BỘ CÔNG THƯƠNG
TRƯỜNG CAO ĐẲNG KỸ THUẬT CAO THẮNG
KHOA ĐIỆN TỬ - TIN HỌC

CHƯƠNG IV
XỬ LÝ SỐ LIỆU TRUYỀN

Môn Học
TRUYỀN SỐ LIỆU

NỘI DUNG

4.1 Các dạng lỗi

4.2 Phát hiện lỗi

4.3 Sửa lỗi

4.4 Nén số liệu

NỘI DUNG

4.1 Các dạng lỗi

4.2 Phát hiện lỗi

4.3 Sửa lỗi

4.4 Nén số liệu

Các dạng lỗi

- Có 2 loại lỗi
  - Lỗi 1 bit (Single-bit errors)
    - Chỉ 1 bit bị lỗi
    - Không ảnh hưởng đến các bit xung quanh
    - Thường xảy ra do nhiễu trắng
  - Lỗi chùm (Burst errors)
    - Một chuỗi liên tục B bit trong đó có bit đầu, bit cuối và các bit bất kỳ nằm giữa chuỗi đều bị lỗi
    - Thường xảy ra do nhiễu xung
    - Ảnh hưởng càng lớn đối với tốc độ truyền cao

CAO ĐẲNG KỸ THUẬT CAO THẮNG

NỘI DUNG

4.1 Các dạng lỗi

4.2 Phát hiện lỗi

4.3 Sửa lỗi

4.4 Nén số liệu

Phát hiện lỗi

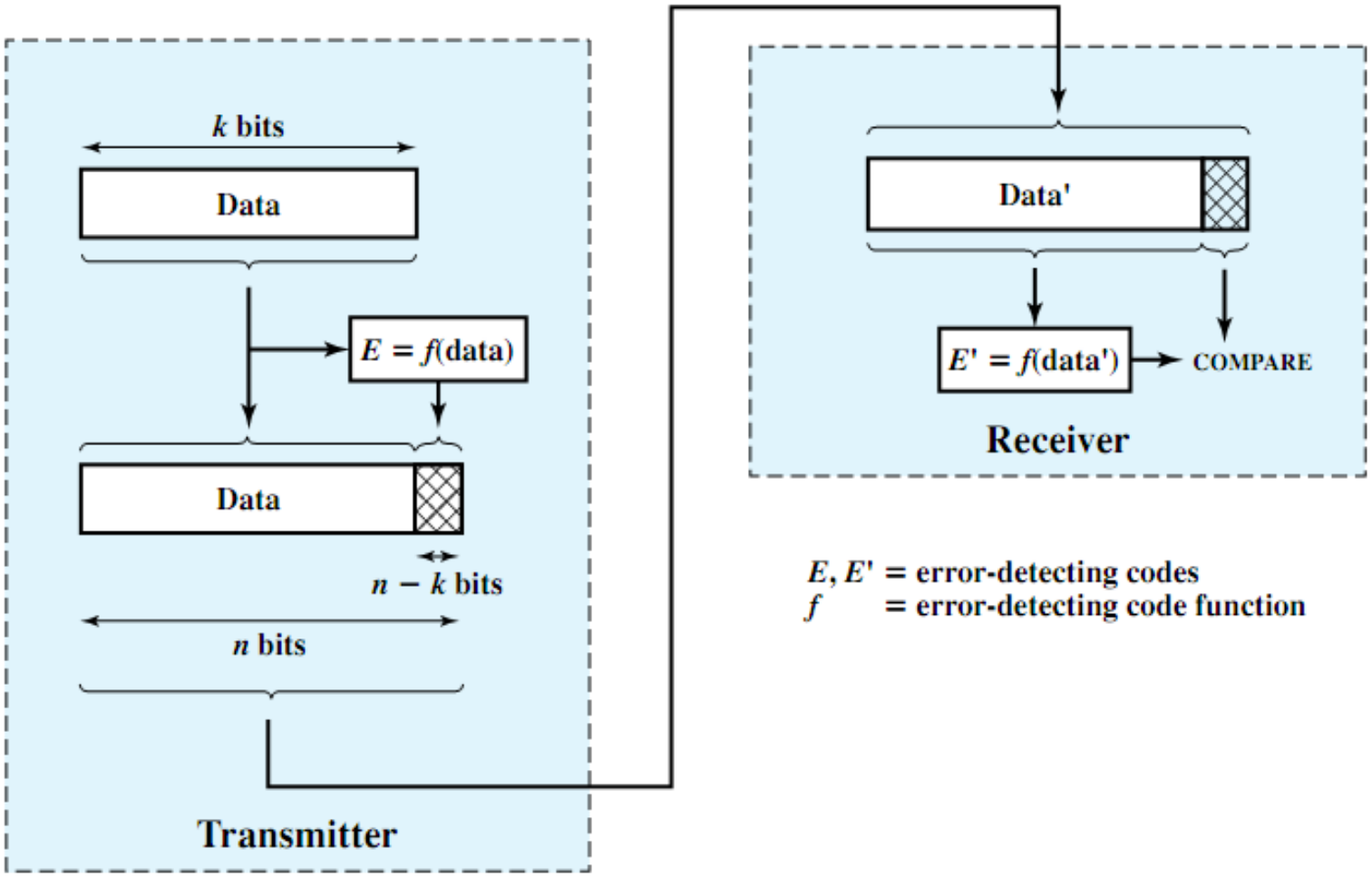


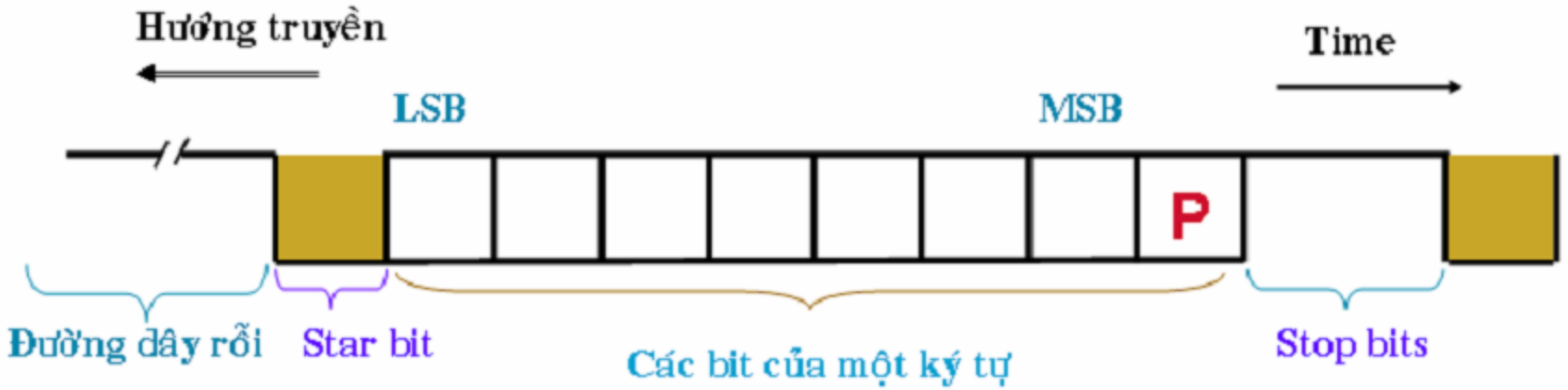
