

# 基礎情報処理

## 第2回目デジタル情報の世界

2004年10月14日

高等教育研究開発推進センター  
小山田耕二

# Outline

1. コンピュータとはなにか
2. デジタル情報の世界
3. 論理回路からコンピュータまで1
4. 論理回路からコンピュータまで2
5. プログラム基礎1
6. プログラム基礎2
7. データ構造とアルゴリズム1
8. データ構造とアルゴリズム2
9. コンピュータネットワーク
10. 情報倫理
11. さまざまな情報処理
12. コンピュータ科学の諸問題

# 2. デジタル情報の世界

コンピュータ内部でデータを表現する方法としてビットという概念を説明する。  
データ圧縮の基本的考えも紹介する。

Digital information world

## 2.1 ビットと情報量

### 2.1.1 ビット

### 2.1.2 情報量

### 2.1.3 データ圧縮

## 2.2 2進法

### 2.2.1 2進法の数値

### 2.2.2 なぜ2進法か？

### 2.2.3 整数演算

### 2.2.4 補数表現

### 2.2.5 少数点数表現

## 2.3 信頼性向上

### 2.3.1 信頼性

### 2.3.2 エラー検出

### 2.3.3 エラー訂正

# 2.1 ビットと情報量

Bit & information amount

2.1.1 ビット

2.1.2 ビットとバイト

2.1.3 情報量

2.1.4 デジタル化

2.1.5 データ圧縮

# 2.1.1 ビット

Bit

**binary digit**

コンピュータが扱う情報の最小単位

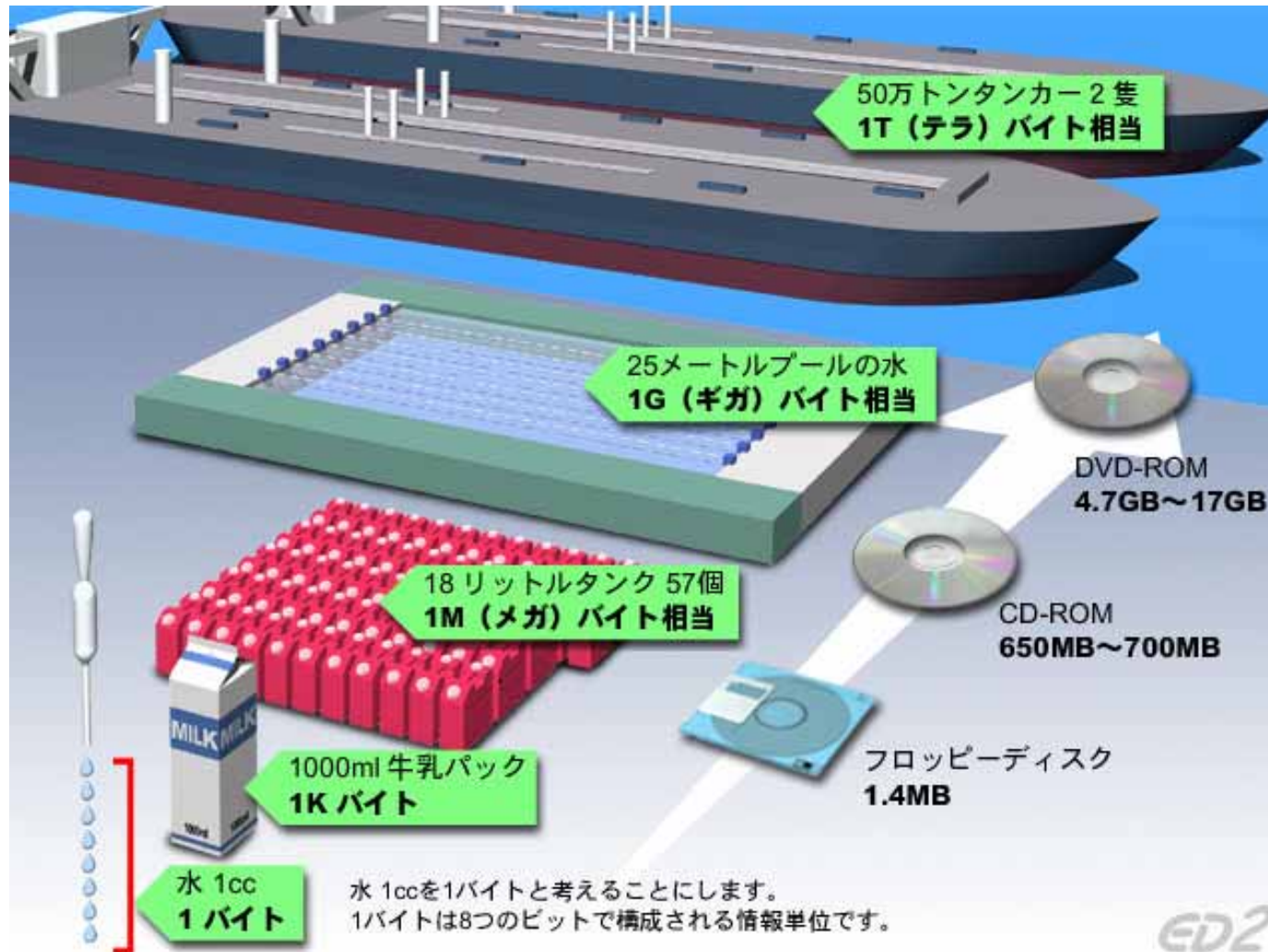
2つの選択肢から1つを特定するのに必要な情報量が1ビット

例) 0と1

一般に、nビットの情報量では2のn乗個までの選択肢からなる情報を表現することができる。

例) アルファベット26文字を表現するのに必要な情報量は  
5ビット( $16 < 26 < 32$ )である

## 2.1.2 ビットとバイト



<http://kyoiku-gakka.u-sacred-heart.ac.jp/jyohou-kiki/2102/2102-A.jpg>

## 2.1.2 ビットパターン

10進数の数え方	2進数の数え方	8ビットの2進数	ランプの点灯で示すと
0	0	00000000	●●●●●●●●
1	1	00000001	●●●●●●●●
2	10	00000010	●●●●●●●●
3	11	00000011	●●●●●●●●
4	100	00000100	●●●●●●●●
5	101	00000101	●●●●●●●●
6	110	00000110	●●●●●●●●
7	111	00000111	●●●●●●●●
8	1000	00001000	●●●●●●●●
9	1001	00001001	●●●●●●●●
10	1010	00001010	●●●●●●●●
11	1011	00001011	●●●●●●●●
12	1100	00001100	●●●●●●●●
⋮	⋮	⋮	⋮
99	01100011	00001111	●●●●●●●●
100	01100100	00010000	●●●●●●●●
101	01100101	00010001	●●●●●●●●
⋮	⋮	⋮	⋮
254	11111110	11111110	●●●●●●●●
255	11111111	11111111	●●●●●●●●
256	100000000		
257	100000001		
⋮	⋮		

