コンピュータ科学II

担当：武田敦志 <takeda@cs.tohoku-gakuin.ac.jp>

http://takeda.cs.tohoku-gakuin.ac.jp/comp3/

復習問題

- 下記の表に従ってエントロピーを求めよ

天気予報と実際の天気

実際の天気＼天気予報	a : 晴の日	b : 曇の日	c : 雨の日
A : 晴の日	6日	3日	3日
B : 曇の日	0日	6日	0日
C : 雨の日	0日	3日	3日

『実際の天気』のエントロピー：

『天気予報が晴の日』のエントロピー：

『天気予報が曇の日』のエントロピー：

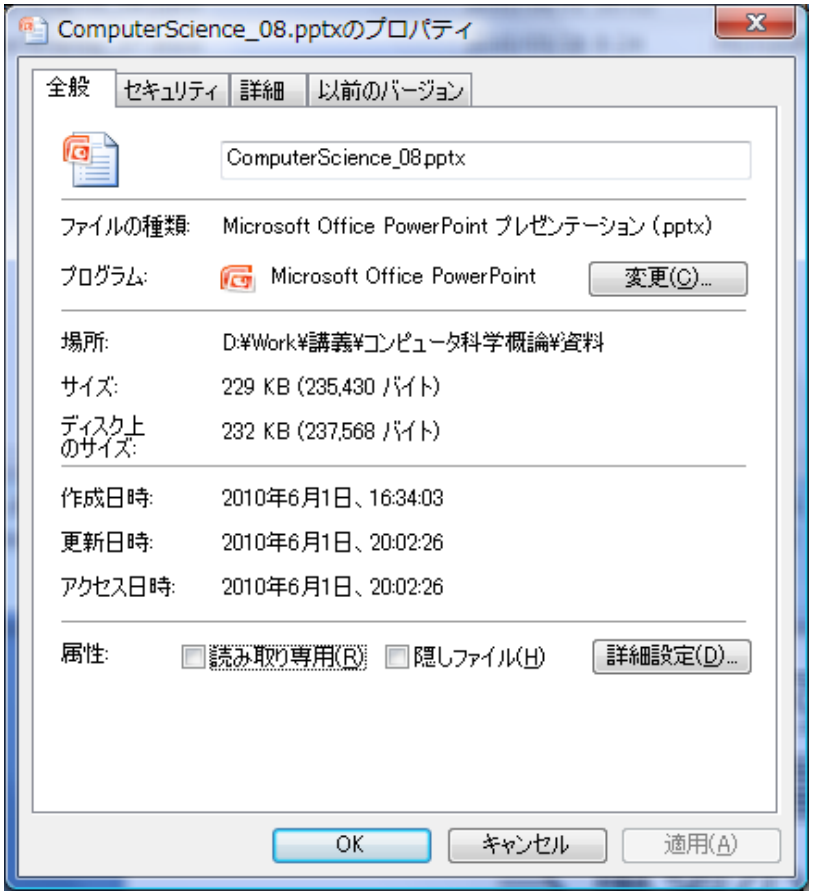
『天気予報が雨の日』のエントロピー：

今日の話

- 圧縮符号
  - ハフマン符号
  - LZ符号
    - 冗長量を減らすことによりデータを圧縮できる
- 誤り訂正符号
  - ハミング符号
    - 符号間の最小ハミング距離を確保することにより
      誤り検出・誤り訂正が可能となる

冗長量(1)

- 表現上の情報量と本当の情報量
  - 表現上の情報量（データ量）
    - 情報を記録・送信するために必要な情報量
      - ⇒ ファイルの大きさなど
  - 本当の情報量
    - 情報を表現するために必要な情報量
      - ⇒ 事象のエントロピー



表現上の情報量 ≧ 本当の情報量

理論上は『本当の情報量』の長さで表現できる

表現上の情報量と本当の情報量の差分：**冗長量**

冗長量(2)

- 表現上の情報量と本当の情報量

事象：P(晴) = 1/2
　　P(曇) = 1/4
　　P(雨) = 1/4

情報：天気は晴です

表現上の情報量

6 [文字] × 24 [bit/文字] ＝ 144 [bit]

本当の情報量

エントロピー ＝ 1.5 [bit]

