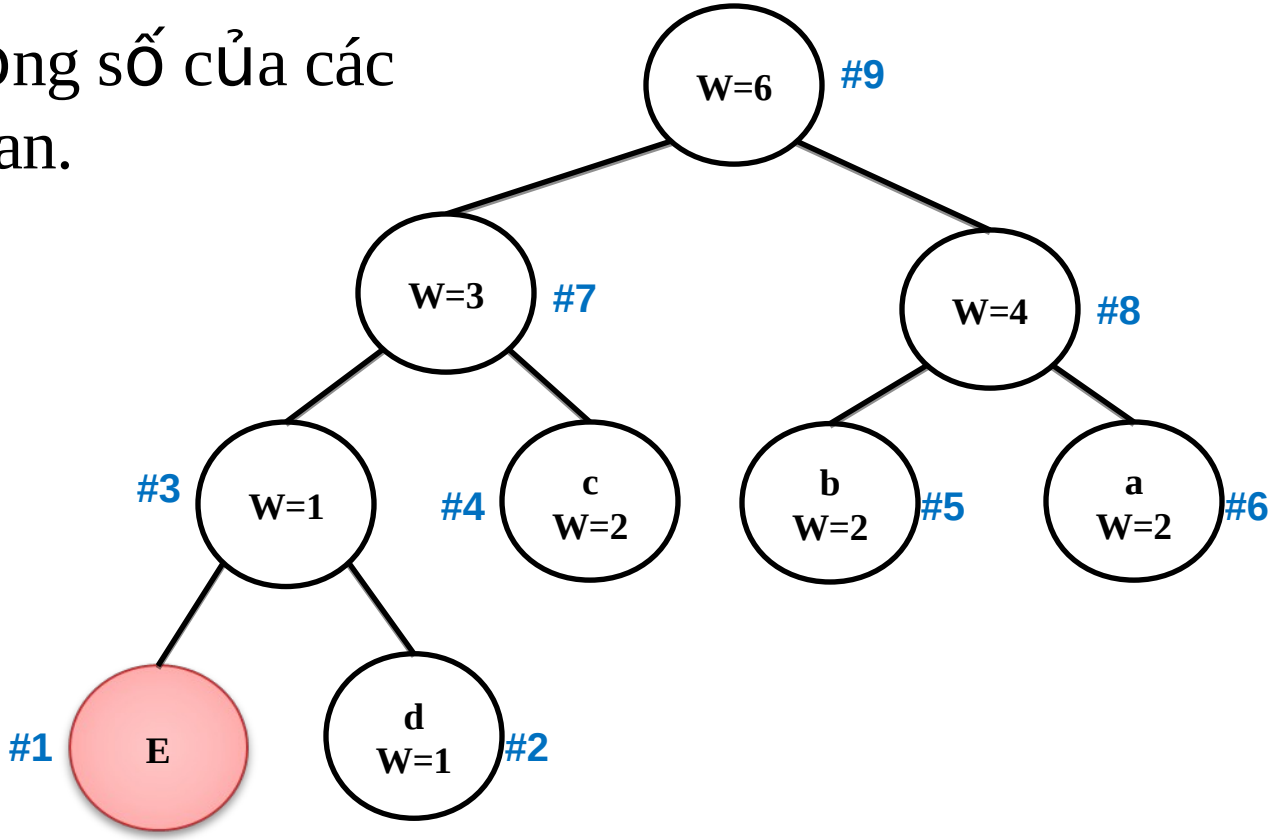
- Vi phạm tính chất anh em

Ta hiệu chỉnh lại cây



Adaptive Huffman – Ví dụ (tt)

kiểm tra trọng số của các node liên quan.



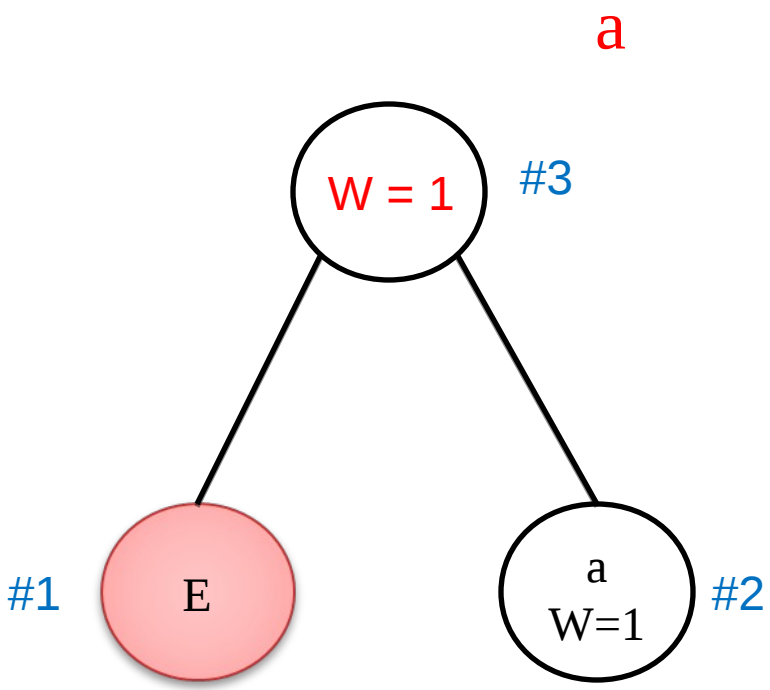
input = 01100001001100010100011000100110001100100101

Adaptive Huffman

Giải nén

Adaptive Huffman – Ví dụ (tt)

- B1: Đọc 8 bit đầu tiên từ file input



output = a

input = 01100001001100010100011000100110001100100101

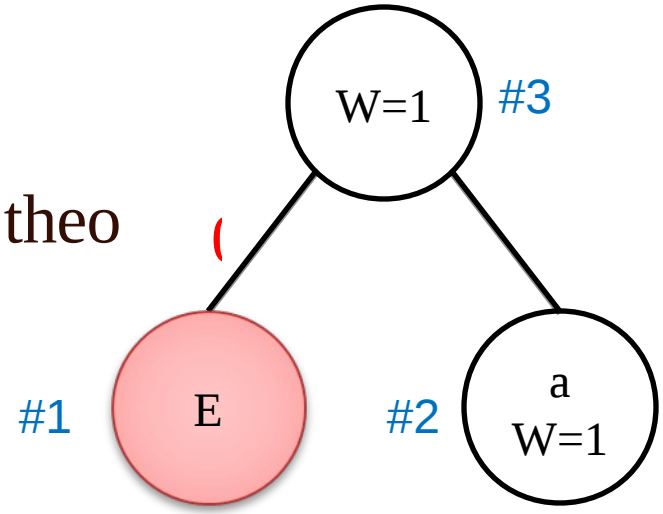
a

Adaptive Huffman – Ví dụ (tt)