↑  
8ビットの2進数では11111111  
(255)までしか数えられません。

ED2

## 2.1.3 情報量

Information amount

### 情報理論(Information theory)の始まり

1948年

“A mathematical theory of communication”

- (1) 情報の量的表示(エントロピー)
- (2) 情報符号化の概念とその限界
- (3) 通信路符号化の概念とその限界



クロード E. シャノン  
(Claude Elwood Shannon)

[http://www.cahners-japan.com/news/200102/20010228belllab\\_shannon.html](http://www.cahners-japan.com/news/200102/20010228belllab_shannon.html)



## 2.1.3 情報量

Information amount

エントロピー (entropy)  $H$

$$H = - \sum_{i=1}^n p_i \log_2 p_i$$

例1) 確率1/2の事柄(コインの表裏)

$$H = -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} = -\frac{1}{2} \times (-1) - \frac{1}{2} \times (-1) = 1$$

例2) 確率1/6の事柄Aと確率5/6の事柄B(1つのAと5つのBのサイコロ)

$$H = -\frac{1}{6} \log_2 \frac{1}{6} - \frac{5}{6} \log_2 \frac{5}{6} = -\frac{1}{6} \times (-2.58) - \frac{5}{6} \times (-0.26) = 0.65$$

小さいほど「あいまいさ」が少ない 確実性が大きい

# 2.1.4 デジタル化

## 文字のデジタル化(1)

上位4ビット  
下位4ビット

**JIS X 0201コード表**

※赤の数字が10進数表記の文字コード  
緑の数字が16進数表記の文字コード

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL 0 00	TC1 1 01	TC2 2 02	TC3 3 03	TC4 4 04	TC5 5 05	TC6 6 06	BEL 7 07	FE0 8 08	FE1 9 09	FE2 10 0A	FE3 11 0B	FE4 12 0C	FE5 13 0D	SO 14 0E	SI 15 0F
1	TC7 16 10	DC1 17 11	DC2 18 12	DC3 19 13	DC4 20 14	TC8 21 15	TC9 22 16	TC10 23 17	CAN 24 18	EM 25 19	SUB 26 1A	ESC 27 1B	IS4 28 1C	IS3 29 1D	IS2 30 1E	IS1 31 1F
2	空白 32 20	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	¥	]	^	
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL
8																
9																
A	。	「	」	、	・	ヲ	ア	イ	ウ	エ	オ	ヤ	ユ	ヨ	ツ	
B	ー	ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ	サ	シ	ス	セ	ソ
C	タ	チ	ツ	テ	ト	ナ	ニ	ヌ	ネ	ノ	ハ	ヒ	フ	ヘ	ホ	マ
D	ミ	ム	メ	モ	ヤ	ユ	ヨ	ラ	リ	ル	レ	ロ	ワ	ン	・	・
E																
F																

制御コード

ASCIIコードに準拠した部分  
(¥はASCIIコードでは\)

未使用

JISで拡張された半角カタカナ部分

未使用

ED2

<http://kyoiku-gakka.u-sacred-heart.ac.jp/jyouhou-kiki/2106/2106-1-A.jpg>

# 2.1.4 デジタル化

## 文字のデジタル化.(2)

文字のデジタル化の例

JIS X 0201コード表に基づいて、1文字ずつ文字コードに置き換える。

This is a pen.

文字コード  
(10進数表記)

84 104 105 83 32 105 83 32 97 32 112 101 110 46

文字列データ

84,104,105,83,32,105,83,32,97,32,112,101,110,46

2進数の文字列データ

1010100,1101000,1101001,1010011,0100000,1101001,1010011,  
0100000,1100001,0100000,1110000,1100101,1101110,0101110

