CHƯƠNG 6: MÃ HÓA NGUỒN

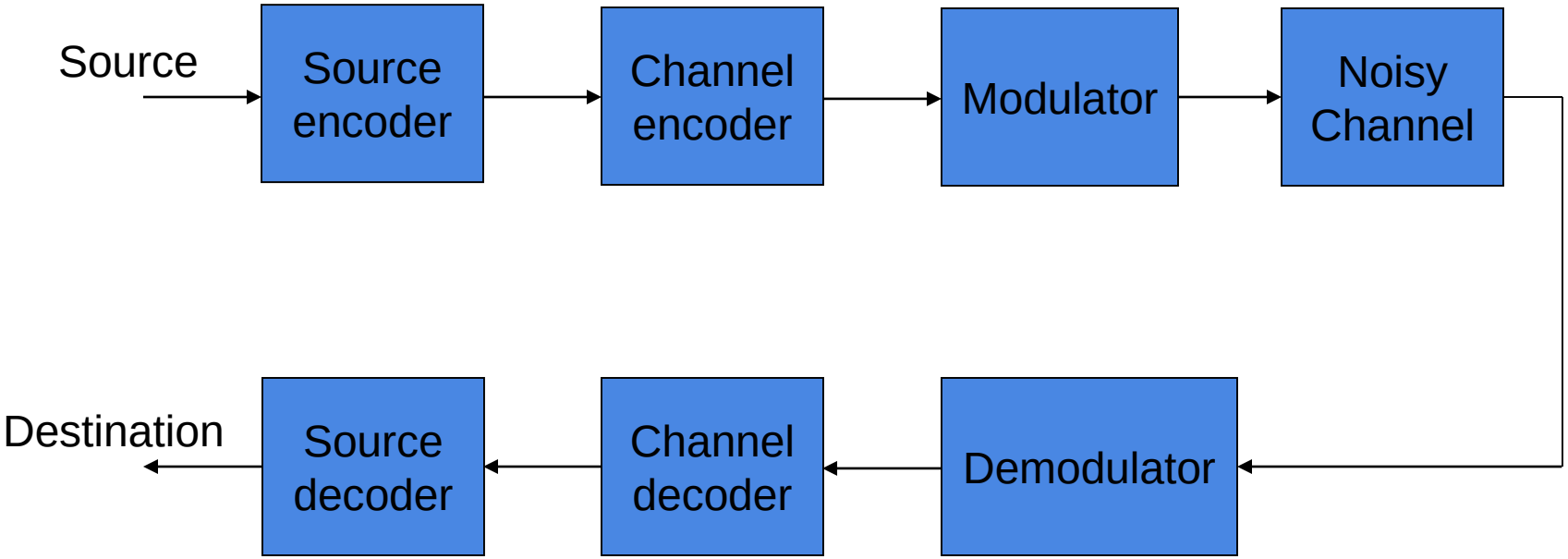
Đặng Lê Khoa

Email: dlkhoa@fetel.hcmus.edu.vn

Facuty of Electronics & Telecommunications, HCMUS

Basic concepts

- Sơ đồ của hệ thống truyền thông số



Information measure

- Lý thuyết thông tin: có bao nhiêu thông tin
  - … chứa trong tín hiệu?
  - … có thể phát ra?
  - … có thể truyền qua một kênh?
- Thông tin được sinh ra từ nguồn chuyển đến người dùng ở đích
- Ví dụ: Pháp vs U23-VN

Information measure

- Xem xét ba kết quả: thắng, hòa, thu

Tình huống	Sự kiện	Thông tin	Xắc suất
1	Pháp thắng	Không thông tin	≈1
2	Pháp hòa U23-VN	Nhiều thông tin	Tương đối thấp
3	Pháp thua	Rất nhiều thông	Xắc suất rất thấp, gần như không xảy ra

- Thông tin ít có khả năng, mang nhiều thông tin
- Làm thế nào xác định thông tin bằng toán học?

Information

- Đặt x_j là sự kiện, $p(x_j)$ là xác suất của sự kiện x_j được truyền
- Nếu x_j xảy ra, chúng ta

$$I(x_j) = \log_a \frac{1}{p(x_j)} = -\log_a p(x_j) \qquad (1)$$

đơn vị của thông tin

  - $I(x_j)$ được gọi self-information (tự thông tin)
    - $I(x_j) \geq 0$ for $0 \leq p(x_j) \leq 1$
    - $I(x_j) \to 0$ for $p(x_j) \to 1$
    - $I(x_j) > I(x_i)$ for $p(x_j) < p(x_i)$

Information

- Ví dụ: Một thí nghiệm với 16 tình huống bằng nhau:
  - Thông tin liên quan với mỗi tình huống:
    - $I(x_j)$=-$\log_2(1/16)$=$\log_2 16$=4 bits
  - Thông tin hơn một bit, do xác suất xảy ra nhỏ hơn ½.