B2: Đọc bit tiếp theo, duyệt cây

Cập nhật kí tự ‘b’ vào trong cây

Di chuyển đầu đọc đến vị trí tiếp theo



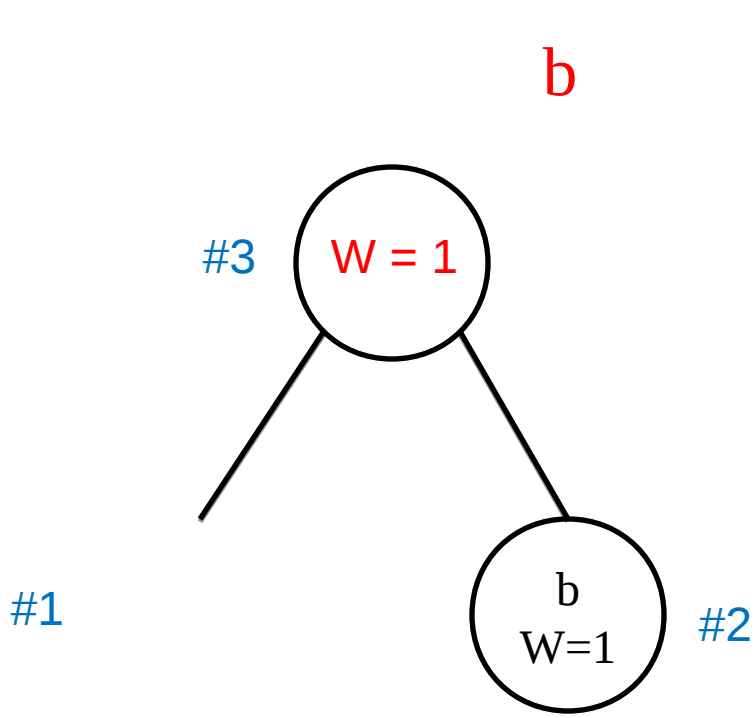
output = a l

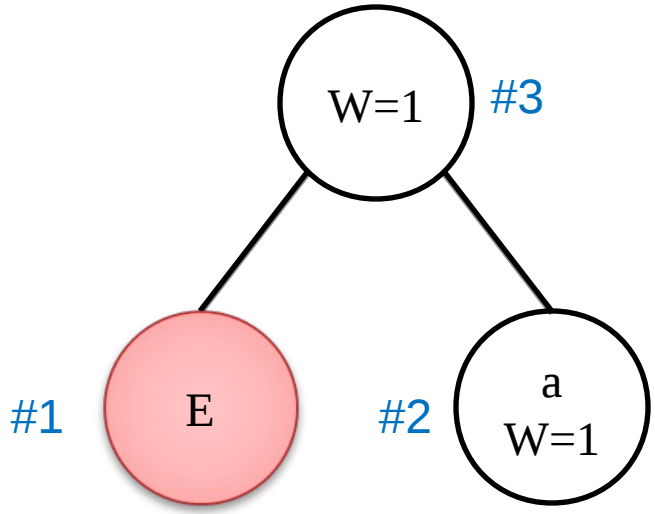
fseek

input = 01100001001100010100011000100110001100100101

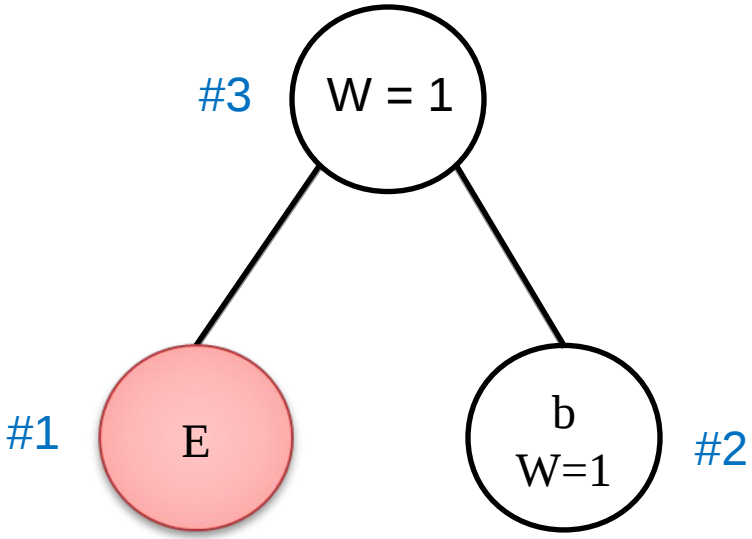
b

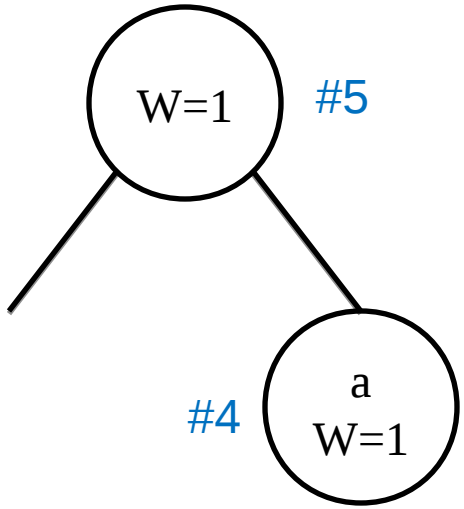
Adaptive Huffman – Ví dụ (tt)





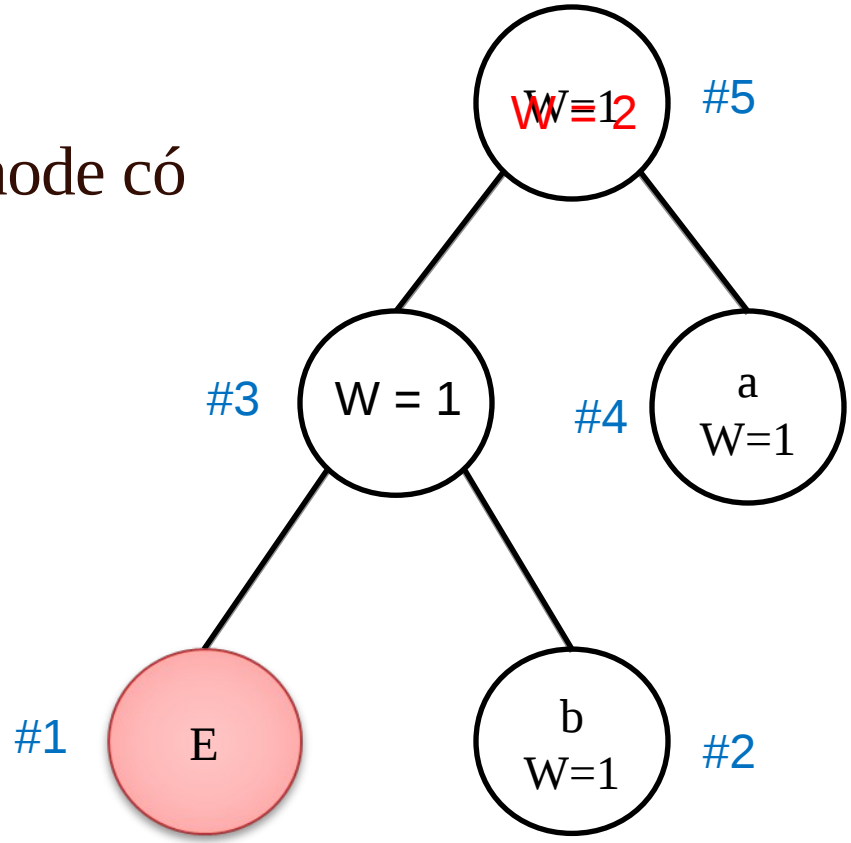
Adaptive Huffman – Ví dụ (tt)





Adaptive Huffman – Ví dụ (tt)