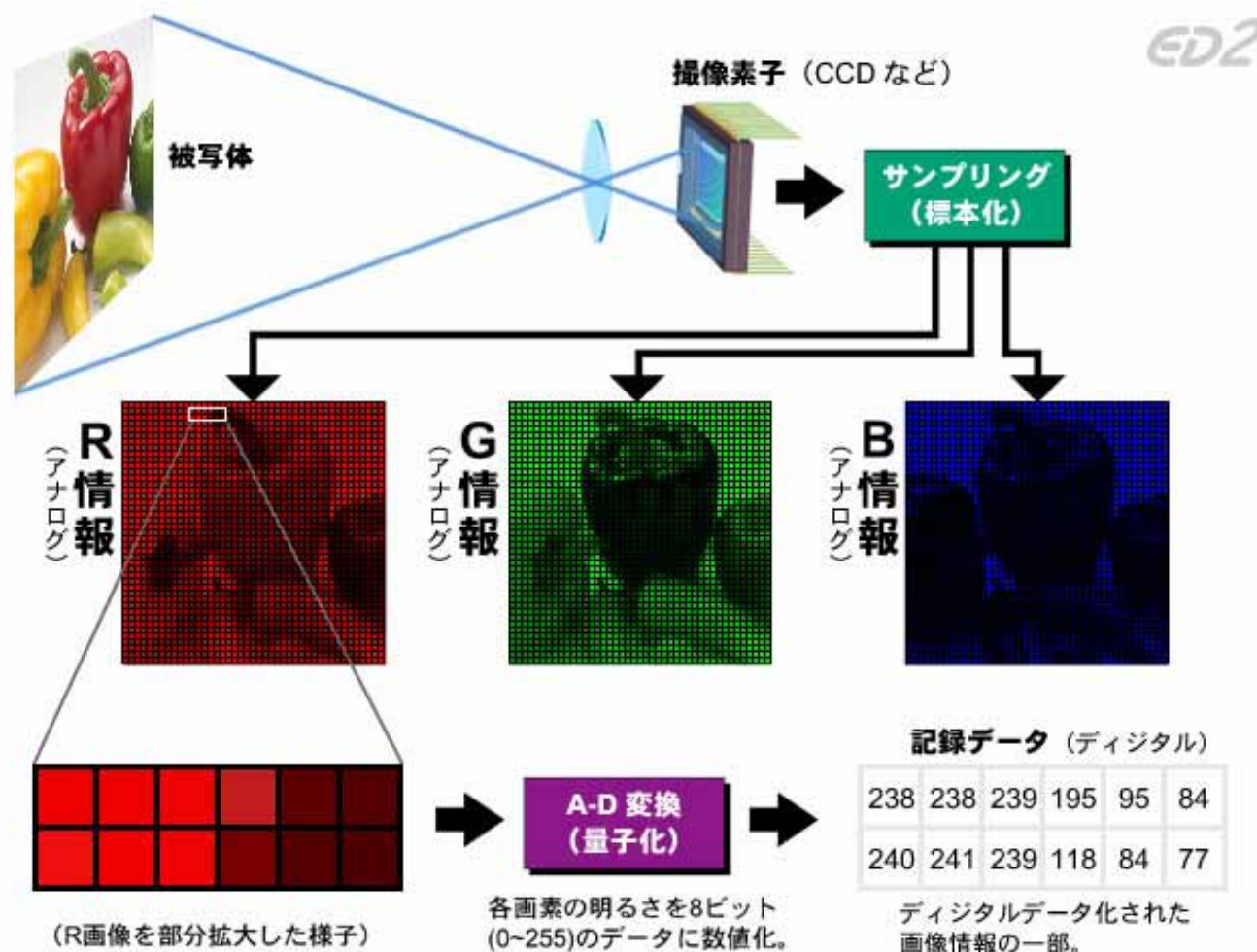
パソコンに記憶



ED2

# 2.1.4 デジタル化

## 画像のデジタル化

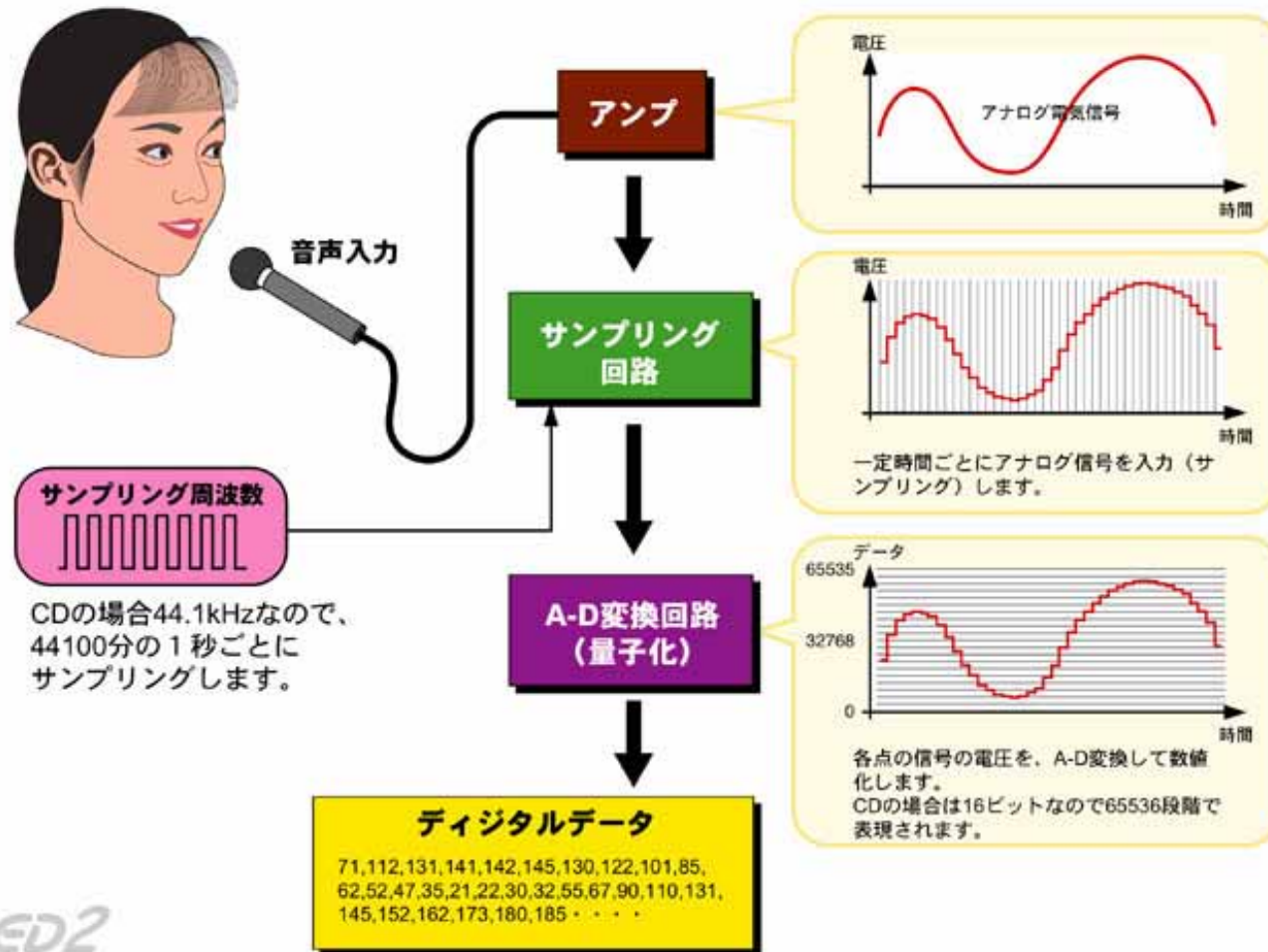


<http://kyoiku-gakka.u-sacred-heart.ac.jp/jyouhou-kiki/2107/2107-A.jpg>



## 2.1.4 デジタル化

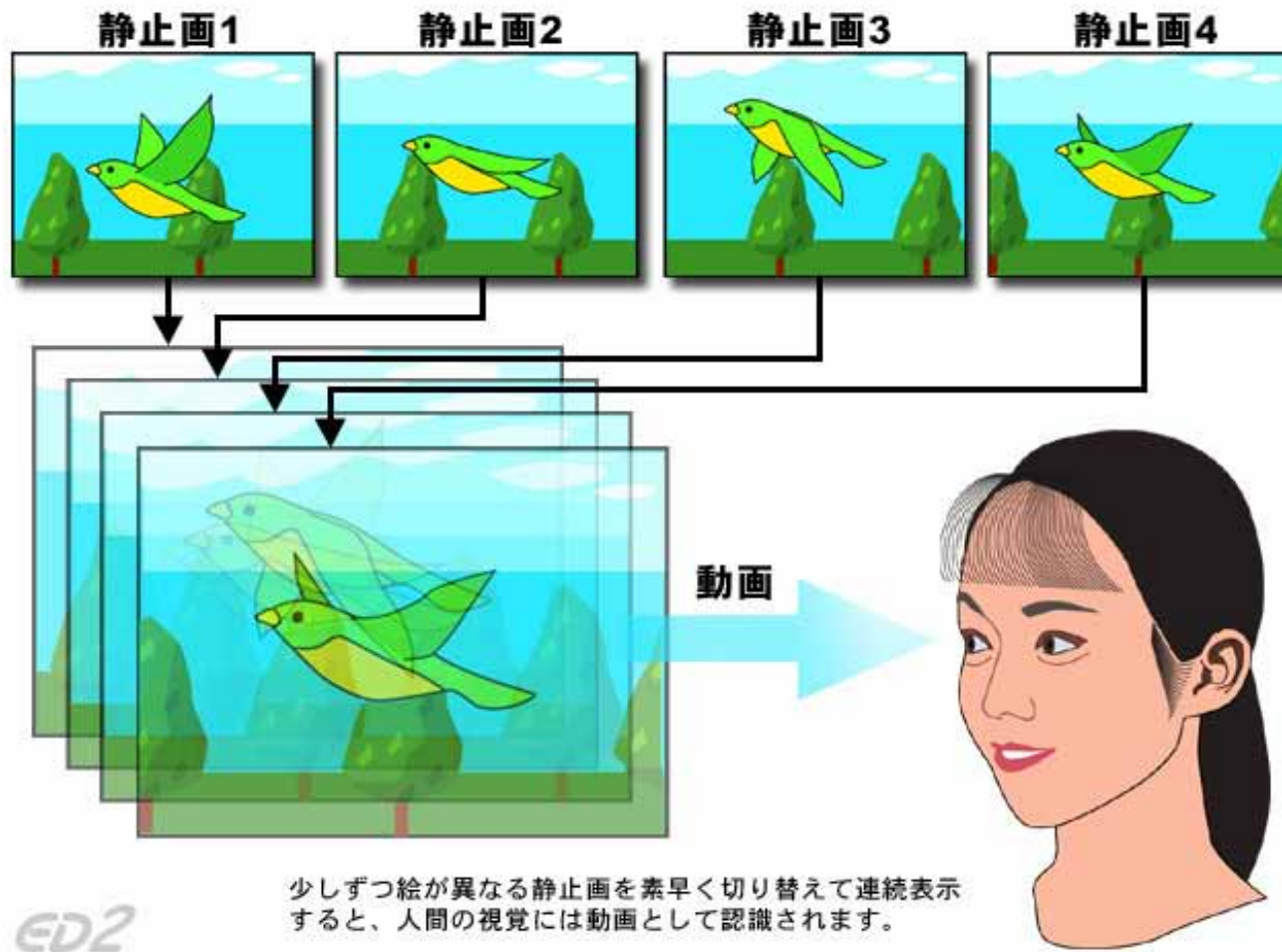
### 音声のデジタル化



ED2

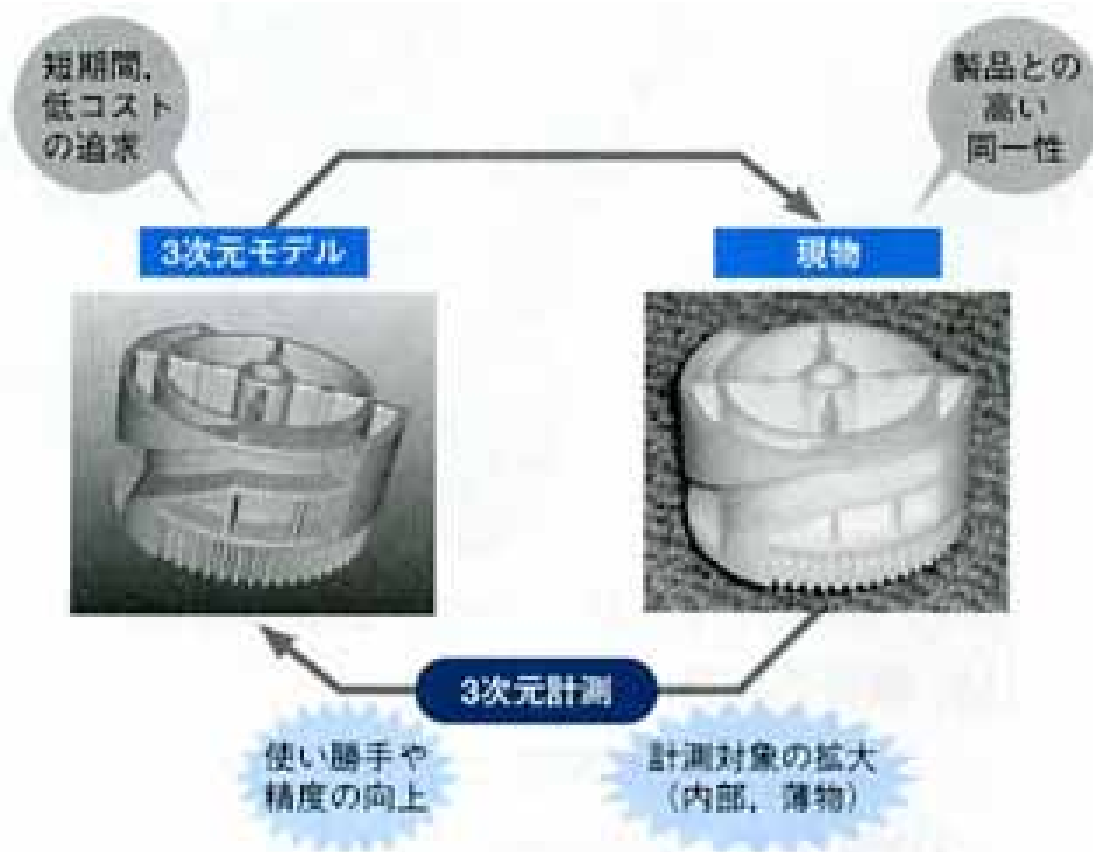
## 2.1.4 デジタル化

### アニメーションのしくみ



## 2.1.4 デジタル化

### 産業への応用(1)



## 2.1.4 デジタル化

### 産業への応用(2)



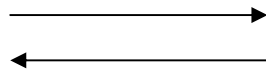


## 2.1.5 データ圧縮

compression

符号化

encode



圧縮


復号化

decode


**データ圧縮**とは、あるデータをそのデータの実質的な性質を保ったまま、データ量を減らした別のデータに変換すること。高効率符号化ともいう。主な目的は、データ転送における通信帯域やトラフィックの減少や、データ蓄積に必要な記憶容量の削減といった、資源の節約である。なお、アナログ技術を用いた通信技術においては通信路の帯域を削減する効果を得るための圧縮ということで帯域圧縮ともいわれた。データ圧縮には大きく分けて可逆圧縮と非可逆圧縮がある。またバイナリデータを対象としたデータ圧縮方式の中には、複数のファイルを一つにまとめて扱えるようにするアーカイブ機能を兼ね備えるものもある。

## 2.1.5 データ圧縮

**静止画**  
(圧縮なし)




画像の大きさが720×480ドットとすると、全体のドット数は345,600ドット。1ドットごとに3バイトのデータ量なので、画像のデータ量は1,036,800バイト＝約1Mバイト。