無駄な情報量

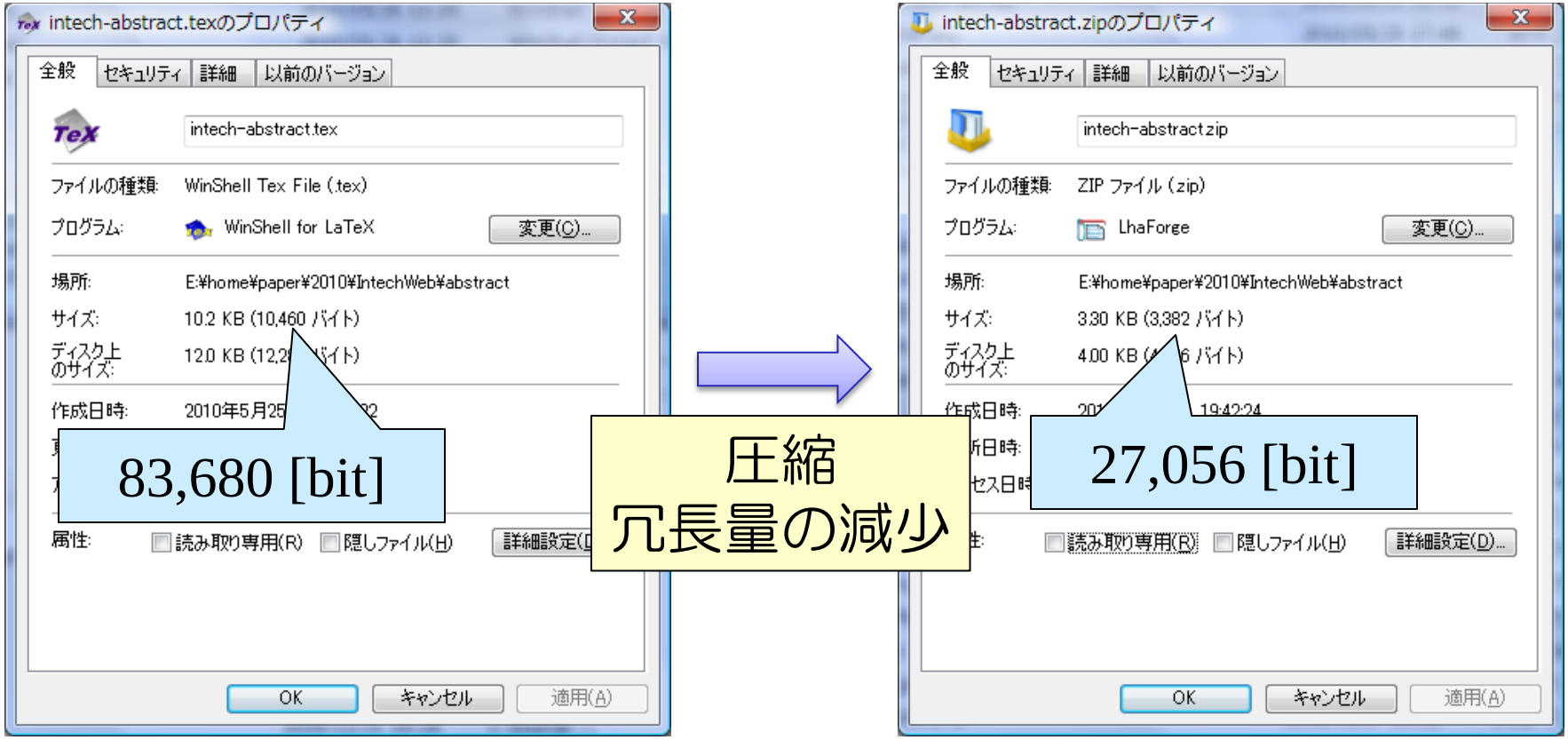
冗長量 ＝ 144 - 1.5 = 142.5 [bit]

冗長量(3)

- 冗長量と圧縮技術

  冗長量を少なくすると、表現のための情報量が減る

  ⇒ データを記録・送信するための容量を少なくできる



ハフマン符号(1)

- データの圧縮

A,B,C,D,E,F,G,H で構成された文字列を表現する

文字列表現　　AABBBBCDEEFFFFGH

P(A) = 1/8	P(E) = 1/8
P(B) = 1/4	P(F) = 1/4
P(C) = 1/16	P(G) = 1/16
P(D) = 1/16	P(H) = 1/16

エントロピー：E =

必要なデータの長さ：

ハフマン符号(2)

- データの表現（その１）

それぞれの文字を以下の符号で表現する

$A = 000_2$　　$B = 001_2$　　$C = 010_2$　　$D = 011_2$

$E = 100_2$　　$F = 101_2$　　$G = 110_2$　　$H = 111_2$

文字列表現　　AABBBBCDEEFFFFGH

データ表現　　000 000 001 001 001 001 010 011
100 100 101 101 101 101 110 111

データの長さ：48 [bit]

ハフマン符号(3)

- データの表現（その２）

それぞれの文字を以下の符号で表現する

$A = 010_2$	$B = 10_2$	$C = 0010_2$	$D = 0011_2$
$E = 011_2$	$F = 11_2$	$G = 0000_2$	$H = 0001_2$

文字列表現　AABBBBCDEEFFFFGH

データ表現　010 010 10 10 10 10 0010 0011
011 011 11 11 11 11 0000 0001

データの長さ：44 [bit]

4bitの圧縮

ハフマン符号(4)

- 圧縮符号の考え方

文字列表現　AABBBBCDEEFFFFGH

出現確率が高い文字：短い符号を使う
出現確率が低い文字：長い符号を使う

P(A) = 1/8	I(A) = 3[bit]
P(B) = 1/4	I(B) = 2[bit]
P(C) = 1/16	I(C) = 4[bit]
P(D) = 1/16	I(D) = 4[bit]
P(E) = 1/8	I(E) = 3[bit]
P(F) = 1/4	I(F) = 2[bit]
P(G) = 1/16	I(G) = 4[bit]
P(H) = 1/16	I(H) = 4[bit]

ハフマン符号の符号化(1)