Cập nhật trọng số của các node có liên quan



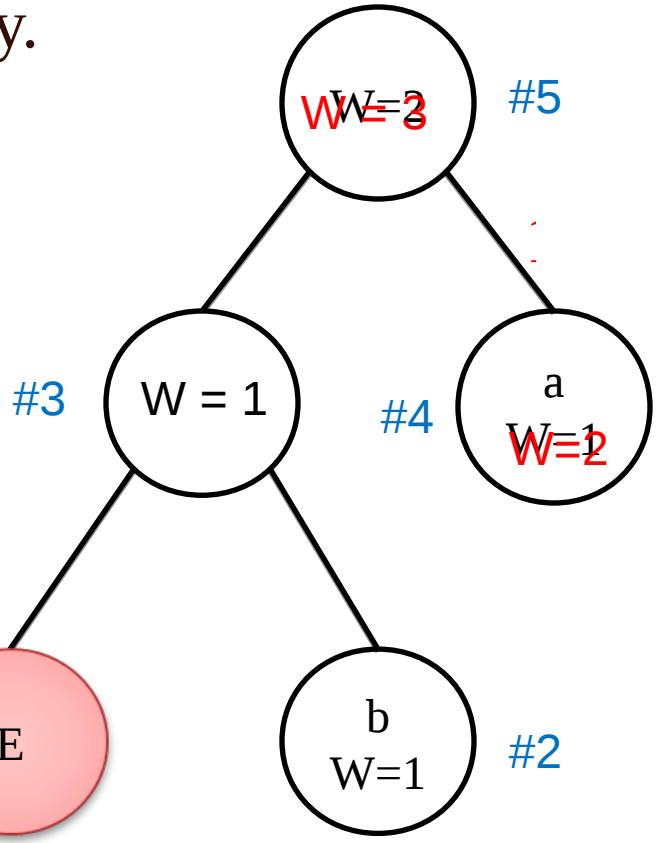
Adaptive Huffman – Ví dụ (tt)

B3:Đọc bit tiếp theo và duyệt cây.

Tăng trọng số của node a thêm 1

Cập nhật lại trọng số của các node liên quan

Di chuyển đầu đọc đến vị trí tiếp theo



output = ab

fseek

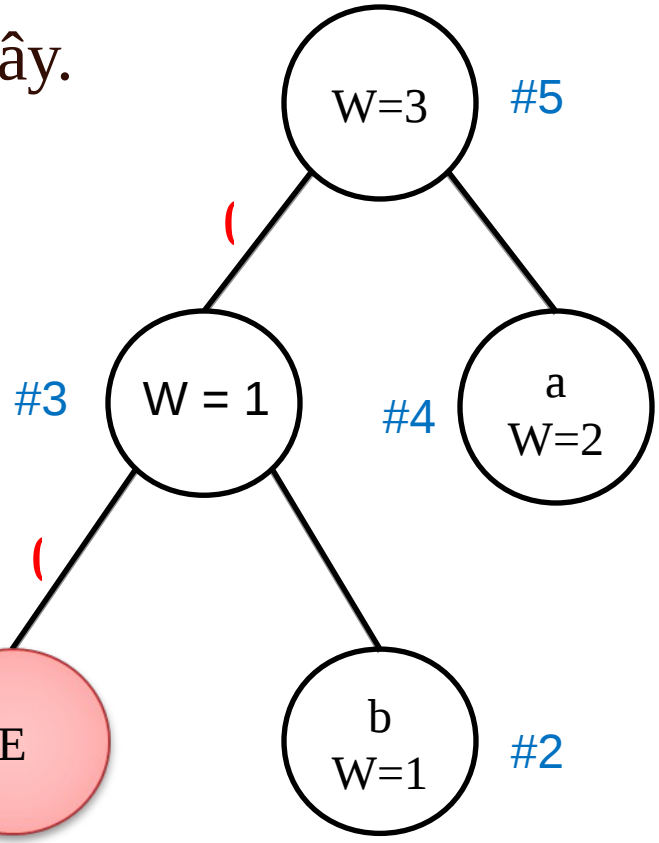
input = 01100001001100010100011000100110001100100101

Adaptive Huffman – Ví dụ (tt)

B4: Đọc bit tiếp theo và duyệt cây.

Ta thêm "c" vào thành node lá trên cây

Di chuyển đầu đọc đến vị trí tiếp theo

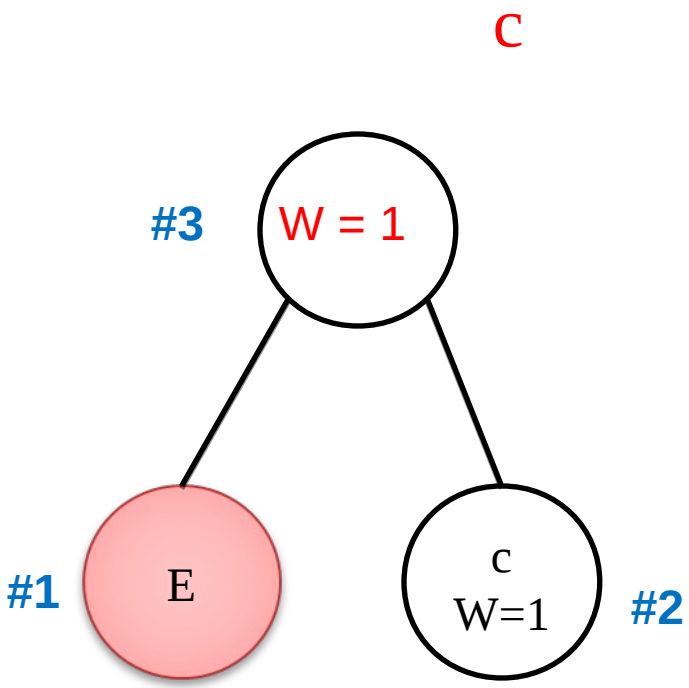


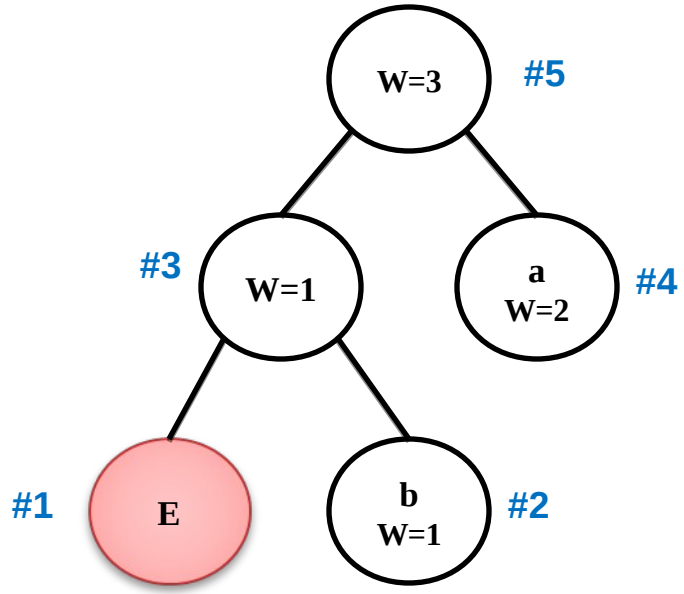
output = aba (

fseek

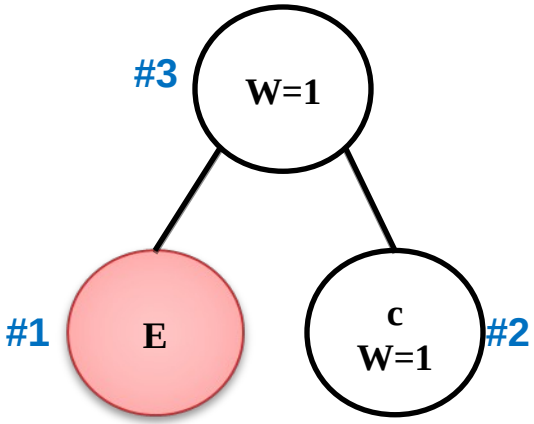
input = 01100001001100010100011000100110001100100101