CD-ROM1枚に約650枚記録可能。

**動画**  
(圧縮なし)




データ量が1Mバイトのフレームを、毎秒30フレームのスピードで表示すると、1秒あたりのデータ量は30Mバイトにもなります。



CD-ROM1枚に約21秒記録可能。

毎秒30フレーム

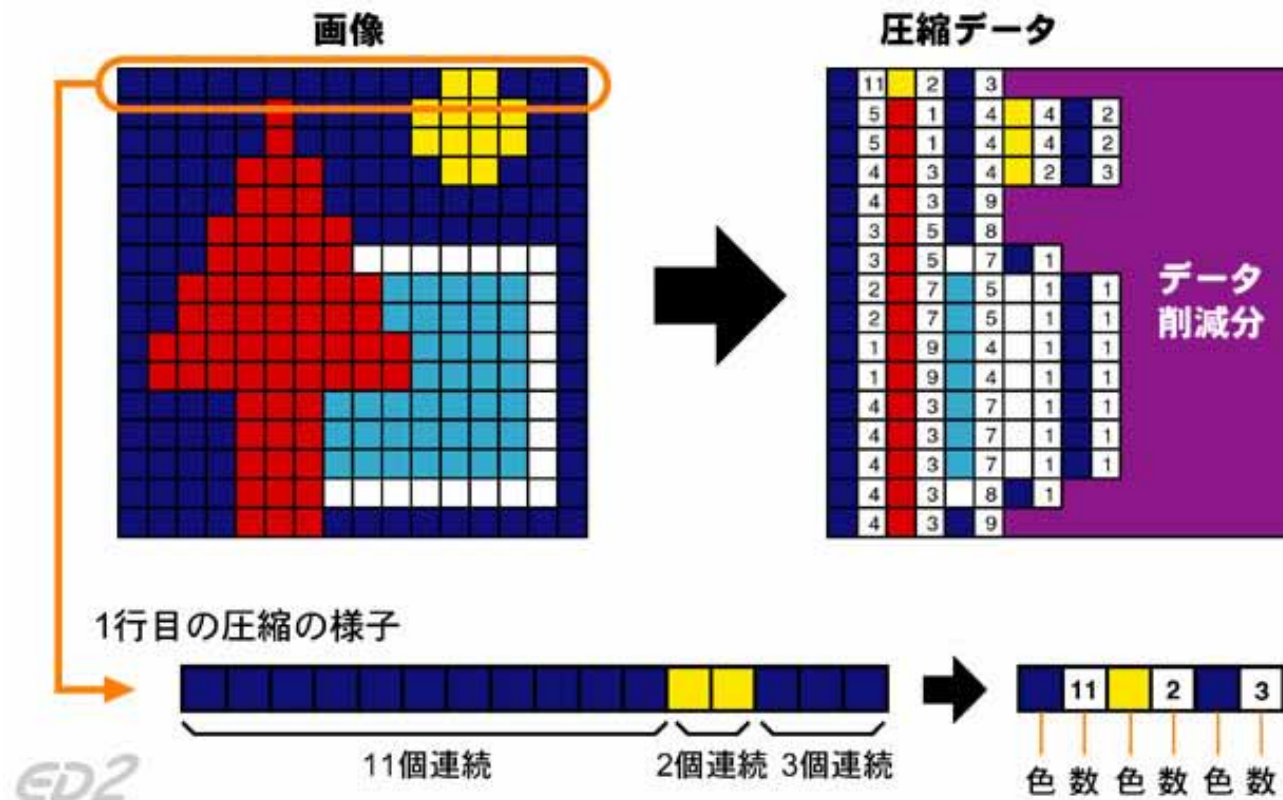


ED2

## 2.1.5 データ圧縮

### データ圧縮のしくみ(1) ラン・レングス法

同じ要素が何個続いているかを数えて、その数をデータ化することで圧縮を行います。



<http://kyoiku-gakka.u-sacred-heart.ac.jp/jyouhou-kiki/2111/2111-1-A.jpg>

## 2.1.5 データ圧縮

### データ圧縮のしくみ(2) ハフマン法

出現頻度の高い要素に短いコードを、出現頻度の低い要素に長いコードを対応させることで、圧縮を行います。

ED2

#### テキストデータ

The Mississippi is the longest river in the United States.

圧縮前のデータ量：57文字×8ビット＝456ビット

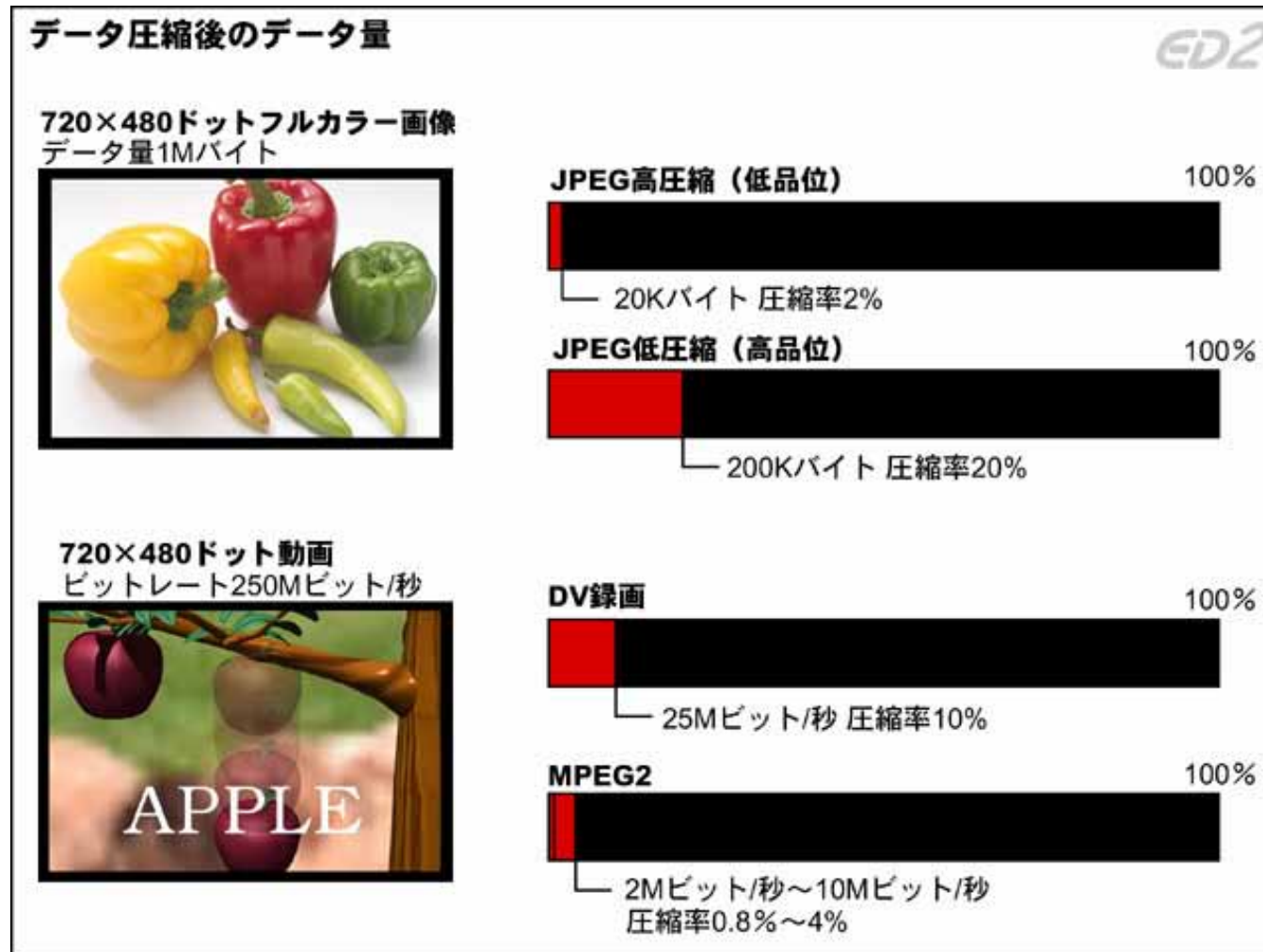
文字	出現数	対応コード
スペース	9	0
i	8	100
e	7	101
s	7	1100
t	6	1101
n	3	11100
h	3	11101
p	2	111100
r	2	111101
l	1	1111100
o	1	1111101
g	1	11111100
v	1	11111101
d	1	111111100
a	1	111111101
T	1	1111111100
M	1	1111111101
U	1	1111111110
S	1	1111111111