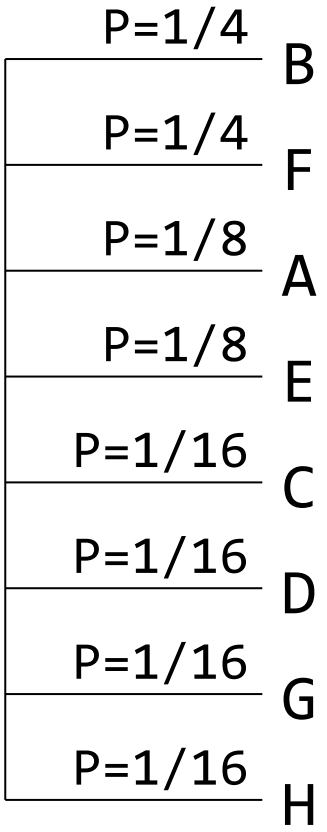
- ハフマン符号の生成（１）

文字列表現　　AABBBBCDEEFFFFGH

P(A) = 1/8
P(B) = 1/4
P(C) = 1/16
P(D) = 1/16
P(E) = 1/8
P(F) = 1/4
P(G) = 1/16
P(H) = 1/16

出現確率で
並べ替える

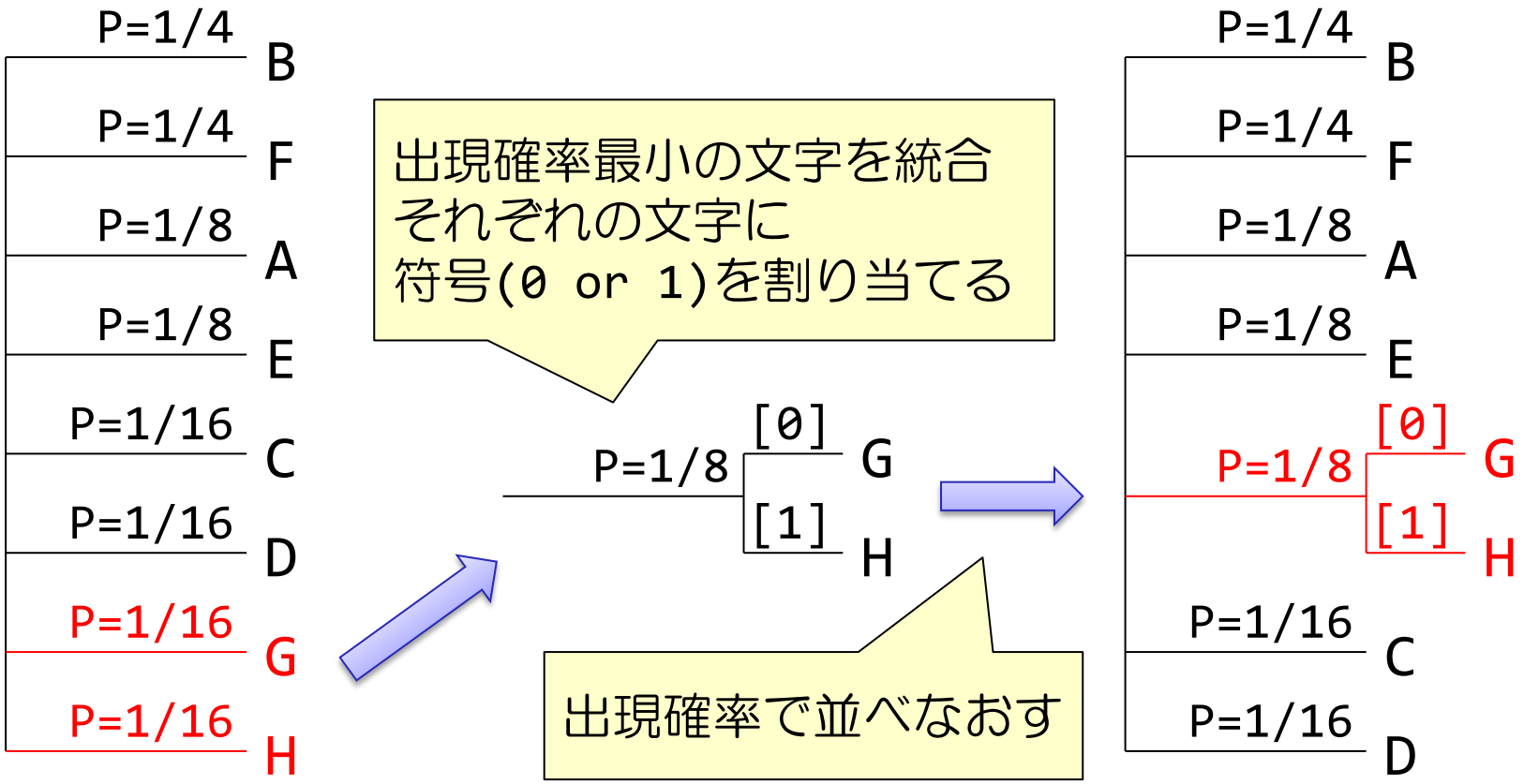
（樹形図で表現）



ハフマン符号の符号化(2)