Entropy and Information rate

- Xem xét một nguồn tin phát ra choỗi ký hiệu từ tập $X=\{x_1,x_2..,x_M\}$
- Mỗi ký hiệu x_i có xác suất $p(x_i)$ và self-information là $I(x_i)$
- Nguồn này có tốc độ trung bình r symbols/sec
- Nguồn không có bộ nhớ và phát rời rạc
- Lượng thông tin sinh ra bởi nguồn trong khoảng thời gian của ký hiệu là biến ngẫu nhiên rời rạc X.
- Thông tin trung bình trên mội ký hiệu được cho bởi:

$$H(X) = E\{I(x_j)\} = -\sum_{j=1}^{M} p(x_j)\log_2 p(x_j) \quad \text{bit/symbol} \quad (2)$$

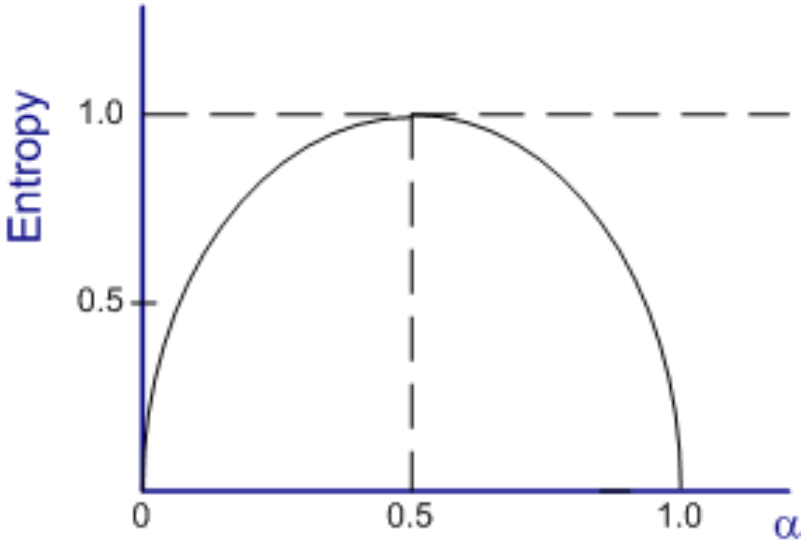
- Entropy = information = uncertainty
- Nếu một tín hiệu hoàn toàn được dự đoán, nó có entropy bằng 0 và không có thông tin
- Entropy = số bit trung bình yêu cầu để truyền tín hiệu

Example

- Biến ngẫu nhiên với phân bố đồng nhất với 32 tình huống
  - $H(X) = -\sum(1/32).log(1/32) = log\ 32 = 5$
  - Số bit yêu cầu = log 32 = 5 bits!
  - Vì vậy H(X) = số bit yêu cầu để biểu diễn một sự ngẫu nhiên
- Bao nhiêu bit cần :
  - Tung đồng xu

Example

- Đối với nguồn nhị phân (M=2), $p(1)=\alpha$ và $p(0)=1-\alpha=\beta$.
- Từ (2), chúng ta có entropy nhị phân:
- $H(X)= -\alpha.\log\alpha -(1-\alpha).\log(1-\alpha)$



Source coding theorem

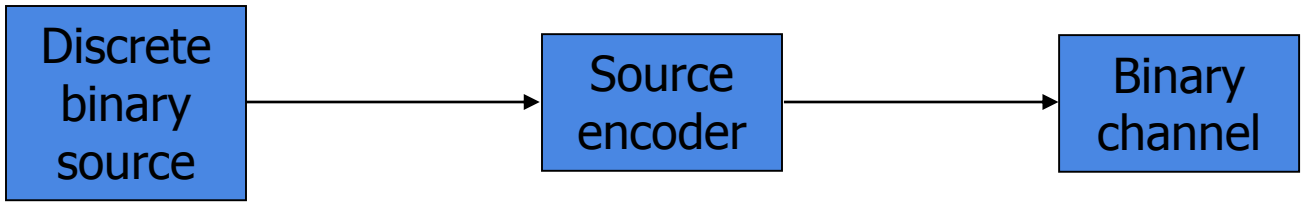
- Thông tin từ nguồn sinh ra các symbol khác nhau có thể diễn tả bằng entropy H(X)
- Tốc độ thông tin nguồn (bit/s):

$$R_s = rH(X) \quad \text{(bit/s)}$$

  - *H(X)*: entropy nguồn (bits/symbol)
  - *r*: tốc độ ký hiệu (symbols/s)
- Giả định nguồn truyền đến kênh truyền:
  - C: dung lượng (bits/symbol)
  - S: tốc độ ký hiệu có thể (symbols/s)
  - S.C: bits/s

Source coding theorem (cont'd)

- Lý thuyết thứ nhất Shannon (noiseless coding theorem):
  - "Cho một kênh và một guồn phát ra thông tin ở tốc độ thấp hơn tốc độ của kênh, có thể mã hóa ngõ ra nguồn dưới một dạng mà nó có thể truyền thông qua kênh"
- Minh họa mã hóa nguồn thông qua ví dụ:



Source symbol rate= r
= 3.5 symbols/s

C = 1 bit/symbol

S = 2 symbols/s

SC = 2 bits/s

Example of Source encoding

- Nguồn nhị phân rời rạc: A(p=0.9), B(p=0.1)
- *Source symbol rate (3.5) >channel capacity (2)*
  ⇨ các ký hiệu nguồn không thể truyền trực tiếp
- Kiểm tra lý thuyết Shannon:
  - H(X)=-0.1 $\log_2$0.1-0.9$\log_2$0.9=0.469bits/symbol
  - $R_s = rH(X)$ = 3.5(0.469)=1.642 bits/s < S.C = 2 bits/s
- Có thể mã hóa nguồn để giảm tốc độ ký hiệu trung bình