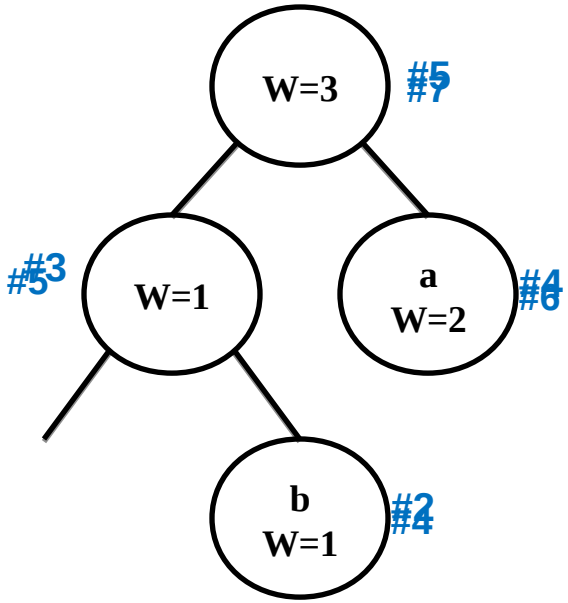
Adaptive Huffman – Ví dụ (tt)





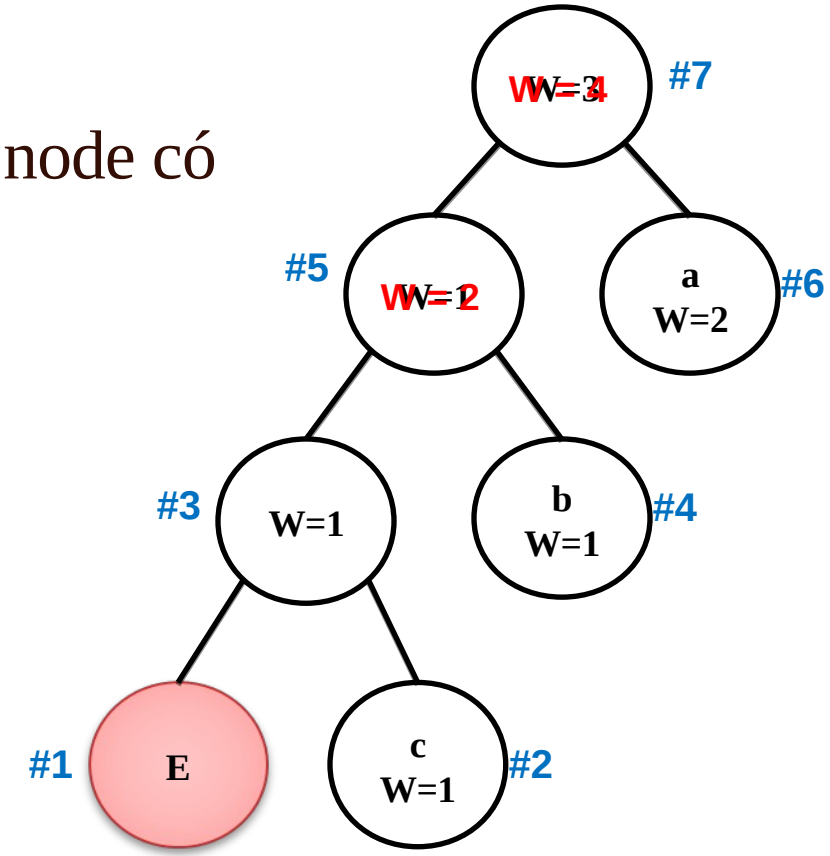
Adaptive Huffman – Ví dụ (tt)





Adaptive Huffman – Ví dụ (tt)

Cập nhật trọng số của các node có liên quan



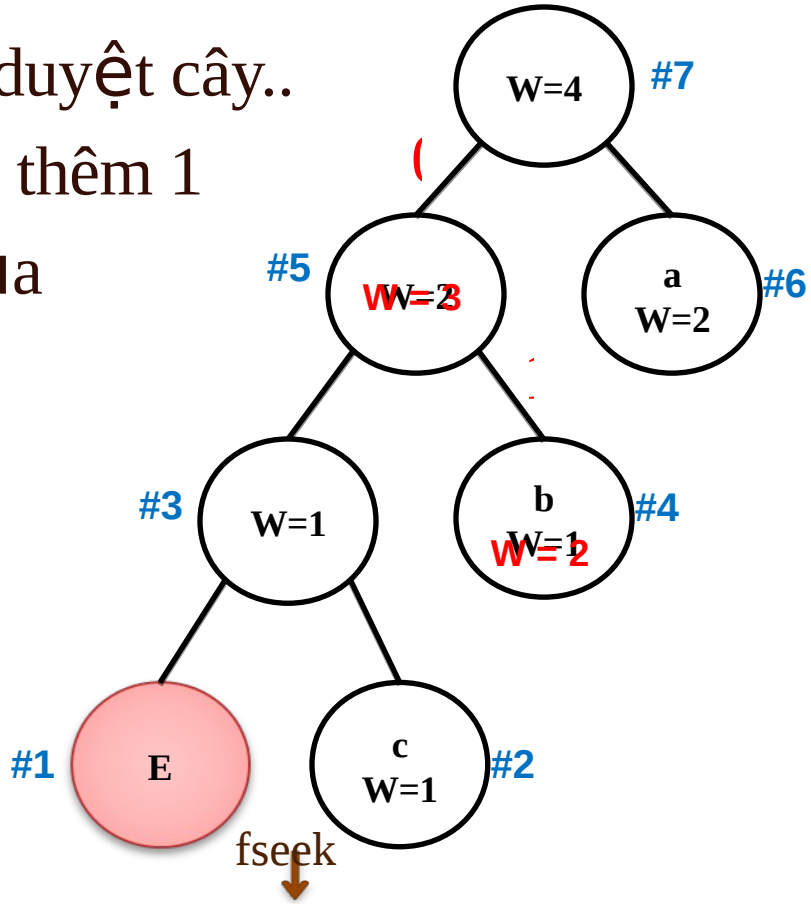
Adaptive Huffman – Ví dụ (tt)

B5: Đọc bit tiếp theo và duyệt cây..