- ハフマン符号の生成（２）

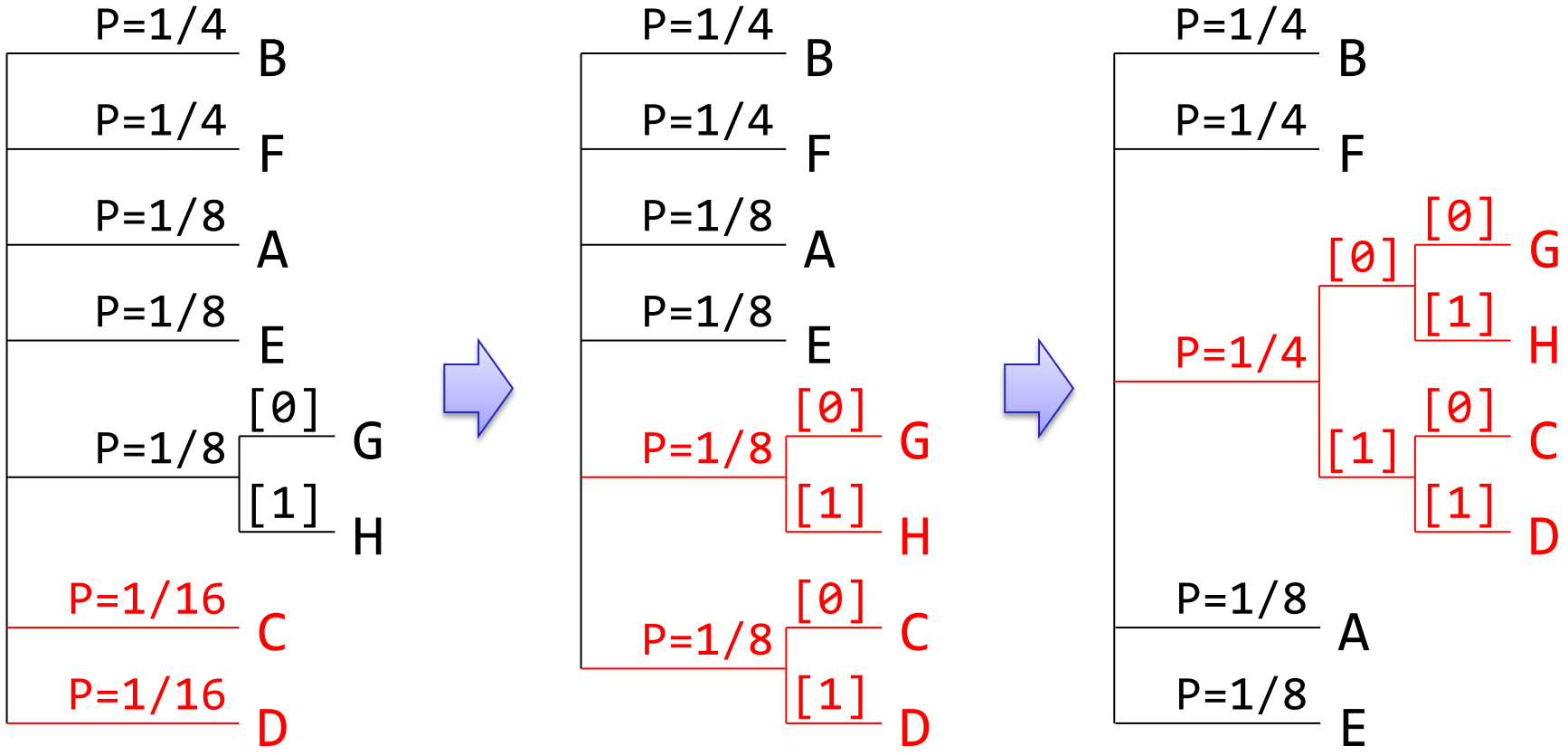
文字列表現　AABBBBCDEEFFFFGH



ハフマン符号の符号化(3)

- ハフマン符号の生成（３）

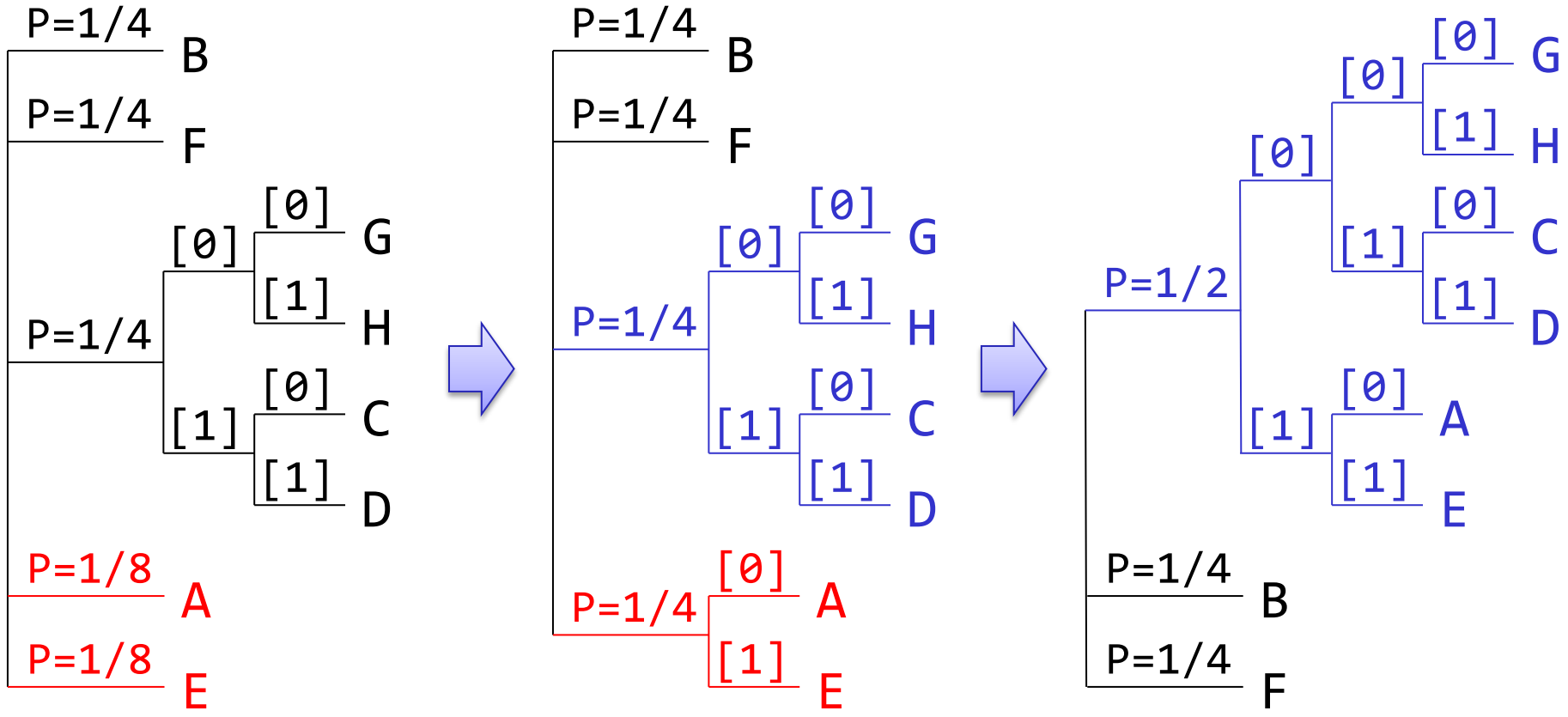
文字列表現　AABBBBCDEEFFFFGH



ハフマン符号の符号化(4)