Figure 6.3 Error Detection Process

Phát hiện lỗi

- Parity check
  - Là phương pháp phát hiện lỗi đơn giản nhất
  - Gắn một bit parity vào khối dữ liệu sao cho tổng số bit 1 của khối dữ liệu là một số chẵn hoặc lẻ
  - Có 2 kiểu kiểm tra parity
    - Parity chẵn
    - Parity lẻ
  - Đặc điểm: chỉ dò được lỗi sai một số lẻ bit, không dò được lỗi sai một số chẵn bit, không sửa được lỗi, ít dùng trong truyền dữ liệu đi xa, đặc biệt ở tốc độ cao

Parity chẵn và lẻ

- Parity check: bit kiểm tra được thêm vào sao cho tổng số bit 1 của chuỗi bit là số chẵn hoặc lẻ



1001001	*1*	*Parity chẵn*
1001001	*0*	*Parity lẻ*

Ví dụ

- Cho biết tín hiệu truyền là kí tự mã **ASCII** với 1 bit kiểm tra chẵn thêm vào dữ liệu. Cho biết dữ liệu nhận được đúng hay sai, và nếu đúng thì ký tự đã truyền là gì nếu chuỗi bit nhận được là:

a) [LSB]10110010[MSB]

b) [LSB]11001011[MSB]

Kiểm tra tổng khối (Block Sum Check)