Tăng trọng số của node b thêm 1

Cập nhật lại trọng số của các node liên quan

Di chuyển đầu đọc đến vị trí tiếp theo



output = abac l

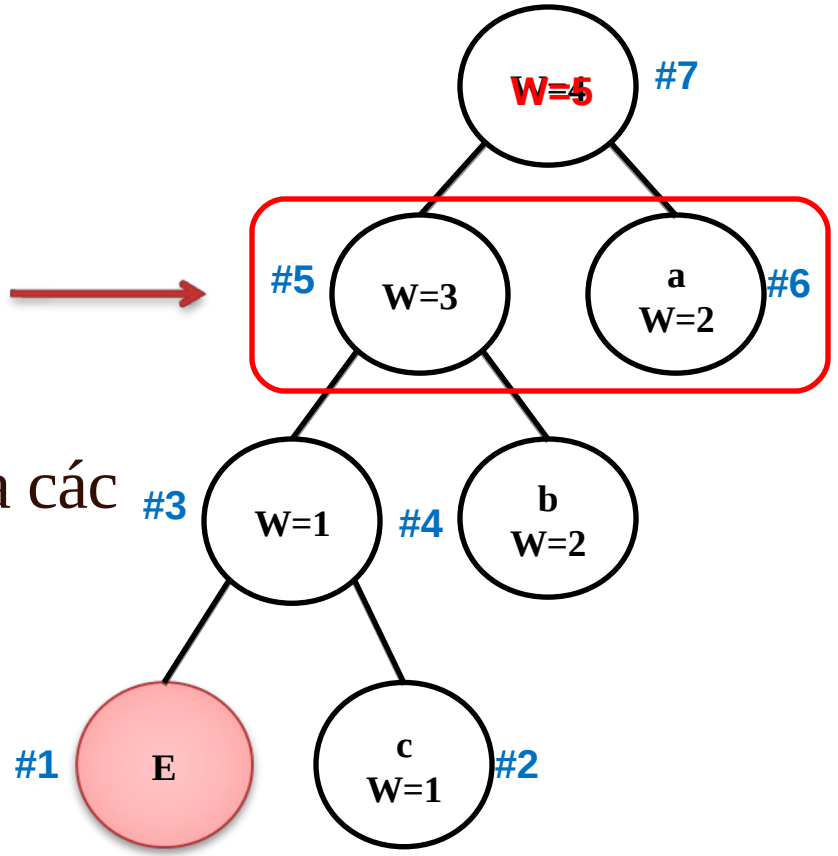
input = 0110000100110001010001100010011000110010010101

Adaptive Huffman – Ví dụ (tt)

Vi phạm tính chất anh em

Ta phải xoay lại cây

Cập nhật lại trọng số của các Node liên quan.

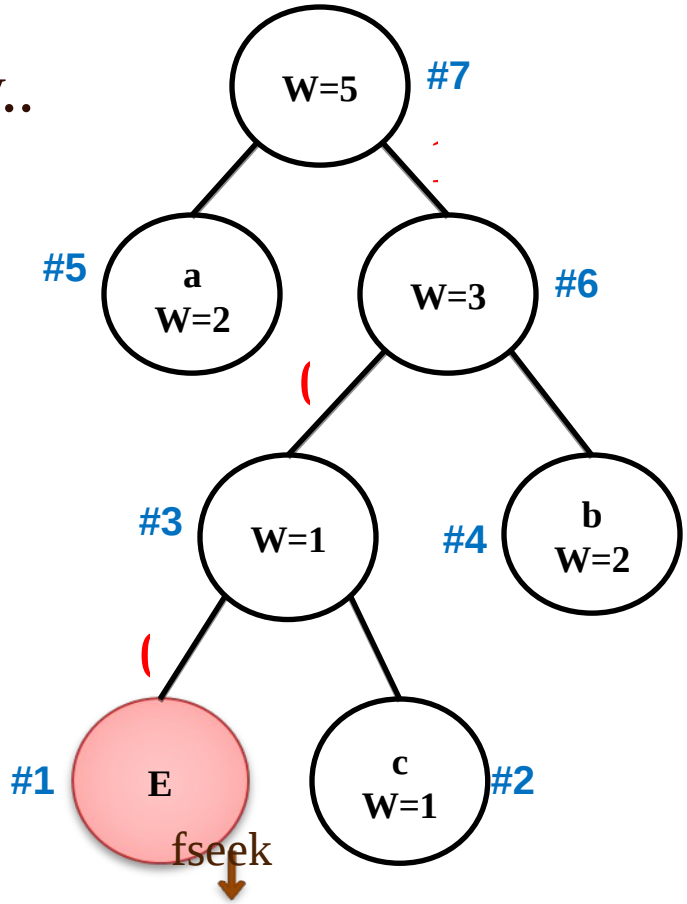


Adaptive Huffman – Ví dụ (tt)

B6: Đọc bit tiếp theo và duyệt cây..