Example of Source encoding(cont'd)

- Từ mã được gán thành nhóm *n*-symbol của các ký hiệu nguồn
- Qui luật:
  - Từ mã ngắn nhất cho nhóm có khả dĩ nhất
  - Từ mã dài nhất cho nhóm có ít khả năng nhất
- *Nhóm n* -symbol của ký hiệu nguồn *# mở rộng bậc n của thông tin gốc*

First-Order extension

Source symbol	P ()	Codeword	[P()].[Number of Code Symbols]
A	0.9	0	0.9
B	0.1	1	0.1
			$\overline{L}$=1.0

- Tốc độ ký hiệu của bộ mã hóa = tốc độ ký hiệu nguồn
- Lớn hơn kênh có thể truyền

Second-Order extension

Nhóm 2 ký hiệu nguồn vào một thời điểm:

Source symbol	*P* ()	Codeword	[*P*()].[Number of Code Symbols]
AA	0.81	0	0.81
AB	0.09	10	0.18
BA	0.09	110	0.27
BB	0.01	111	0.03
			$\overline{L}$=1.29

$$\overline{L} = \sum_{i=1}^{2^n} p(x_i) * l_i$$

L: chiều dài từ mã trung bình

$\overline{p}(x_i)$: xác suất ký hiệu thứ i *sau khi nhóm*

l_i: chiều dài của từ mã tương ứng với ký hiệu thứ i

Second-Order extension

$$\frac{\bar{L}}{n} = \frac{1.29}{2} = 0.645 \,\text{code symbols/source symbol}$$

Tốc độ ký hiệ ở ngõ ra bộ mã hóa:

$$r\frac{\bar{L}}{n} = 3.5(0.645) = 2.258 \quad \text{code symbols/sec}$$

>2

Vẫn lớn hơn 2 symbols/second của kênh

Vì vậy, chúng ta tiếp tục mở rộng bậc ba

Third-Order extension

Nhóm 3 ký hiệu nguồn ở cùng thời điểm:

Source symbol	*P* ()	Codeword	[*P*()].[Number of Code Symbols]
AAA	0.729	0	0.729
AAB	0.081	100	0.243
ABA	0.081	101	0.243
BAA	0.081	110	0.243
ABB	0.009	11100	0.045
BAB	0.009	11101	0.045
BBA	0.009	11110	0.045
BBB	0.001	11111	0.005
			$\overline{L}$=1.598

Third-Order extension

$$\frac{\bar{L}}{n} = \frac{1.598}{3} = 0.533$$ code symbols/source symbol

Tốc độ ký hiệu ở ngõ ra bộ mã hóa:

$$r\frac{\bar{L}}{n} = 3.5(0.533) = 1.864\,\text{code symbols/second}$$

Tốc độ này có thể gửi qua kênh

Efficiency of a source code

- Hiệu suất được sử dụng để đo lường hữu ích của mã hóa nguồn

$$eff = \frac{\bar{L}_{\min}}{\bar{L}} = \frac{\bar{L}_{\min}}{\sum_{i=1}^{n} p(x_i) l_i}$$