- Sử dụng khi truyền dữ liệu dưới dạng một khối các ký tự, trong kiểu kiểm tra này, mỗi ký tự truyền đi sẽ được phân phối 2 bit kiểm tra là parity hàng và parity cột. Các bit parity theo từng cột được gọi là ký tự kiểm tra khối BCC (Block Check Character)
- Phát hiện và sửa sai nếu lỗi bit đơn
- Không phát hiện sai nếu các bit sai kiểu chùm như: sai 4 bit, 2 bit cùng hàng và 2 bit cùng cột
- Các trường hợp còn lại thì phát hiện sai được

Kiểm tra tổng khối (Block Sum Check)

P_R	B_6	B_5	B_4	B_3	B_2	B_1	B_0	
0	0	0	0	0	0	1	0	STX
1	0	1	0	1	0	0	0	
0	1	0	0	0	1	1	0	
0	0	1	0	0	0	0	0	
1	0	1	0	1	1	0	1	
0	1	0	0	0	0	0	0	
1	1	1	0	0	0	1	1	
1	0	0	0	0	0	1	1	ETX
1	1	0	0	0	0	0	1	BCC (chẵn)

Parity lẻ

Các ký tự của khung

Kiểm tra tổng khối (Block Sum Check)

P_R	B_6	B_5	B_4	B_3	B_2	B_1	B_0	
0	0	0	0	0	0	1	0	STX
1	0	1	0	1	0	0	0	
0	1	1	0	0	0	1	0	
0	0	1	0	0	0	0	0	
1	0	1	0	1	1	0	1	
0	1	0	0	0	0	0	0	
1	1	0	0	0	1	1	1	
1	0	0	0	0	0	1	1	ETX
1	1	0	0	0	0	0	1	BCC (chẵn)

Parity lẻ

Các ký tự của khung

- *Độ tin cậy của BCC khoảng 98 %*

Cyclic Redundant Check (CRC)

- Nguyên lý
  - k bit message
  - Bên phát tạo ra chuỗi (n-k) bit FCS (Frame Check Sequence) sao cho frame gửi đi gồm n bit chia hết cho một số xác định trước
  - Bên thu chia frame nhận được cho cùng một số và nếu không có phần dư thì có khả năng không có lỗi

Cyclic Redundant Check (CRC)

- Số học modulo 2
  - Cộng hai số nhị phân (không nhớ)
  - Exclusive OR (XOR)

$$\begin{array}{r} 1111 \\ +1010 \\ \hline 0101 \end{array} \qquad \begin{array}{r} 1111 \\ -0101 \\ \hline 1010 \end{array} \qquad \begin{array}{r} 11001 \\ \times\ 11 \\ \hline 11001 \\ 11001 \\ \hline 101011 \end{array}$$

Cyclic Redundant Check (CRC)

- Xác định
  - T = frame có n bit cần truyền
  - D = khối dữ liệu k bit (message) (k bit đầu của T
  - F = (n-k) bit FSC (n-k) bit cuối của T
  - P = số chia được xác định trước gồm n-k +1 bit

$$T = 2^{n-k}D + F$$

  - Giả sử

$$\frac{2^{n-k}D}{P} = Q + \frac{R}{P}$$

Cyclic Redundant Check (CRC)

- Xác định
  - Nếu lấy F = R thì $T = 2^{n-k}D + R$
  - Chia T cho P ta có

$$\frac{T}{P} = \frac{2^{n-k}D + R}{P} = \frac{2^{n-k}D}{P} + \frac{R}{P}$$

  - Suy ra $\frac{T}{P} = Q + \frac{R}{P} + \frac{R}{P}$
  - Mà phép cộng modulo 2 của một số với chính nó bằng 0
  - Vậy $\frac{T}{P} = Q + \frac{R + R}{P} = Q$

Ví dụ

- Cho khối dữ liệu D = 1010001101 (10 bit)
- Số chia xác định trước P = 110101 (6 bit)
- Tìm FCS = ? , T = ?
- Giải:
  - Ta có k = 10
  - n – k + 1 = 6
  - Suy ra n = 6-1+10 = 15
  - Lấy 2^{n-k} D chia cho P
  - $2^{n-k}D = 2^5$ D = 101000110100000
  - Lấy kết quả trên chia cho P ta được thương là 110001010 dư 01110

Ví dụ

- Vậy suy ra F = 01110
- Từ đó suy ra T = 1010001101**01110**

Cyclic Redundant Check (CRC)