Ta thêm "d" vào thành node lá trên cây

Di chuyển đầu đọc đến vị trí tiếp theo

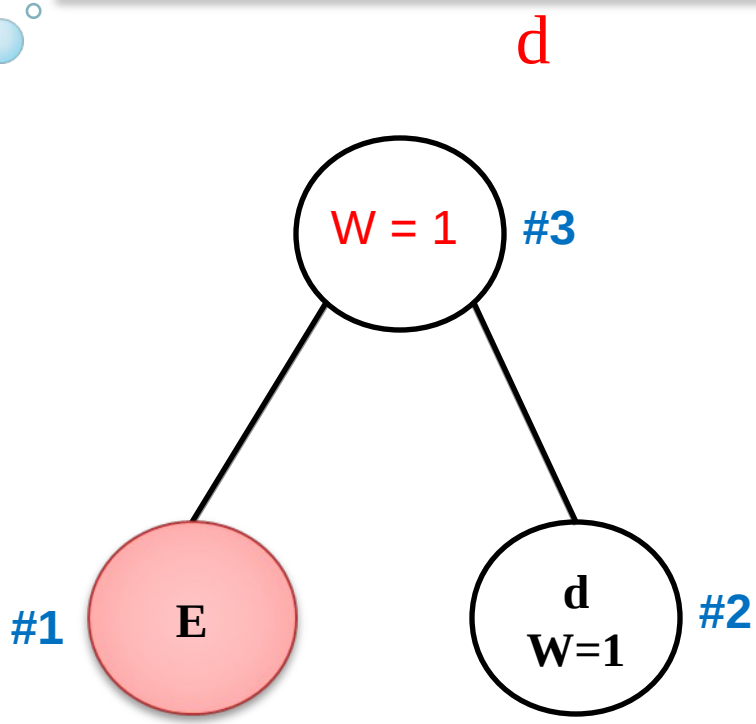


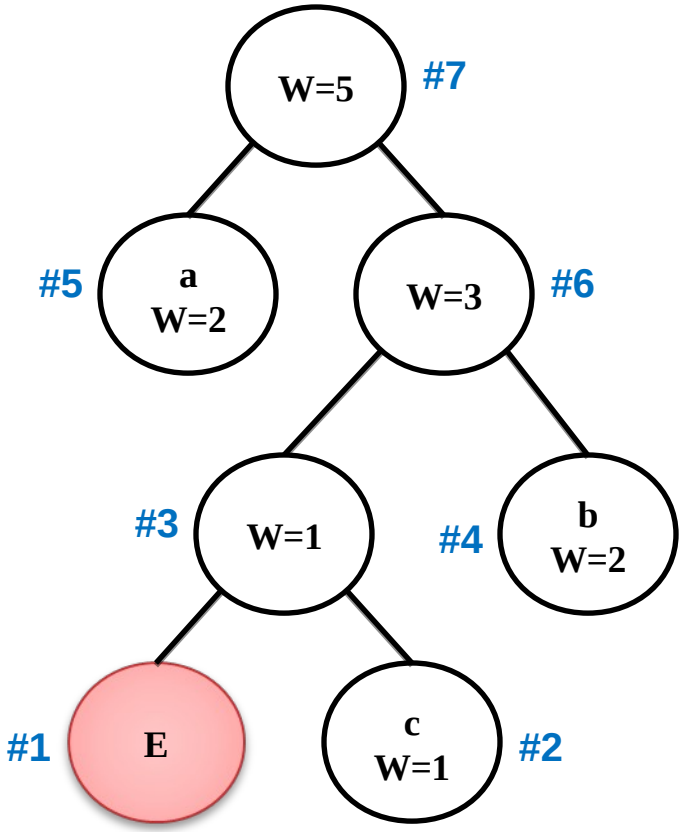
output = abacb (

input = 01100001001100010100011000100110001100100101

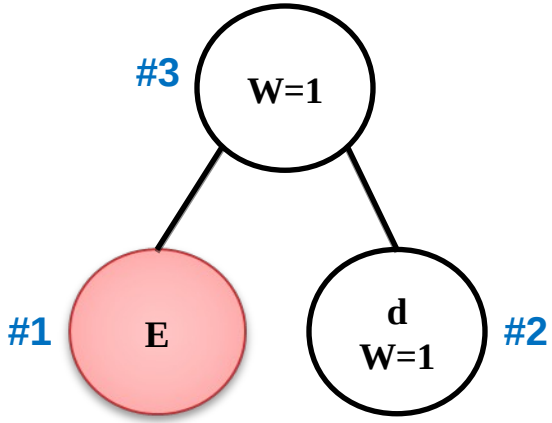
d

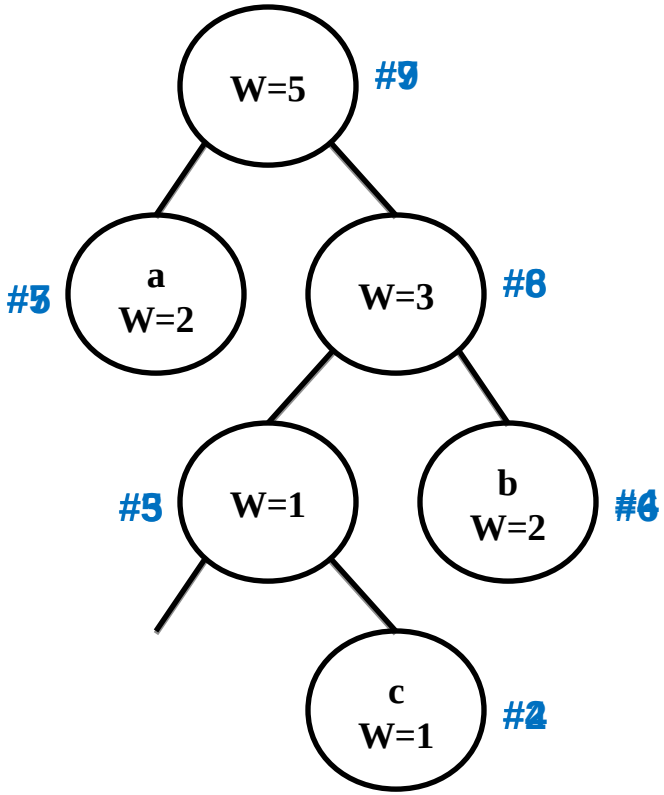
Adaptive Huffman – Ví dụ (tt)





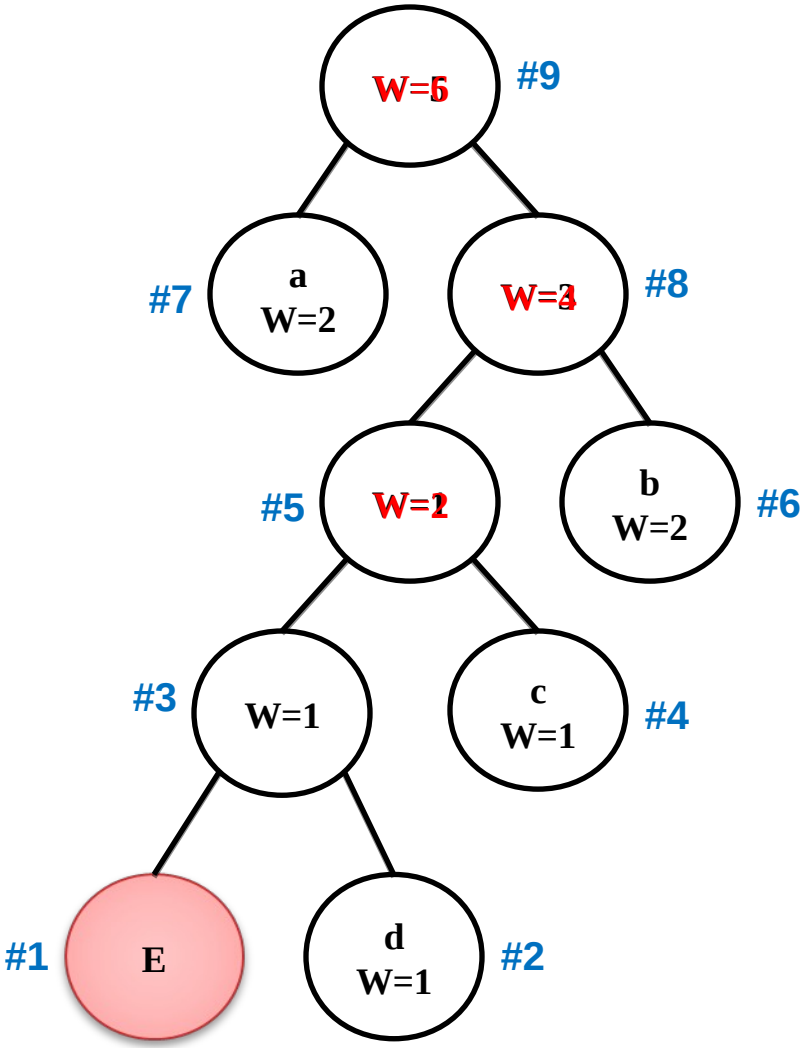
Adaptive Huffman – Ví dụ (tt)





Adaptive Huffman – Ví dụ (tt)

Cập nhật lại trọng số của các
node có liên quan



Adaptive Huffman – Ví dụ (tt)