左表の対応コードを使って各文字を置き換えた時のデータ量：248ビット

111111110011101101011111110110011001100  
T h eスペース M i s s  
11001100100111100111100100010011000110111101  
s s i p p lスペース sスペース t h  
10101111100 1111101 111100 111111001011100 1101  
eスペース l o n g e s t  
01111011001111110110111110101001110001101  
スペース r i v e rスペース nスペース t  
1110110101111111110111110001110010111111100  
h eスペース U n i t e d  
0111111111111100 111111101 11100 1011100  
スペース S t a t e s



## 2.1.5 データ圧縮



<http://kyoiku-gakka.u-sacred-heart.ac.jp/jyouhou-kiki/2112/2112-A.jpg>

## 2.1.5 データ圧縮

### XVLによる情報統合とCAEへの応用

#### XVLとは

eXtensible  
Virtual world description  
Language

3D データを最大数  
百分の一以下に軽量化  
インターネット環境で  
活用することを目的として開発

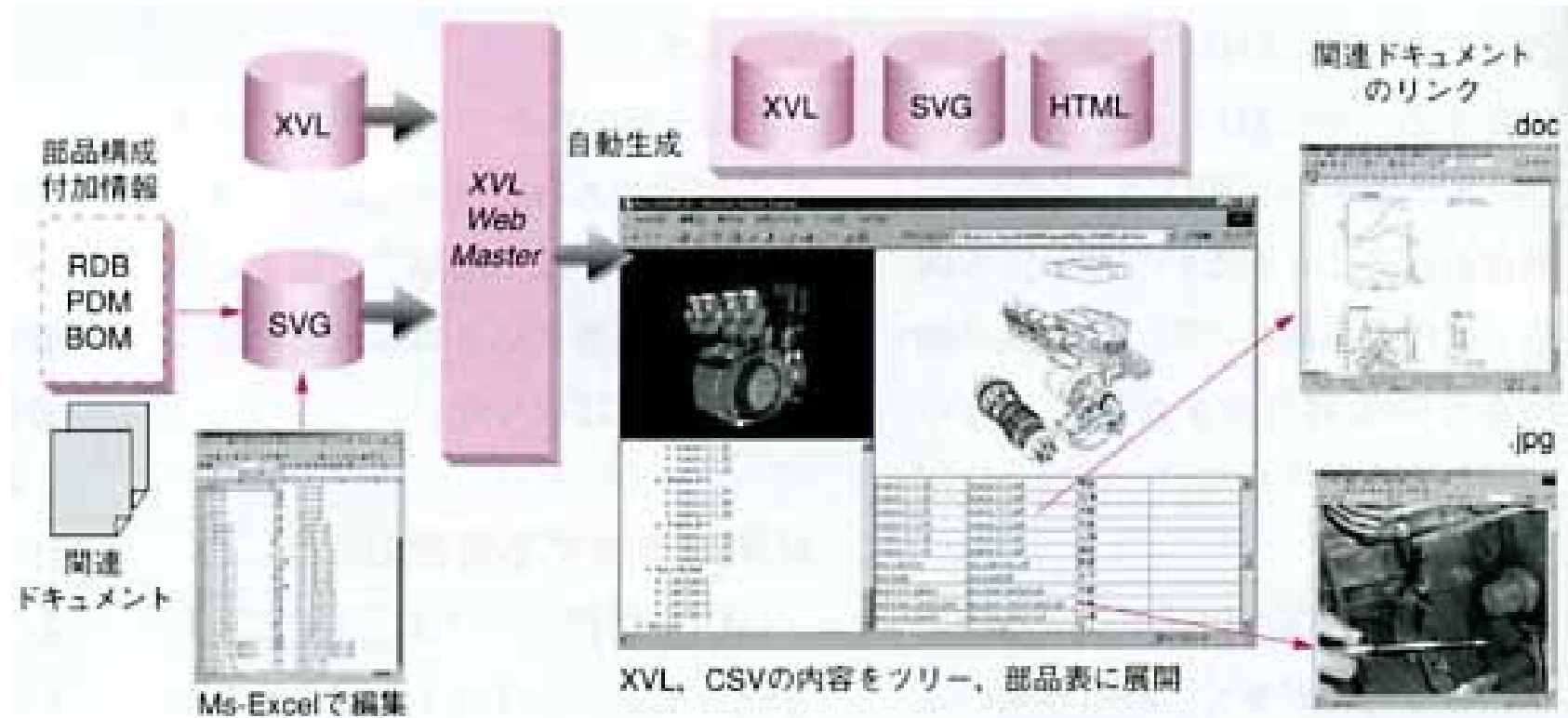


<http://www.xvl3d.com/ja/whatsxvl/index.htm>

<http://www.xvl3d.com/ja/demo/engineering.htm>

## 2.1.5 データ圧縮

### XVLによる情報統合とCAEへの応用



XVLからWeb3Dとして情報統合データを生成するXVL Web Master

## 2.1.5 データ圧縮

### XVLによる情報統合とCAEへの応用

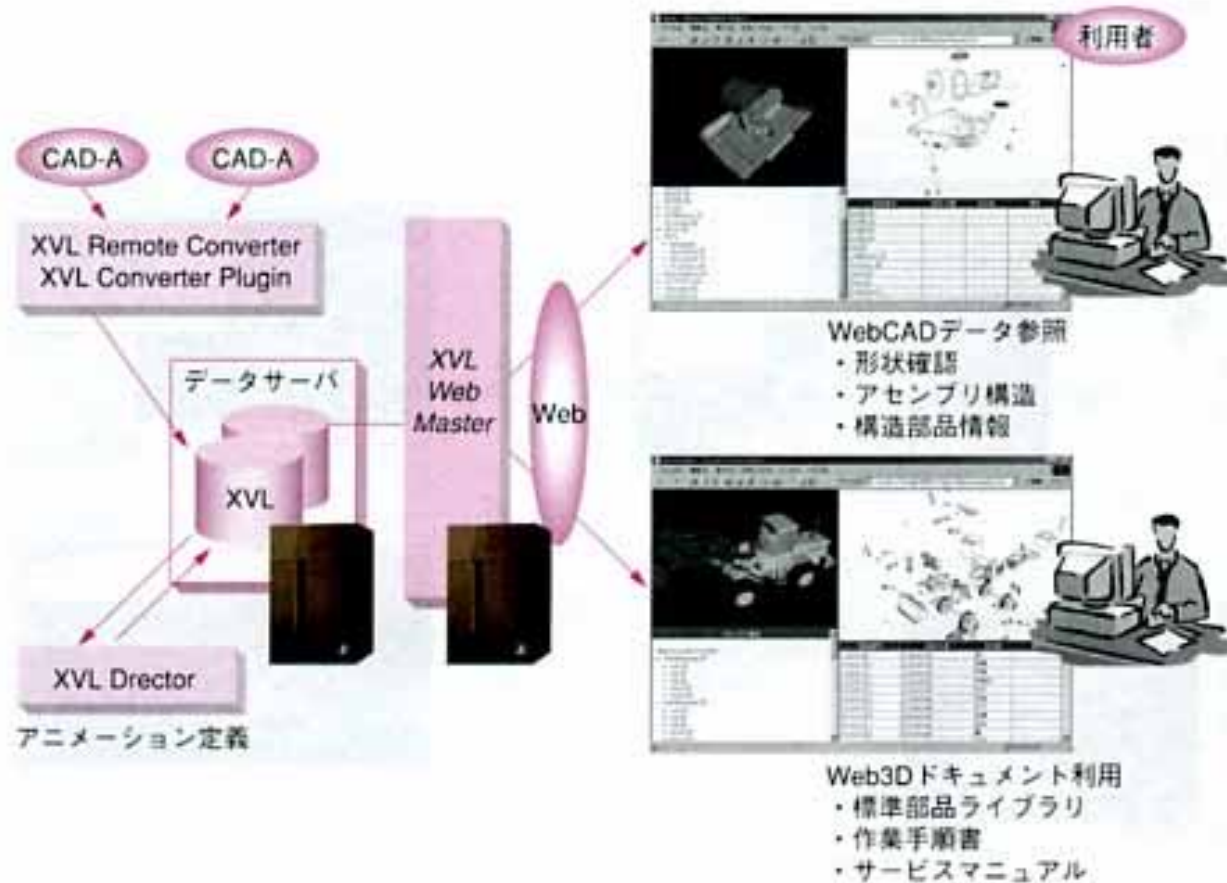


XVLからXML形式である2Dベクタフォーマット:SVGを自動出力する



## 2.1.5 データ圧縮

### XVLによる情報統合とCAEへの応用



XVL Web Masterの活用例

## 2.2 2進法

2.2.1 2進法の数値

2.2.2 なぜ2進法か？

2.2.3 整数演算

2.2.4 補数表現

2.2.5 少数点数表現

## 2.2.1 2進法の数値

2進法 (binary system)

10進法 (decimal system)

16進法 (hexadecimal system)

2進法	10進法	16進法
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

## 2.2.2 なぜ2進法か？

2進数は、コンピュータで数値を正確に表現するのに適している

### 理由

- 1) 信頼性
- 2) 論理回路は2進数 (yes or no)
- 3) 演算回路の簡潔さ (10進法より少ない)

(問) 2進数がコンピュータの演算に適している上記以外の理由を考えよ

## 2.2.3 整数演算

4通り

加算	減算	乗算	除算
$0+0=0$	$0-0=0$	$0 \times 0=0$	$0 \div 0=\text{invalidity}$
$0+1=1$	$0-1=-1$	$0 \times 1=0$	$0 \div 1=0$