- Số chia P
  - Dài hơn 1 bit so với FCS mong muốn
  - Được chọn tùy thuộc vào loại lỗi mong muốn phát hiện
  - Yêu cầu tối thiểu: msb và lsb phải là 1
- Biểu diễn lỗi
  - Lỗi = nghịch đảo bit (i.e. xor của bit đó với 1)
    - T: frame được truyền
    - Tr: frame nhận được
    - E: error pattern với 1 tại những vị trí lỗi xảy ra
  - Nếu có lỗi xảy ra (E ≠0) thì bộ thu không phát hiện ra lỗi đó khi và chỉ khi Tr chia hết cho P, nghĩa là E chia hết cho khó có khả năng xảy ra

Cyclic Redundant Check (CRC)

- Cách khác để xác định FCS là dùng đa thức
  - D = 110011 → D(x) = $X^5 + X^4 + X + 1$
  - P = 11001 → P(x) = $X^4 + X^3 + 1$

$$\frac{X^{n-k}D(X)}{P(X)} = Q(X) + \frac{R(X)}{P(X)}$$

$$T(X) = X^{n-k}D(X) + R(X)$$

Ví dụ

- Dữ liệu cần truyền 1010001101 (k = 10) → Đa thức biểu diễn $X^9 + X^7 + X^3 + X^2 + 1$
- Cho đa thức sinh: $P(x) = X^5 + X^4 + X^2 + 1$ (n – k + 1 = 6 hay n – k = 5 hay n = 15)
- Dữ liệu D dịch trái 5 bit. $X^{n-k}\ D(x) = X^5\ D(x) = X^{14} + X^{12} + X^8 + X^7 + X^5$

Ví dụ

- Thực hiện phép chia

$$
\begin{array}{l|l}
x^5 + x^4 + x^2 + 1 & x^9 + x^8 + x^6 + x^4 + x^2 + x \\
 & x^{14} + x^{12} + x^8 + x^7 + x^5 \\
 & x^{14} + x^{13} + x^{11} + x^9 \\
 & x^{13} + x^{12} + x^{11} + x^9 + x^8 + x^7 + x^5 \\
 & x^{13} + x^{12} + x^{10} + x^8 \\
 & x^{11} + x^{10} + x^9 + x^7 + x^5 \\
 & x^{11} + x^{10} + x^8 + x^6 \\
 & x^9 + x^8 + x^7 + x^6 + x^5 \\
 & x^9 + x^8 + x^6 + x^4 \\
 & x^7 + x^5 + x^4 \\
 & x^7 + x^6 + x^4 + x^2 \\
 & x^6 + x^5 + x^2 \\
 & x^6 + x^5 + x^3 + x \\
 & x^3 + x^2 + x = R(x)
\end{array}
$$

Ví dụ

- Vậy F = 01110
- Dữ liệu được truyền là T= 1011101000**01110**

Cyclic Redundant Check (CRC)

- 4 đa thức sinh được sử dụng rộng rãi
  - CRC-12 = $X^{12} + X^{11} + X^3 + X^2 + X + 1$
    - 12-bit FCS
    - Dùng để truyền chuỗi các ký tự có độ dài 6-bit
  - CRC-16 = $X^{16} + X^{15} + X^2 + 1$
    - 16-bit FCS
    - Dùng để truyền chuỗi các ký tự có độ dài 8-bit
    - USA
  - CRC-CCITT = $X^{16} + X^{12} + X^5 + 1$
    - Europe
  - CRC-32 = $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
    - 32-bit FCS
    - Point-point synchronous transmission, DVB-T…

Cyclic Redundant Check (CRC)

- Các lỗi được phát hiện

–Tất cả các lỗi bit đơn

–Tất cả các lỗi kép nếu P(x) có ít nhất 3 toán hạng

– Một số lẻ lỗi bất kỳ nếu P(x) chứa 1 thừa số (x+1)

– Bất kỳ lỗi chùm nào mà chiều dài của chùm nhỏ hơn hoặc bằng chiều dài FCS (n=k)

–Hầu hết các lỗi chùm lớn hơn

- CRC là một trong những phương pháp thông dụng và hiệu quả nhất để phát hiện lỗi