$$\bar{L}_{\min} = \frac{H(X)}{\log_2 D}$$

Với *H(X)*: entropy của nguồn

D: số lượng ký hiệu trong bảng mã

$$\Rightarrow \quad eff = \frac{H(X)}{\bar{L} \log_2 D}$$

or $$eff = \frac{H(X)}{\bar{L}}$$ truyền nhị phân

Entropy and efficiency of an

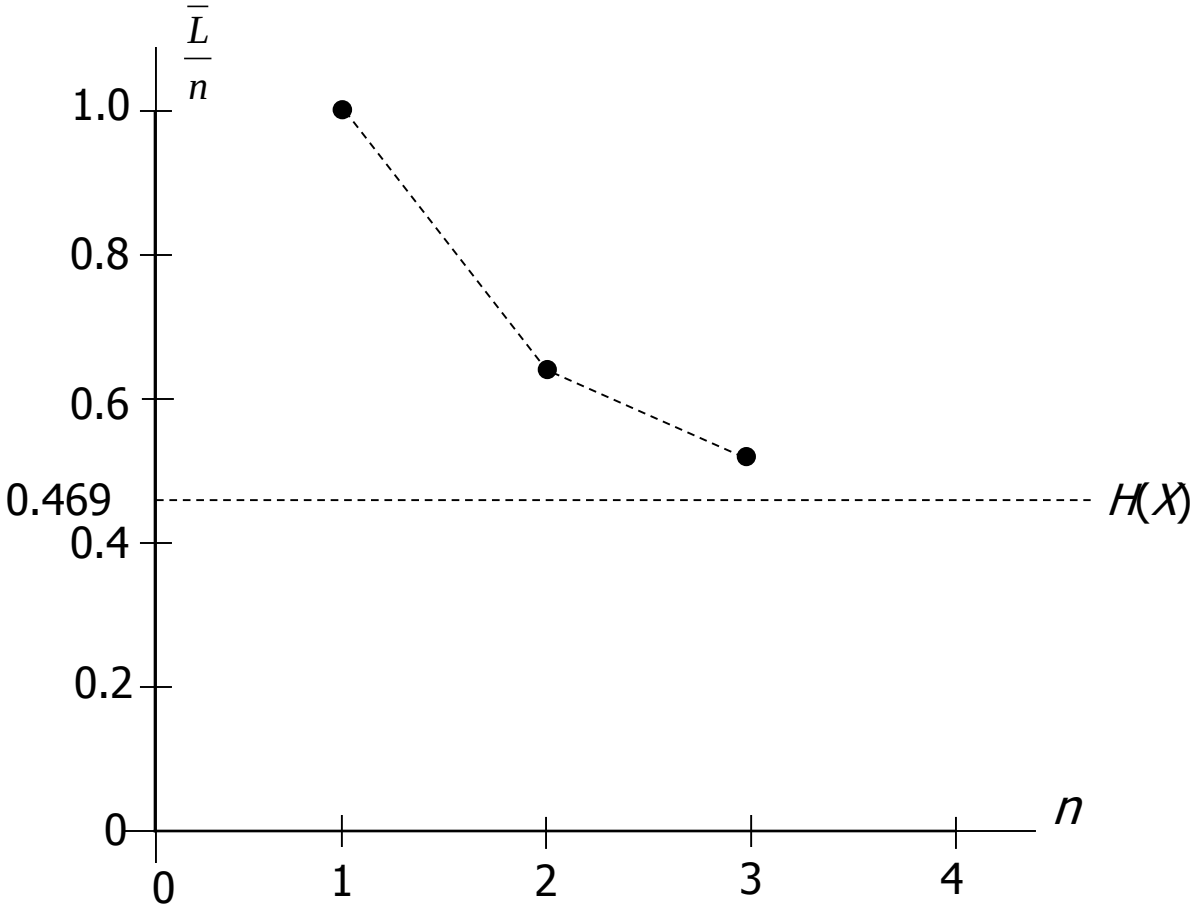
- Entropy mở rộng bậc n của nguồn không bộ nhớ rời rạc:

$$H(X^n) = n * H(X)$$

- Hiệu quả của nguồn mở rộng:

$$eff = \frac{n.H(X)}{\overline{L}}$$

Behavior of *L*/*n*



- $\bar{L}/n$ càng giảm khi tăng n
- Giảm chiều dài từ mã trung bình dẫn đến gia tang độ phức tạp giải mã

Shannon-Fano Coding [1]

- Thuật toán: 3 bước
  1. Liệt kê các ký hiệu nguồng theo xác suất giảm dần
  2. Phân tập nguồn thành 2 nhóm nhỏ sao cho xác suất càng bằng nhau càng có thể. 0 được gán cho nhóm trên và 1 được gán cho nhóm dưới
  3. Tiếp tục phân chia nhóm con đến khi không thể phân chia nữa

Example of Shannon-Fano Coding

U_i	p_i	1	2	3	4	5	Codewords
U_1	.34	0	0				00
U_2	.23	0	1				01
U_3	.19	1	0				10
U_4	.1	1	1	0			110
U_5	.07	1	1	1	0		1110
U_6	.06	1	1	1	1	0	11110
U_7	.01	1	1	1	1	1	11111