- ハフマン符号の生成（４）

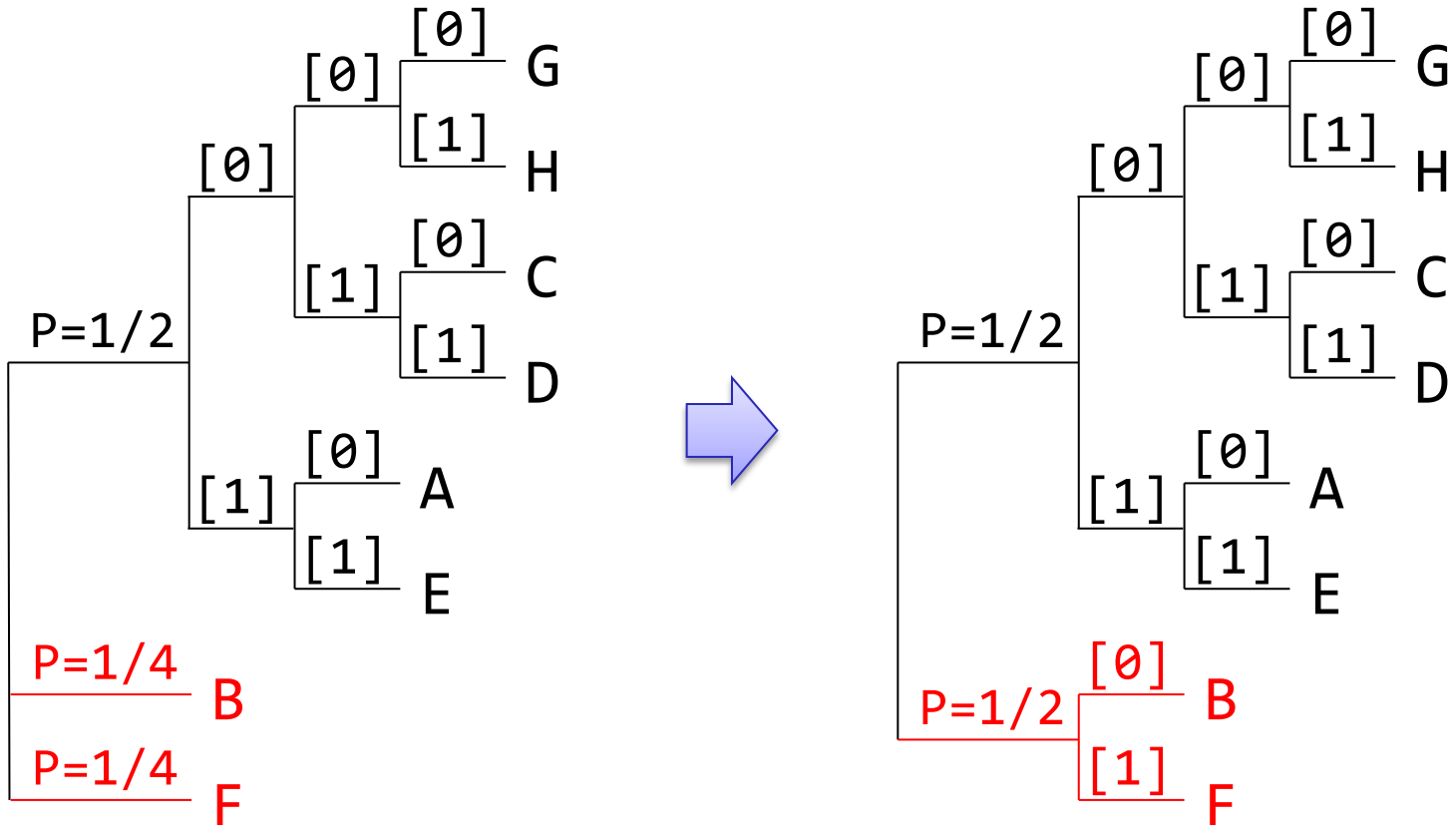
文字列表現　AABBBBCDEEFFFFGH



ハフマン符号の符号化(5)

- ハフマン符号の生成（５）

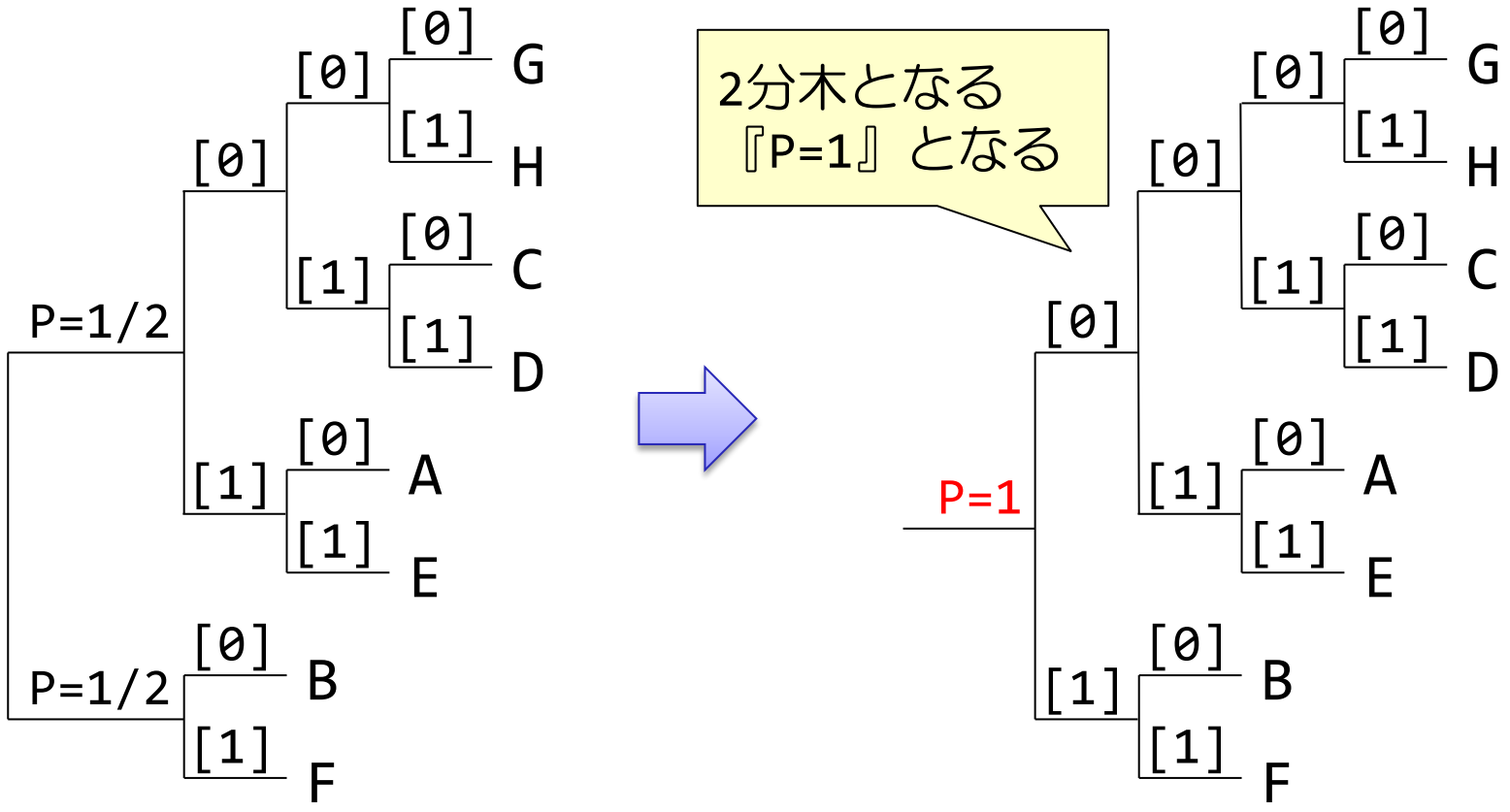
文字列表現　　AABBBBCDEEFFFFGH



ハフマン符号の符号化(6)