NỘI DUNG

4.1 Các dạng lỗi

4.2 Phát hiện lỗi

4.3 Sửa lỗi

4.4 Nén số liệu

Sửa lỗi

- Cách sửa lỗi thông thường là yêu cầu truyền lại khối dữ liệu bị lỗi
- Không thích hợp cho các ứng dụng trao đổi dữ liệu không dây
  - Xác suất lỗi cao, dẫn đến việc phải truyền lại nhiều
  - Thời gian trễ truyền lớn hơn nhiều thời gian truyền 1 khối dữ liệu
  - Cơ chế truyền lại là truyền lại khối dữ liệu bị lỗi và nhiều

  khối dữ liệu khác tiếp theo
- Cần thiết sửa lỗi dựa vào các dữ liệu nhận được

NỘI DUNG

4.1 Các dạng lỗi

4.2 Phát hiện lỗi

4.3 Sửa lỗi

4.4 Nén số liệu

Mã hoá Huffman

- Dựa vào tần suất xuất hiện của các ký tự trong một khung truyền
- Mã hoá số bit nhỏ hơn cho các ký tự có tần suất xuất hiện nhiều và mã hoá với số bit nhiều hơn cho các ký tự có tần số xuất hiện ít
- Trước tiên xác định tần suất xuất hiện của từng ký tự
- Dùng cây Huffman (cây nhị phân với các nhánh được gán các giá trị 0 1)

Mã hóa Huffman

- Xét ví dụ: Cho nguồn tạo một thông điệp gồm các ký tự AAAABBCD biết rằng tốc độ ký hiệu bằng 2000 symbols trong 1 giây

a. Cho biết các từ mã A, B, C, D trường hợp mã hóa đồng đều

b. Lặp lại câu a với mã Huffman

Mã hóa Huffman

- Giải:

a. Nếu mã hóa đồng đều thì ta có 4 ký hiệu nên dùng 2 bit để mã hóa. Cụ thể có thể chọn như sau:

A: 00

B: 01

C: 10

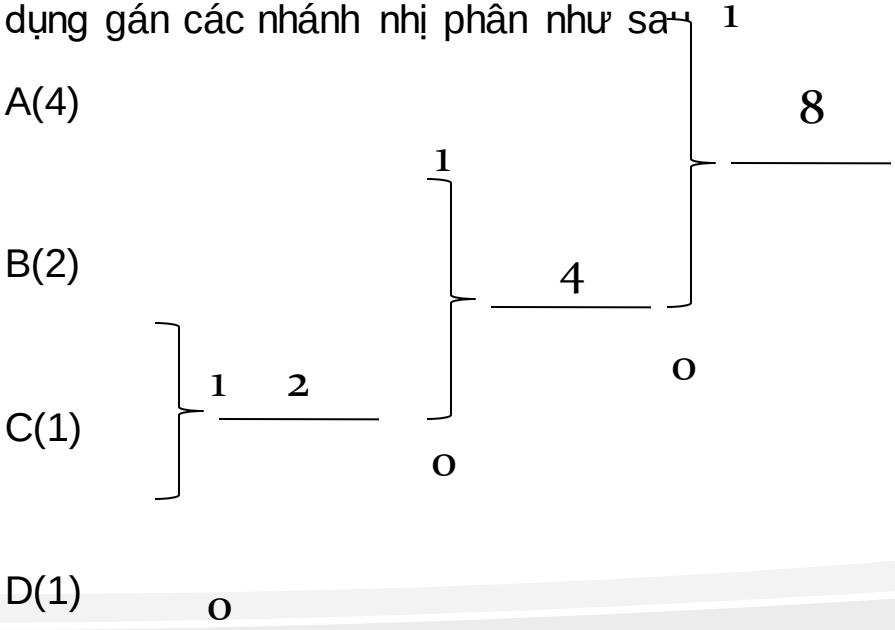
D: 11

Mã hóa Huffman

- Giải:

b. Số lần xuất hiện của A là 4, của B là 2, của C và D đều là 1

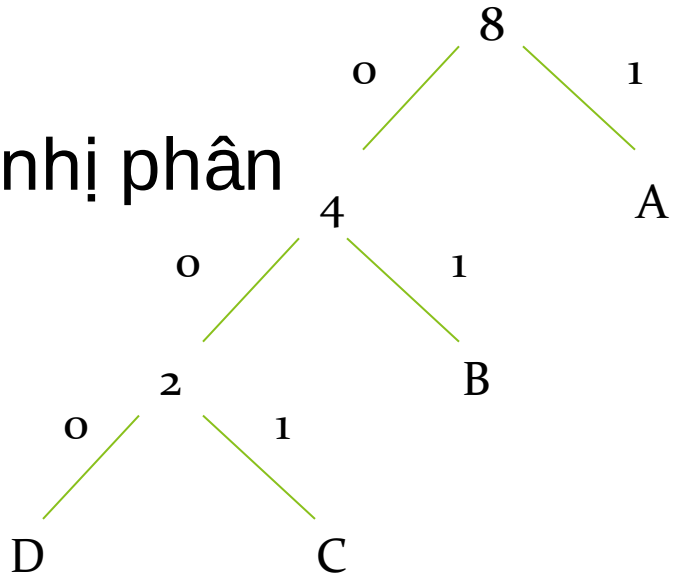
Sắp xếp các ký hiệu theo tần suất xuất hiện giảm dần và áp dụng gán các nhánh nhị phân như sau