B7: Đọc bit tiếp theo và duyệt cây...
Tăng trọng số của node c thêm 1

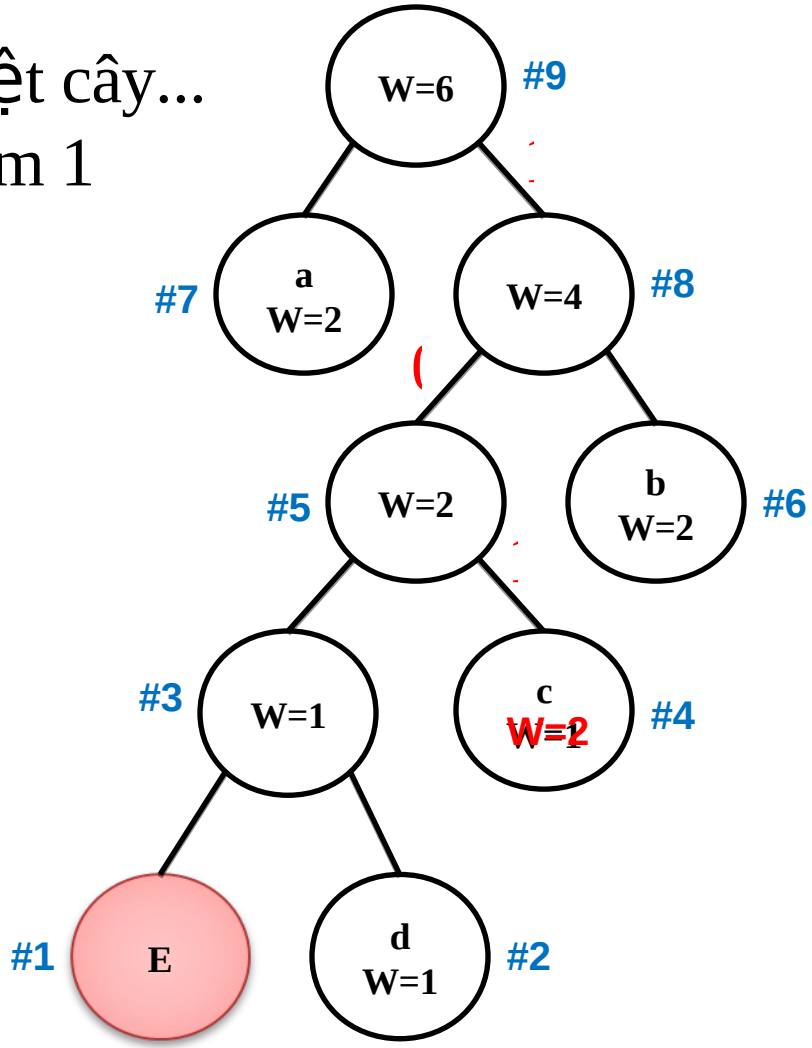
output = abacbd

input = 01100001001100010

1000110001001100

01100100101

fseek

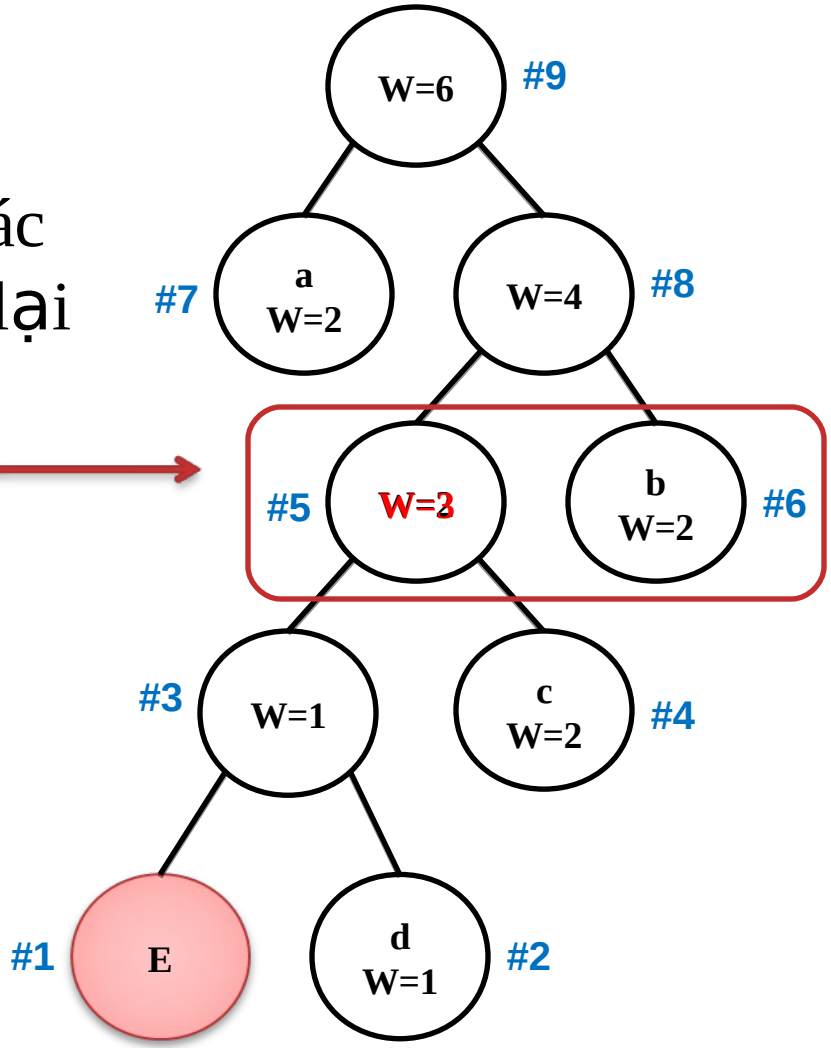


Adaptive Huffman – Ví dụ (tt)

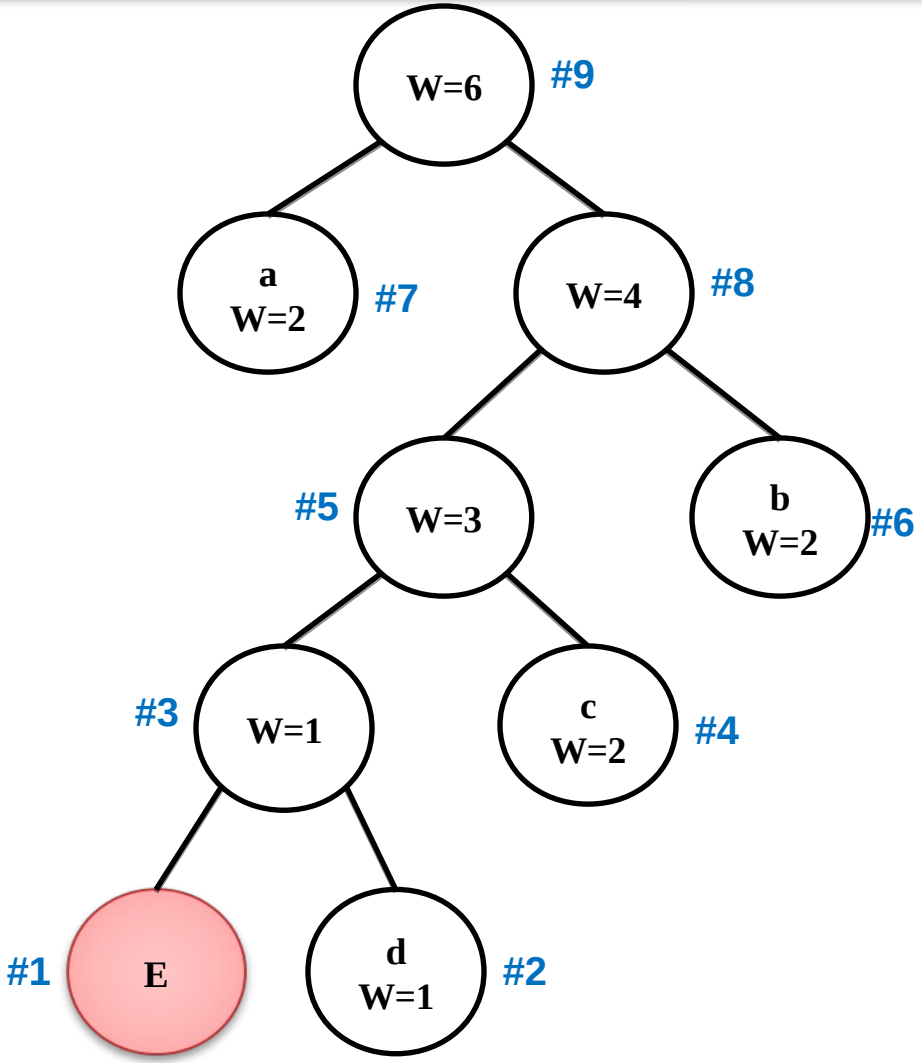
Cập nhật lại trọng số của các node liên quan và hiệu chỉnh lại cây.