Shannon-Fano coding

$$\overline{L} = \sum_{i=1}^{7} p_i l_i = 2.41$$

$$H(U) = -\sum_{i=1}^{7} p_i \log_2 p_i = 2.37$$

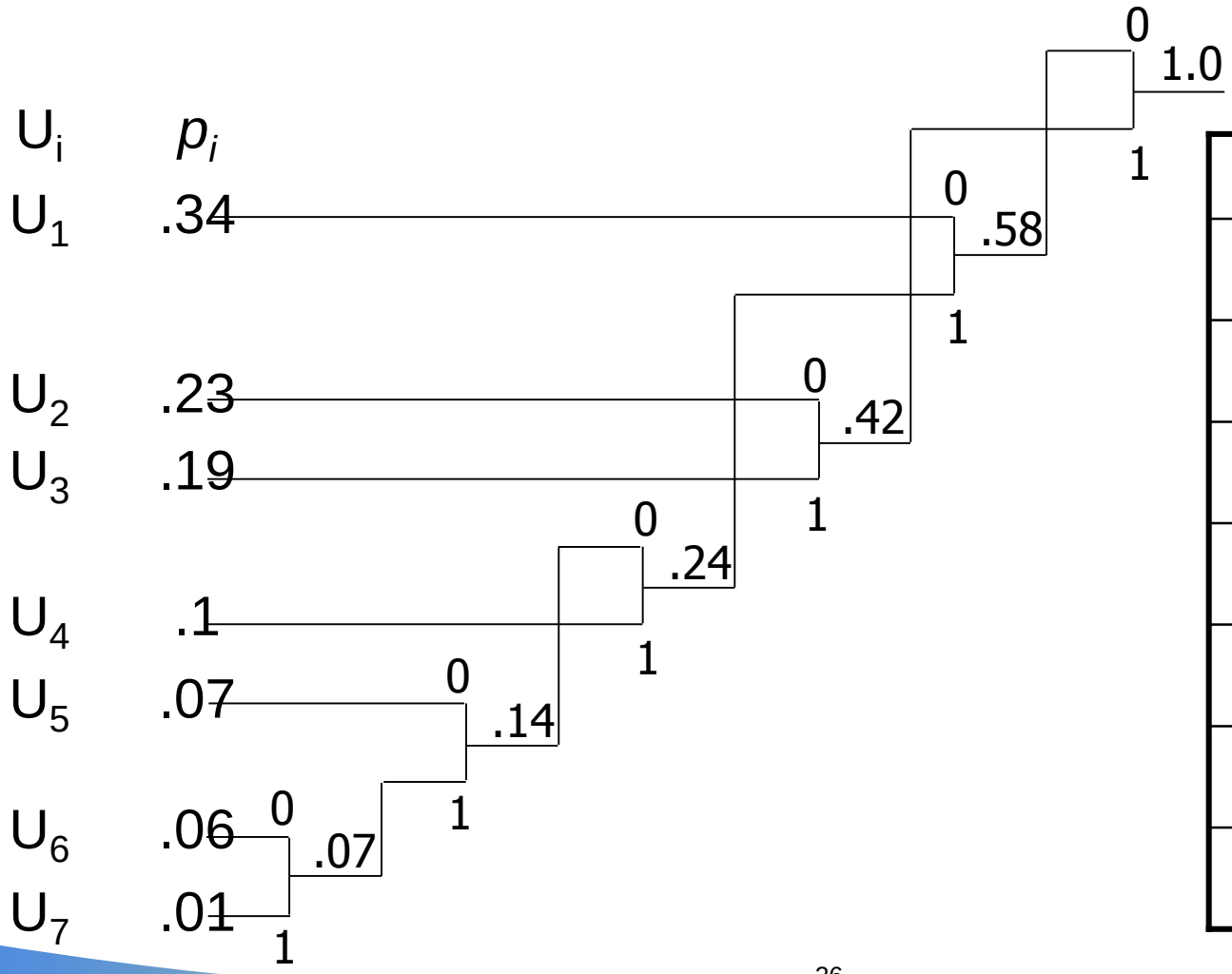
$$eff = \frac{H(U)}{\overline{L}} = \frac{2.37}{2.41} = 0.98$$

- Từ mã được phát ra là phần đầu của từ mã do việc phân nhóm bằng nhau

Huffman Coding

- Thủ tục: 3 bước
  1. Liệt kê ký hiệu nguồn theo xác suất giảm dần. Hai ký hiệu nguồn có xác suất thấp nhất được gán là 0 và 1.
  2. Hai ký hiệu nguồn kết hợp tạo thành một ký hiệu nguồn mới với xác suất bằng với tổng xác suất hai nguồn. Liệt kê lại trật tự sau khi cập nhất xác suất mới.
  3. Lặp lại cho đến khi sắc xuất cuối cùng của ký hiệu mới vừa kết hợp là 1.0.
- Ví dụ:

Examples of Huffman Coding



U_i	Codewords
U_1	00
U_2	10
U_3	11
U_4	011
U_5	0100
U_6	01010
U_7	01011

Huffman Coding: disadvantages

- Khi nguồn có nhiều ký hiệu (outputs/messages), từ mã trở nên cồng kềnh⇨mã hóa Hufman + từ mã có chiều dài cố định.
- Vẫn còn có thể giảm và việc giảm lớn với một một tập nhỏ của thông tin⇨ nhóm nhiều nhiều thông tin độc lập

Huffman Coding: disadvantages

- Việc nhóm sẽ làm cho suy giảm nhỏ nhưng số từ mã tang theo hàm mũ, từ mã sẽ phức tạp hơn và sinh ra độ trễ.

Entropy example 2

- Cuộc đua ngựa với 8 con ngựa, với xác suất thắng là ½, ¼, 1/8, 1/16, 1/64, 1/64, 1/64, 1/64
- Entropy H(X) = 2 bits
- Chúng ta cần bao nhiêu bit?
- (a) chỉ số mỗi con ngựa ➔ log8 = 3 bits
- (b) gán từ mã ngắn với xác suất cao:

  0, 10, 110, 1110, 111100, 111101, 111110, 111111

➔ chiều dài từ mã trung bình = 2 bits!