- ハフマン符号の生成（６）

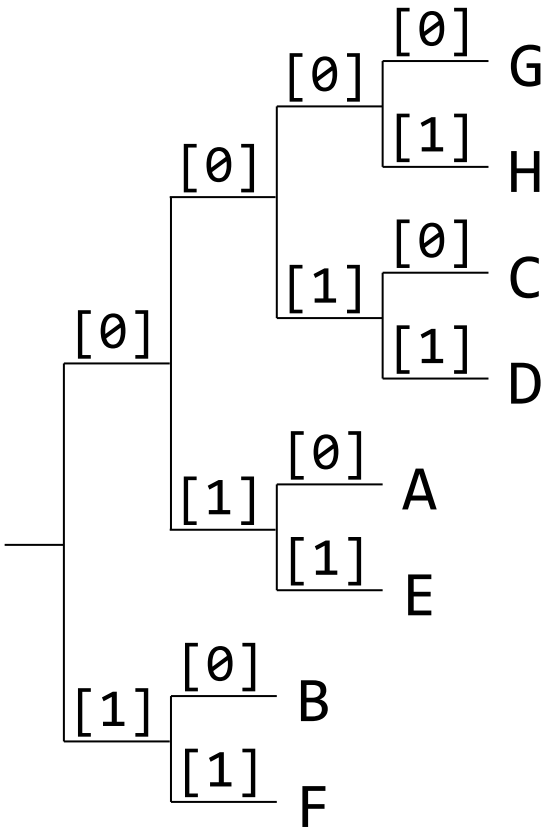
文字列表現　　AABBBBCDEEFFFFGH



ハフマン符号の符号化(7)

- ハフマン符号の生成（７）

文字列表現　　AABBBBCDEEFFFFGH



作成した2文木から
符号を生成する

A = 010_2

B = 10_2

C = 0010_2

D = 0011_2

E = 011_2

F = 11_2

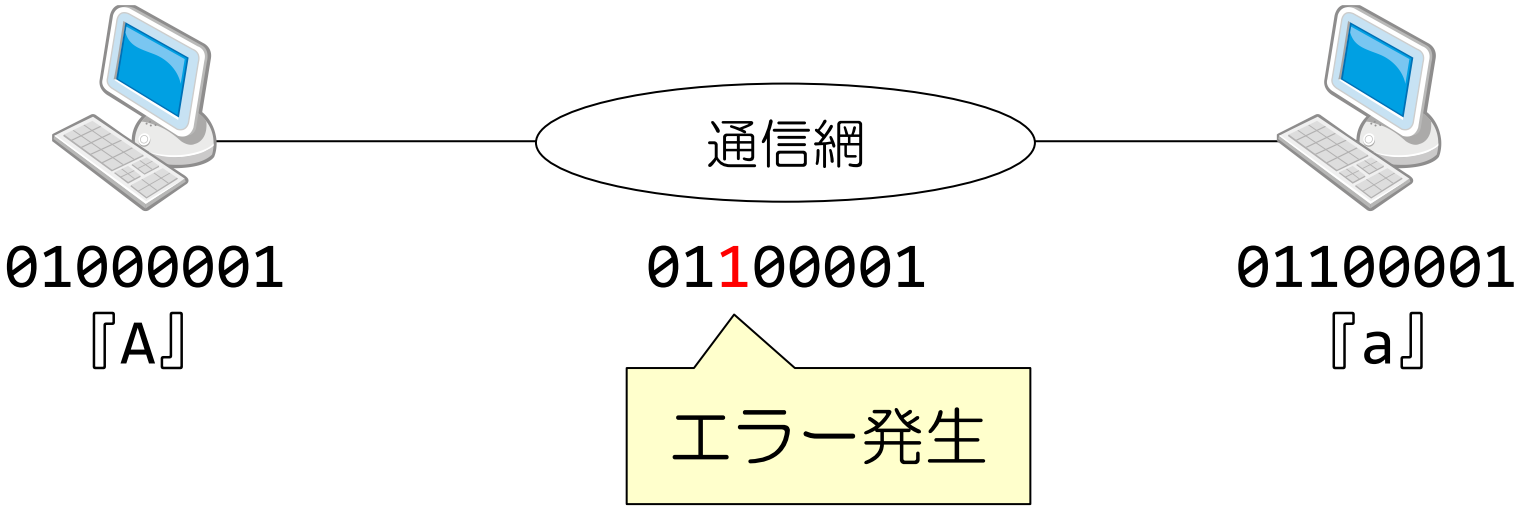
G = 0000_2

H = 0001_2

誤り検出(1)

- データ転送のエラー

データの保存・転送でエラーが発生する場合がある



エラーの発生を検出する方法が必要

誤り検出(2)