$1+0=1$	$1-0=1$	$1 \times 0=0$	$1 \div 0=\text{invalidity}$
$1+1=10$	$1-1=0$	$1 \times 1=1$	$1 \div 1=1$

## 2.2.4 補数表現

complement

「1の補数」と「2の補数」

「1の補数」・・・正の数の2進数表現の0と1をすべて反転

「2の補数」・・・1の補数にさらに1を加えたもの

負の数の表現

例) - 3の「2の補数」表現

0011 (反転) 1100 (1加える) 1101

## 2.2.5 少数点数表現

浮動小数点 (floating point) 方式

$$a = m \times 2^e$$

数

仮数部  
(mantissa)

基底  
(base)

指数部  
(exponent)

## 2.3 信頼性向上

2.3.1 信頼性

2.3.2 エラー検出

2.3.3 エラー訂正



## 2.3.1 信頼性

reliability

デジタル技術 > アナログ技術

例) 完璧に近い信頼性が要求される

銀行の預金管理

新幹線の運行管理・座席指定管理

発電所・電力網の管理

シャノンの情報理論における信頼性

## 2.3.2 エラー検出

### パリティ検査 (parity check)

1バイトあたりの信号に誤りが含まれているか否かを調べる方法

#### 奇数パリティチェック (1の個数が奇数個)

0 1 0 0 0 0 0 1 1

↑  
データ  
(2個)

↑  
パリティビット  
(1にすると合計3個になり, 奇数個になる。)

#### 偶数パリティチェック (1の個数が偶数個)

0 1 0 0 0 0 0 1 0

↑  
データ  
(2個)

↑  
パリティビット  
(0にすると合計2個になり, 偶数個になる。)

## 2.3.2 エラー検出

### パリティ検査 (parity check)

例 1) 奇数パリティチェックを採用  
送信側で次のデータ送信した

0	1	0	0	0	0	0	1
---	---	---	---	---	---	---	---

1
---

受信側で次のデータ受信した

0	1	0	0	0	0	0	0
---	---	---	---	---	---	---	---

1
---

1の個数が2個 途中でビットが反転



誤り検出

受信側はどのビットが反転したかまでは判定できない 誤り訂正は出来ない

## 2.3.2 エラー検出

### パリティ検査 (parity check)

例2) 奇数パリティチェックを採用  
送信側で次のデータ送信した

0	1	0	0	0	0	0	1
---	---	---	---	---	---	---	---

1
---

(問) 10進数でこの  
検査を行うことを  
考えて見なさい

受信側で次のデータ受信した

0	1	0	0	1	0	0	0
---	---	---	---	---	---	---	---

1
---

1の個数が3個 途中で2つのビットが反転



誤り検出出来ない

ビット反転確率 $p$ ならば、2つの反転が起こる確率は $p^2$ と小さい

## 2.3.3 エラー訂正

Error correction

### ハミング符号 (Hamming code)

コンピュータ内部の回路間や、通信回線を使ったコンピュータ同士のデータ通信において、データの誤り(エラー)を検出する手法の一つ。誤りを検出するだけでなく、正しい値に訂正することもできる。

1950年にアメリカのベル研究所のHamming氏によって考案された。本来のデータに、データから演算によって割り出したチェック用のデータ(ハミングコード)を付加して転送する。