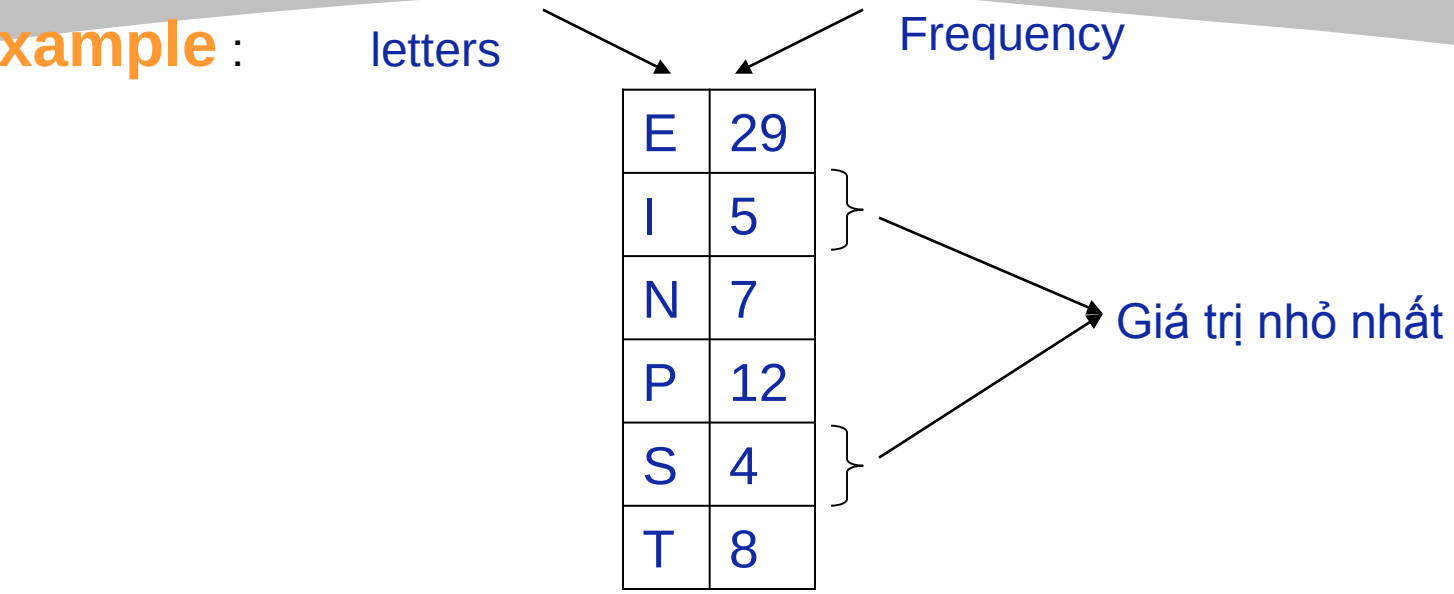
Thuật toán

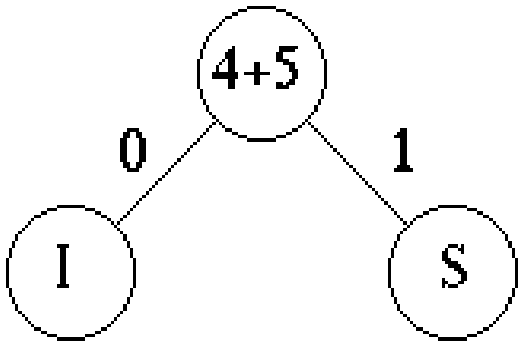
Xây dựng cây Huffman :

1. liệt kê tần số giảm dần.
2. chọn hai giá trị nhỏ nhất.
3. tạo thành cây nhị phân với gán nhãn trên mỗi nhánh.
4. thay thế hai giá trị nhỏ nhất với tổng
5. tạo thành một chuỗi mới
6. lặp lại với hai giá trị xác suất nhỏ nhất và gán nhãn cây nhị phân.
7. Trở lại bước 2 đến khi không còn ký tự.
8. kết thúc!

Ví dụ

Example :