- パリティビット

  データにエラー検出用のビットを追加する

01000001**1**　　01100001**0**

データ　パリティ　　データ　パリティ

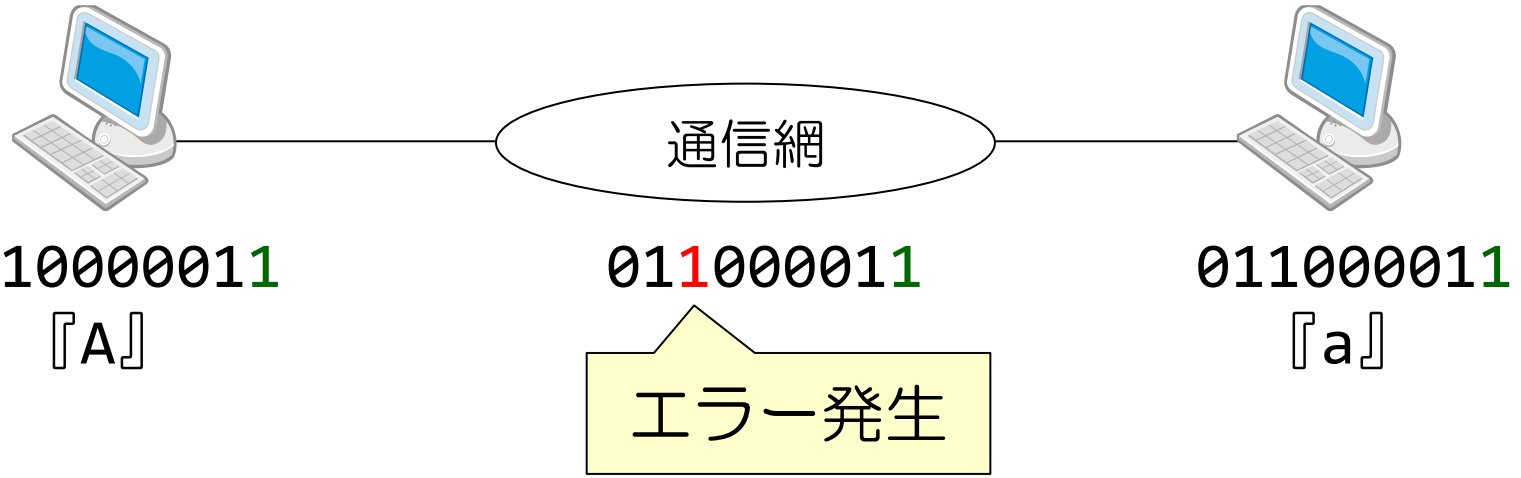
『1』となっているビットの数を奇数にする

⇒ 1ビットまでのエラーを検出できる

（誤り検出符号）

誤り検出(3)

- 誤り検出



（受信データ）
01100001 1

『1』となっているビットの数が偶数
⇒ エラーが発生していることを検出

ハミング距離(1)

- 誤り検出・誤り訂正

符号間の最小ハミング距離を2以上にする

ハミング距離

２個の符号（データ）間の異なるビットの数

01010001
01000111

} ハミング距離 = ３

最小ハミング距離が２以上の場合：誤り検出が可能

最小ハミング距離が３以上の場合：誤り訂正が可能

ハミング距離(2)

- パリティビットのハミング距離

  パリティビットの最小ハミング距離 ＝ ２

  ⇒ 1ビットの誤り検出が可能

符号：(a_1, a_2, a_3, ..., a_n, p) とすると

a_1～a_n のいずれか1個が変化した場合 ⇒ pも変化する

- a_1～a_n によるハミング距離 ＝ 1
- p によるハミング距離 ＝ 1

} ハミング距離 ＝ 2

a_1～a_n のいずれか2個が変化した場合 ⇒ pは変化しない

- a_1～a_n によるハミング距離 ＝ 2
- p によるハミング距離 ＝ 0

} ハミング距離 ＝ 2

ハミング距離(3)

- パリティビットの誤り検出

変化しない場合（ハミング距離 = 0 で変化）

01000001 ⟹ 01000001 （誤りなし）

1ビット変化した場合（ハミング距離 = 1 で変化）

01000001 ⟹ 01010001

01010001 のはず ⇒ 誤り検出

2ビット変化した場合（ハミング距離 = 2 で変化）

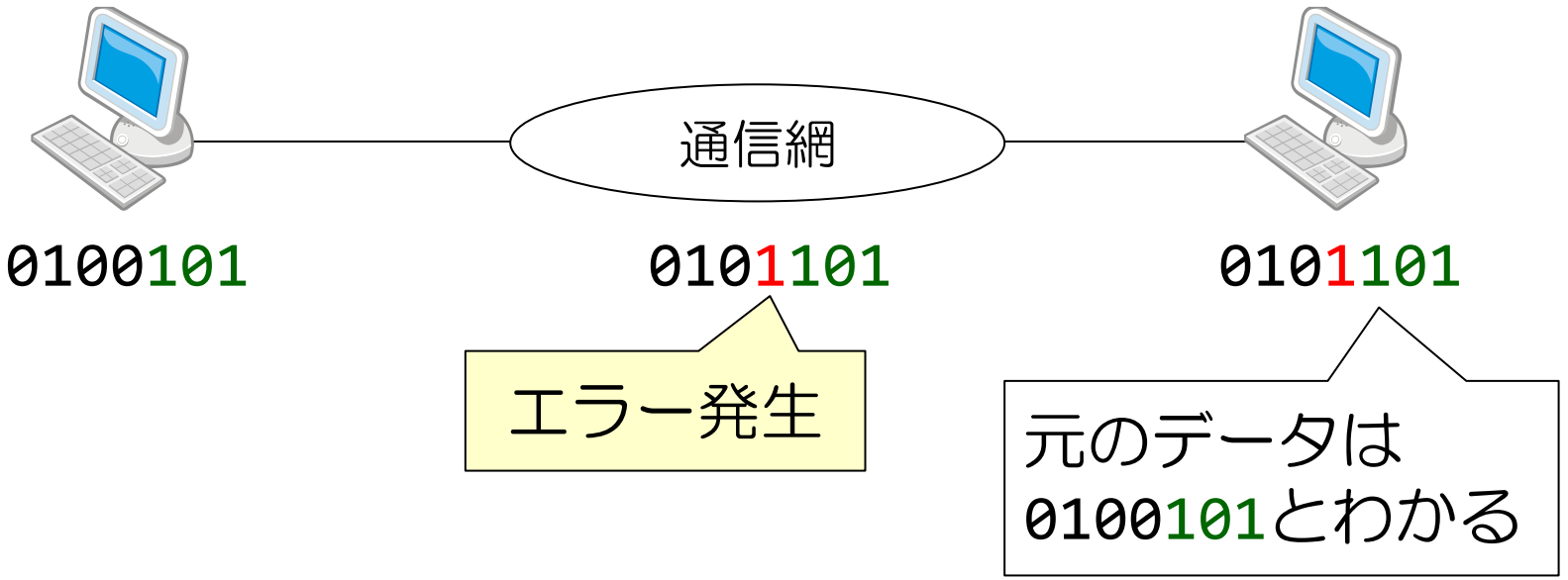
01000001 ⟹ 00010001 （検出不能）

ハミング符号(1)