Vi phạm tính chất anh em

Ta phải xoay lại cây



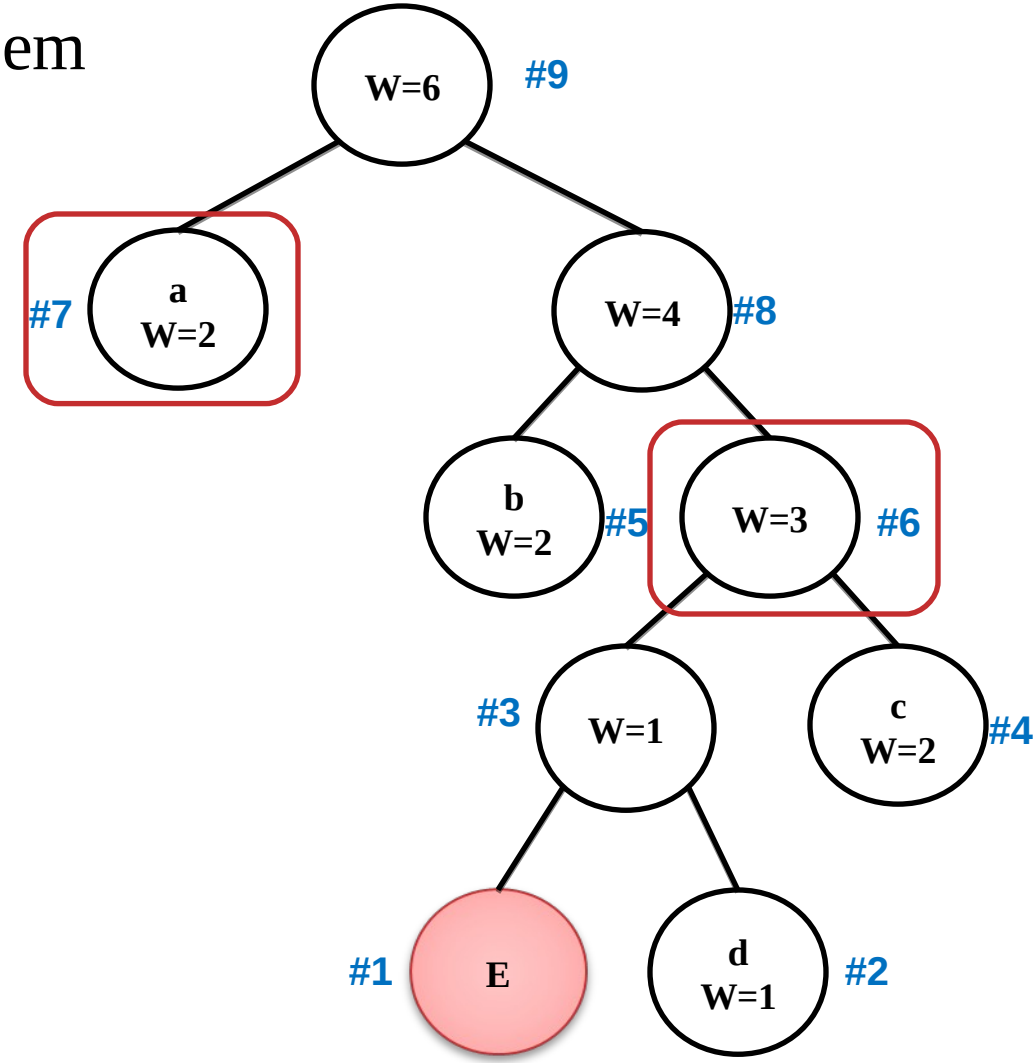
Adaptive Huffman – Ví dụ (tt)



Adaptive Huffman – Ví dụ (tt)

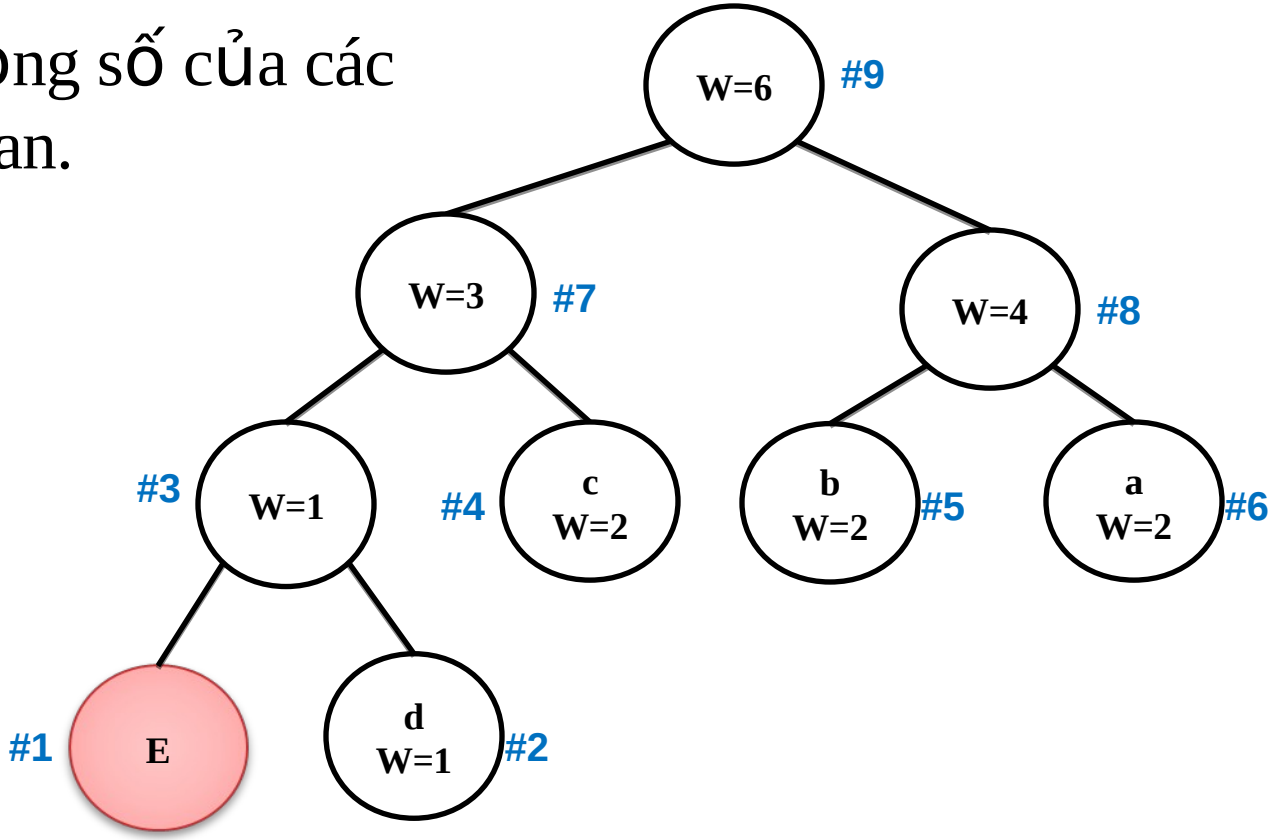
- Vi phạm tính chất anh em

Ta hiệu chỉnh lại cây



Adaptive Huffman – Ví dụ (tt)

kiểm tra trọng số của các node liên quan.



output = abacbdc

Adaptive Huffman