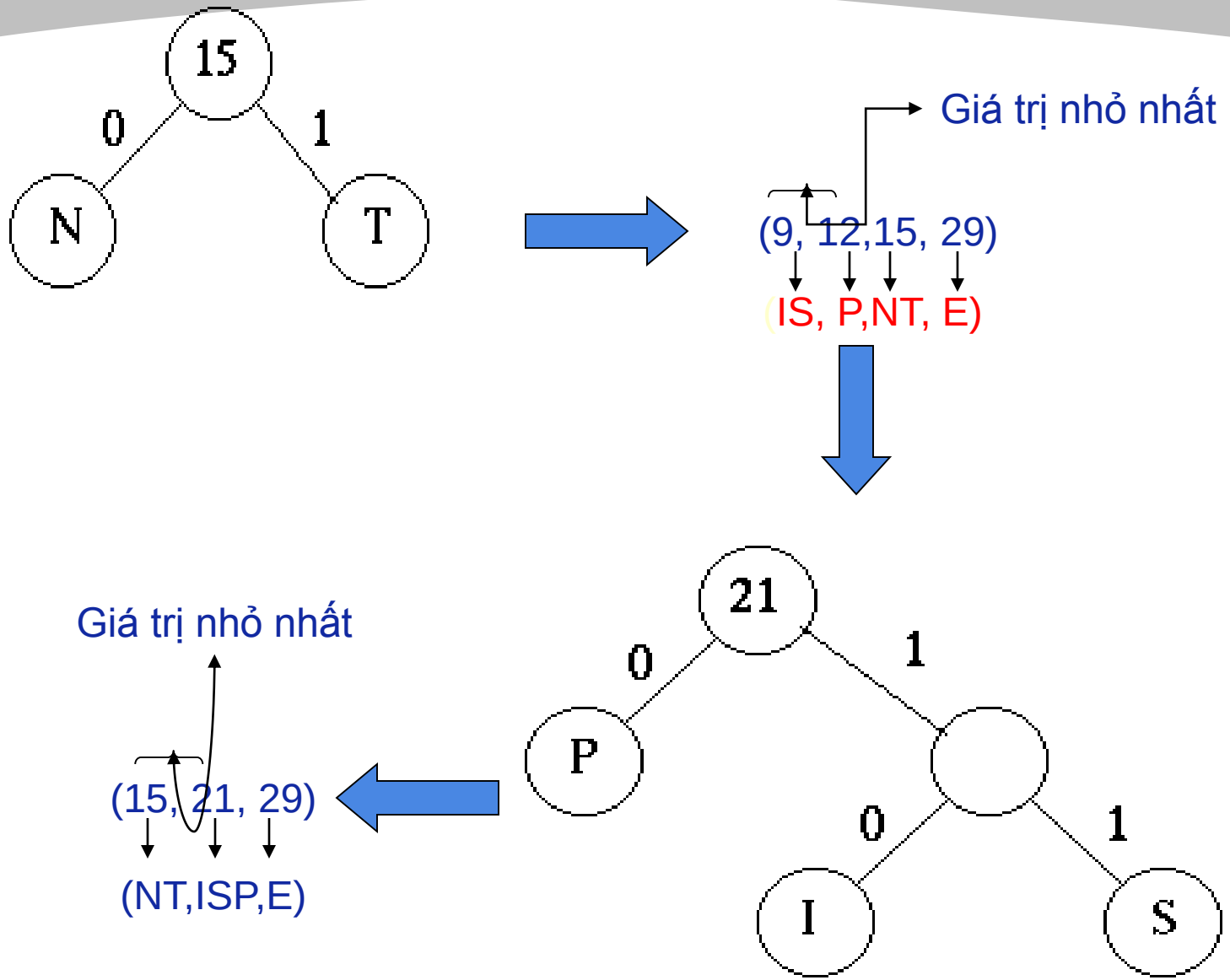


4+5=9 → chuỗi mới :(7, 8, 9, 12, 29).

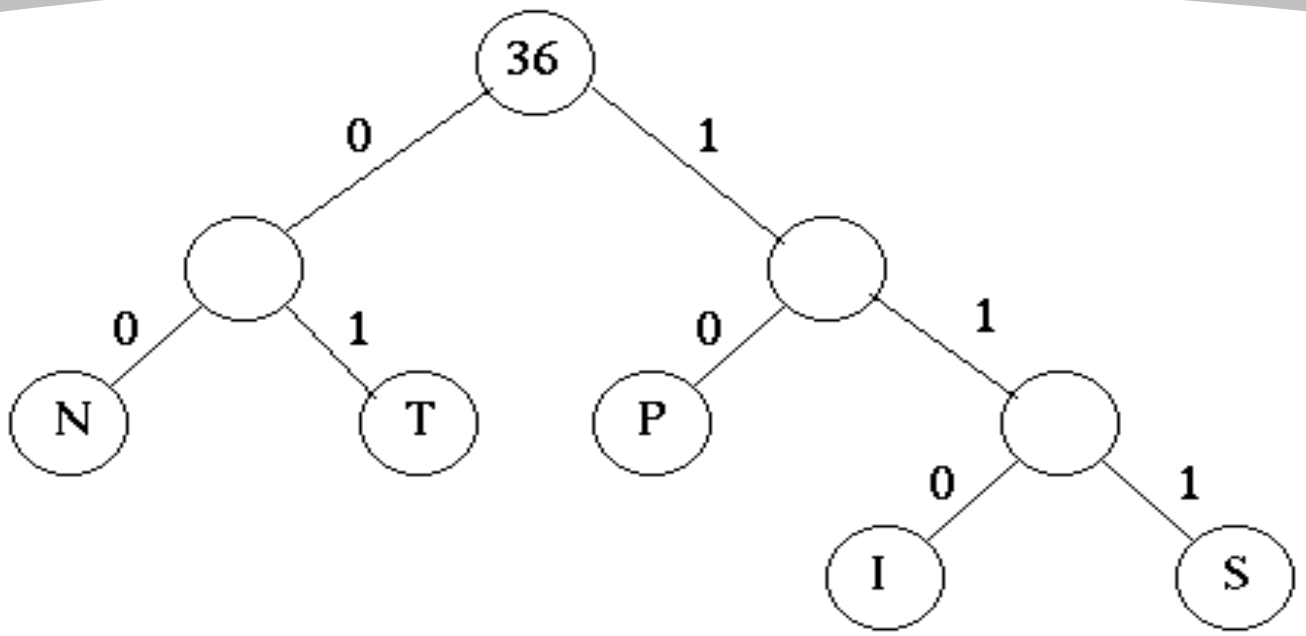
Giá trị nhỏ nhất

(N ,T,IS, P, E)

Ví dụ (tt)


15
0
1
N
T
Giá trị nhỏ nhất
(9, 12,15, 29)
IS, P,NT, E)
21
0
1
P
0
1
I
S
Giá trị nhỏ nhất
(15, 21, 29)
(NT,ISP,E)

Ví dụ (tt)