- ハミング符号

  最小ハミング距離 ＝ 3

  1ビットまでの誤り訂正を行うことができる



ハミング符号(2)

- ハミング符号の生成

データが4ビットの場合 ⇒ ハミング符号は7ビット

0100101　　0101010

データ　訂正符号　　データ　訂正符号

符号：$(a_1, a_2, a_3, a_4, h_1, h_2, h_3)$ とすると

$$h_1 = a_2 + a_3 + a_4$$

$$h_2 = a_1 + a_3 + a_4$$

$$h_3 = a_1 + a_2 + a_4$$

ハミング符号(3)

- ハミング符号（7ビット）の一覧

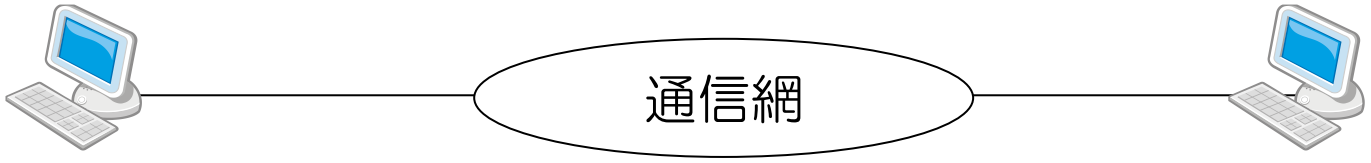
データ	ハミング符号
0000	0000 000
0001	0001 111
0010	0010 110
0011	0011 001
0100	0100 101
0101	0101 010
0110	0110 011
0111	0111 100

データ	ハミング符号
1000	1000 011
1001	
1010	1010 101
1011	1011 010
1100	
1101	1101 001
1110	
1111	1111 111

ハミング符号(4)

- 誤り訂正

  ハミング符号の最小ハミング距離は3となる



0101101
（受信データ）

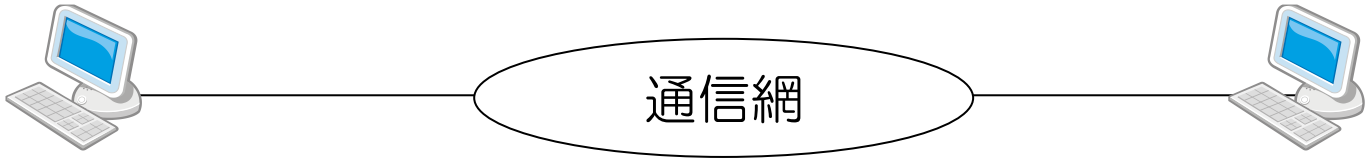
- 0101010:ハミング距離=3
- 1101001:ハミング距離=2
- 0001111:ハミング距離=2
- 0111100:ハミング距離=2
- 0100101:ハミング距離=1

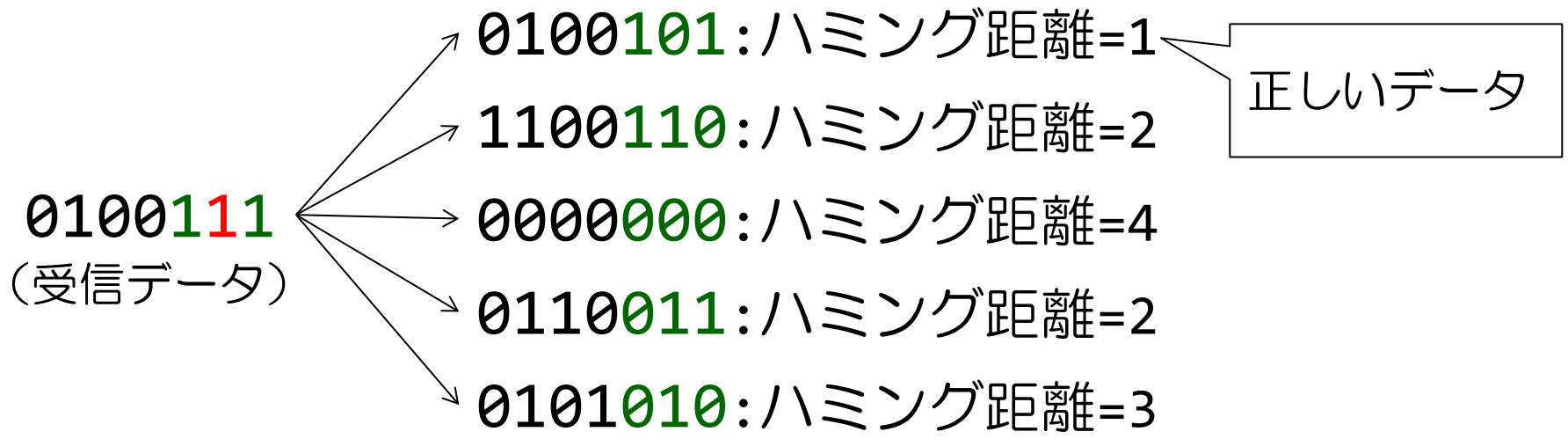
正しいデータ

ハミング符号(5)

- 誤り訂正

  訂正符号が変化しても、正しい符号を取り出せる





練習問題

- 次のハミング符号を訂正し、エラー発生前の符号を求めよ

(1) 1110011

(2) 1100010

誤り検出・誤り訂正

- パリティビット以外にも誤り検出符号はある
  - チェックサム

    インターネット通信におけるパケットの誤り検出
  - ハッシュ（MD5, SHA-1など）

    暗号化通信・電子署名
- ハミング符号以外にも誤り訂正符号はある
  - リード・ソロモン符号

    ORコードの読み込みエラー訂正