Mã hóa Huffman

- Giải:

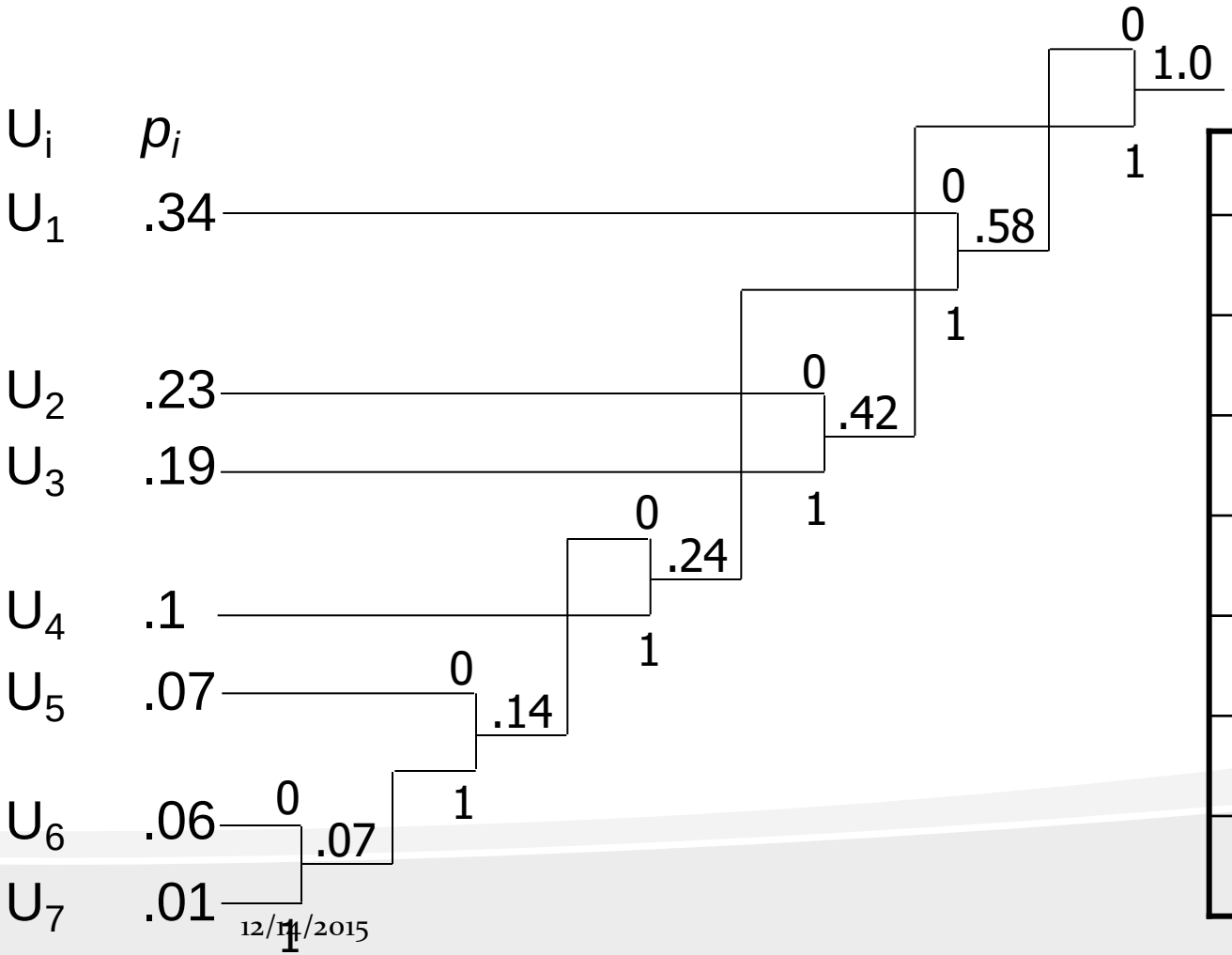
b. Lập cây nhị phân



Khi mã hóa các ký hiệu thí gán các bit nhị phân từ rễ tới lá, ta có

A= 1; B= 01; C=001; D=000

Mã hoá Huffman



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