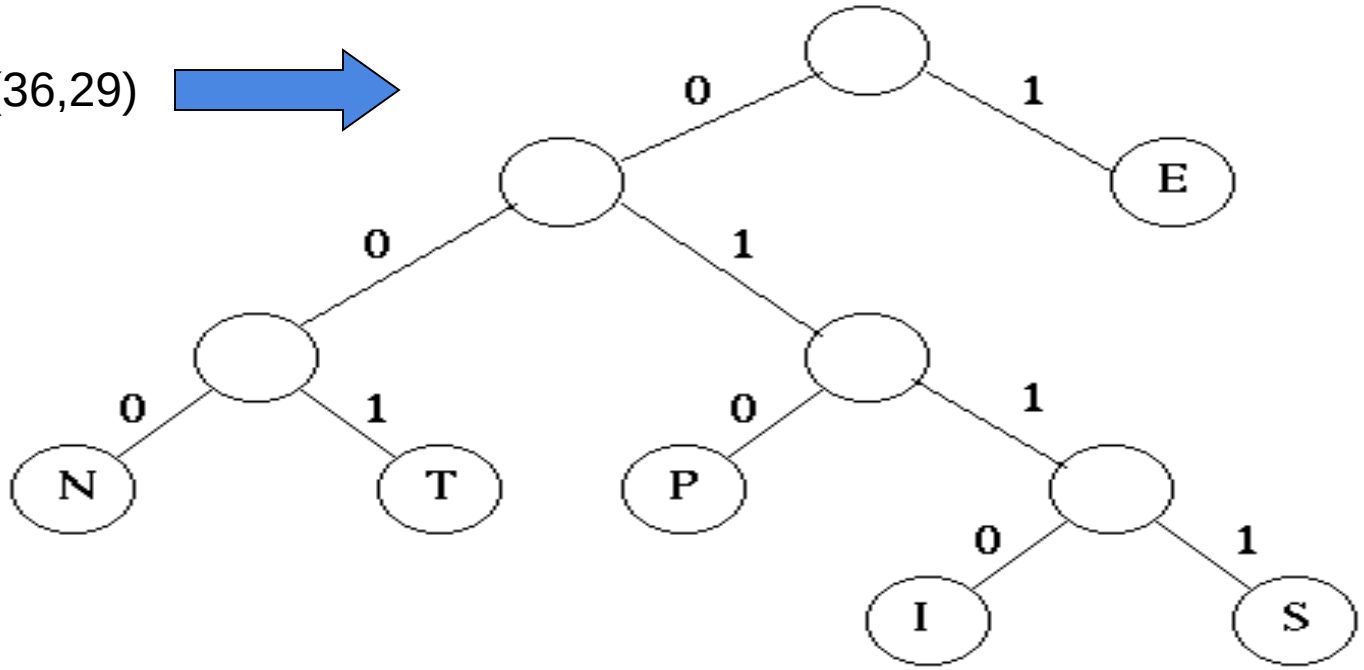
(29 , 36)

(E,INPST)

Kết hợp cuối cùng

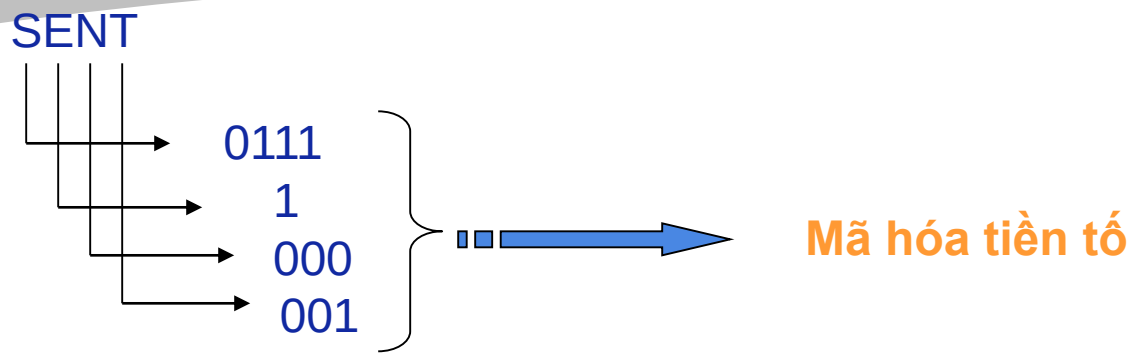
Ví dụ (tt)

(36,29)



E	1
I	0110
P	010
N	000
S	0111
T	001

Ví dụ (tt)



Khi đọc là 1, chúng ta biết là E. Chúng ta không cần xem thêm bit nữa.

0110 là I.

Khi nhận thấy 01, chưa biết là I hay P hay S, ...

NOTE: nó không tạo ra sự khác biệt giữa nhánh phía bên trái và bên phải

Ví dụ

Chúng ta có 6 ký tự, vậy chúng ta cần 3 bit cho mỗi ký tự với mã hóa thông thường. Nếu bản tin có 65 ký tự, vì vậy cần 3*65=195 bits

Để mã hóa, chúng ta sử dụng thuật toán “Huffman” :

1*29+4*5+3*12+3*7+4*4+3*8=146 bits.

Chúng ta tiết kiệm: 100-(146/195)*100=25% bộ nhớ!