広く普及しているパリティチェックは、ハミングコードチェックの特殊な場合にあたる。RAID-2の誤り訂正符号に採用されている。

## 2.3.3 エラー訂正

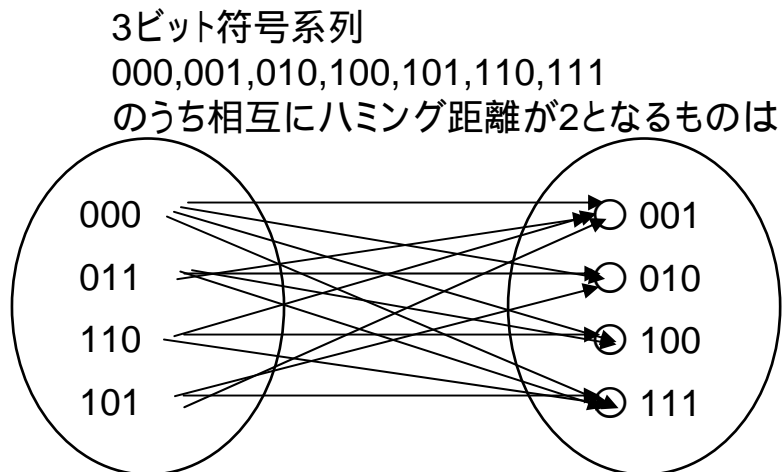
### ハミング距離 (Hamming distance)

長さ $n$ ビットの2元(0,1)の符号系列 $X, Y$ に対して  
次式でハミング距離が定義される

$$\begin{cases} d(X, Y) = (x_i \oplus y_i) \\ X = x_1 x_2 \cdots x_n (x_i=0,1) \\ Y = y_1 y_2 \cdots y_n (y_i=0,1) \end{cases}$$

ここで、演算子  $\oplus$  は排他的論理和であり、次のような演算を表す

$$\left\{ \begin{array}{l} 0 \oplus 0 = 0 \\ 0 \oplus 1 = 1 \\ 1 \oplus 0 = 1 \\ 1 \oplus 1 = 0 \end{array} \right.$$



## 2.3.3 エラー訂正

### ハミング符号 (Hamming code)

4個の情報ビット(a, b, c, d)  
検査ビット(e, f, g)をつくる

$$\begin{cases} e = b \oplus c \oplus d \\ f = a \oplus c \oplus d \\ g = a \oplus b \oplus d \end{cases} \quad (1)$$

という符号語に符号化

$$= (a, b, c, d, e, f, g)$$

単一の誤りを訂正することができる

ハミング符号語

a	b	c	d	e	f	g
0	0	0	0	0	0	0
0	0	0	1	1	1	1
0	0	1	0	1	1	0
0	0	1	1	0	0	1
0	1	0	0	1	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	0	1	1
1	0	0	1	1	0	0
1	0	1	0	1	0	1
1	0	1	1	0	1	0
1	1	0	0	1	1	0
1	1	0	1	0	0	1
1	1	1	0	0	0	0
1	1	1	1	1	1	1

## 2.3.3 エラー訂正

### ハミング符号 (Hamming code)

検査ビット			
s1	s2	s3	誤りの位置
0	0	0	誤りなし
0	0	1	a
0	1	0	b
0	1	1	c
1	0	0	d
1	0	1	e
1	1	0	f
1	1	1	g

$$\begin{cases} s1 = d \oplus e \oplus f \oplus g \\ s2 = b \oplus c \oplus f \oplus g \\ s3 = a \oplus c \oplus e \oplus g \end{cases}$$

$$\begin{cases} 0 = d \oplus e \oplus f \oplus g & (2) \\ 0 = b \oplus c \oplus f \oplus g & (3) \\ 0 = a \oplus c \oplus e \oplus g & (4) \end{cases}$$

検査ビットe,f,gに関する連立方程式とみなし、e,f,gについて解く

$$(2)+(3) \text{より} \quad 0 = d + e + f + g + b + c + f + g$$

$$\text{順序を入れ替えて} \quad 0 = b + c + d + e + f + f + g + g$$

$$f+f=0, g+g=0 \text{なので} \quad 0 = b + c + d + e$$

$$\text{両辺にeを加えて} \quad e = b + c + d$$

$$e = b \oplus c \oplus d$$

$$f = a \oplus c \oplus d$$

$$g = a \oplus b \oplus d$$



## 2.3.3 エラー訂正

### ハミング符号 (Hamming code)

受信語 $y$ は  $a$  と  $n$  (誤りまたはノイズ) の和である

$$\begin{aligned} y &= a + n \\ &= (a + n_1, b + n_2, \dots, g + n_7) \end{aligned}$$

$$s_1 = (d+n_4) + (e+n_5) + (f+n_6) + (g+n_7) = (d+e+f+g) + (n_4+n_5+n_6+n_7) = n_4+n_5+n_6+n_7$$

$$s_2 = (b+n_2) + (c+n_3) + (f+n_6) + (g+n_7) = (b+c+f+g) + (n_2+n_3+n_6+n_7) = n_2+n_3+n_6+n_7$$

$$s_3 = (a+n_1) + (c+n_3) + (e+n_5) + (g+n_7) = (a+c+e+g) + (n_1+n_3+n_5+n_7) = n_1+n_3+n_5+n_7$$

## 2.3.3 エラー訂正

### ハミング符号 (Hamming code)

単一誤りに対するシンドローム

誤りパターン							シンドローム		
n1	n2	n3	n4	n5	n6	n7	s1	s2	s3
1	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	1	1
0	0	0	1	0	0	0	1	0	0
0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	1	1	1	1

すべての単一誤りに対して、シンドロームパターンが異なる

シンドロームパターンから、単一誤り位置が判る **誤り訂正可能**

## 2.3.3 エラー訂正

### ハミング符号 (Hamming code)

あ～た までの16個のひらがなを用いてデータを  
送信する。このとき、  
1010110  
という符号系列を受け取ったとき

S1=0

S2=0

S3=1

$$s1 = d \oplus e \oplus f \oplus g$$

$$s2 = b \oplus c \oplus f \oplus g$$

$$s3 = a \oplus c \oplus e \oplus g$$

となり、左から1ビット目に誤りがあることが判明

0010110

と訂正され、“う”であることがわかる

### ハミング符号語

	a	b	c	d	e	f	g
あ	0	0	0	0	0	0	0
い	0	0	0	1	1	1	1
う	0	0	1	0	1	1	0
え	0	0	1	1	0	0	1
お	0	1	0	0	1	0	1
か	0	1	0	1	0	1	0
き	0	1	1	0	0	1	1
く	0	1	1	1	1	0	0
け	1	0	0	0	0	1	1
こ	1	0	0	1	1	0	0
さ	1	0	1	0	1	0	1
し	1	0	1	1	0	1	0
す	1	1	0	0	1	1	0
せ	1	1	0	1	0	0	1
そ	1	1	1	0	0	0	0
た	1	1	1	1	1	1	1

# 小テスト(氏名: )

- 前述のひらがなデータを用いるとき、次の符号系列を受信したとする。このとき送信された元データを示せ。
- 00011110111010101010011010