



コンピュータ・サイエンス1

第11回 データ圧縮

人間科学科コミュニケーション専攻
白銀 純子

第11回の内容

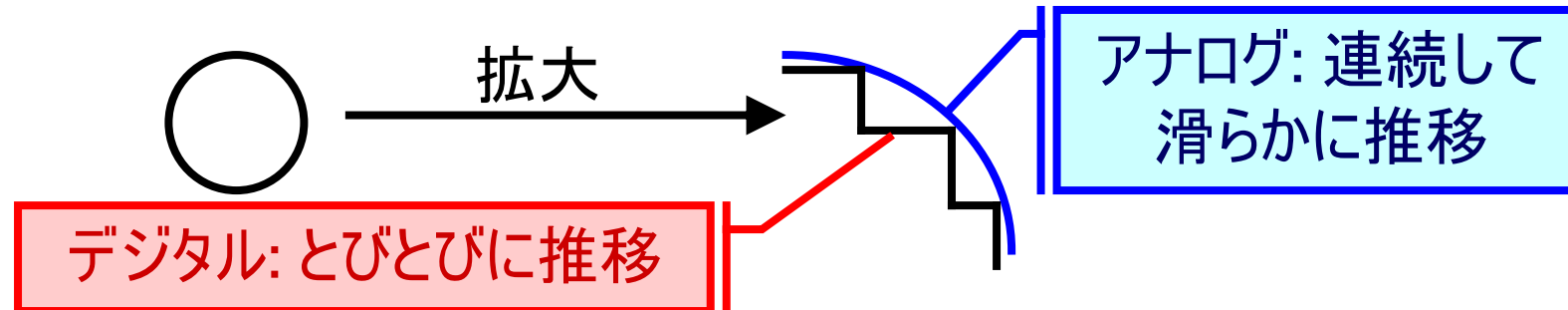
- データ圧縮



前回の復習

アナログ情報とデジタル情報(p. 19)

- アナログ情報: 連続的な数値で表現できる情報
 - 物事を表現する数値の桁数が無限
 - Ex. アナログ時計: 1秒から2秒になるまで秒針が止まらず動き続ける(1秒と2秒の間も1.xxxx....秒が存在する)
- デジタル情報: 離散的な数値(とびとびの数値)で表現される情報
 - 物事を表現する数値の桁数が有限
 - Ex. デジタル時計: 1秒の次は2秒(1秒と2秒の間がない)



アナログ/デジタル情報の例(p. 19)

- アナログ情報
 - 現実世界の音
 - カセットテープやレコードに記録された音情報
 - 人の手で描いた絵
 - フィルムに記録された映像
- デジタル情報
 - CDやMDに記録された音情報
 - デジタルカメラが記録した画像情報
 - DVDに記録された音と動画の情報

アナログ情報のデジタル化(p. 20)

- アナログ情報: 数値化すると連続的

- 音の強弱の変化
- 画像の色の濃さ光の強弱の変化

グラフで表すと、なめらかな曲線(正弦波) = アナログ信号

デジタル化するには...

- 標本化
- 量子化

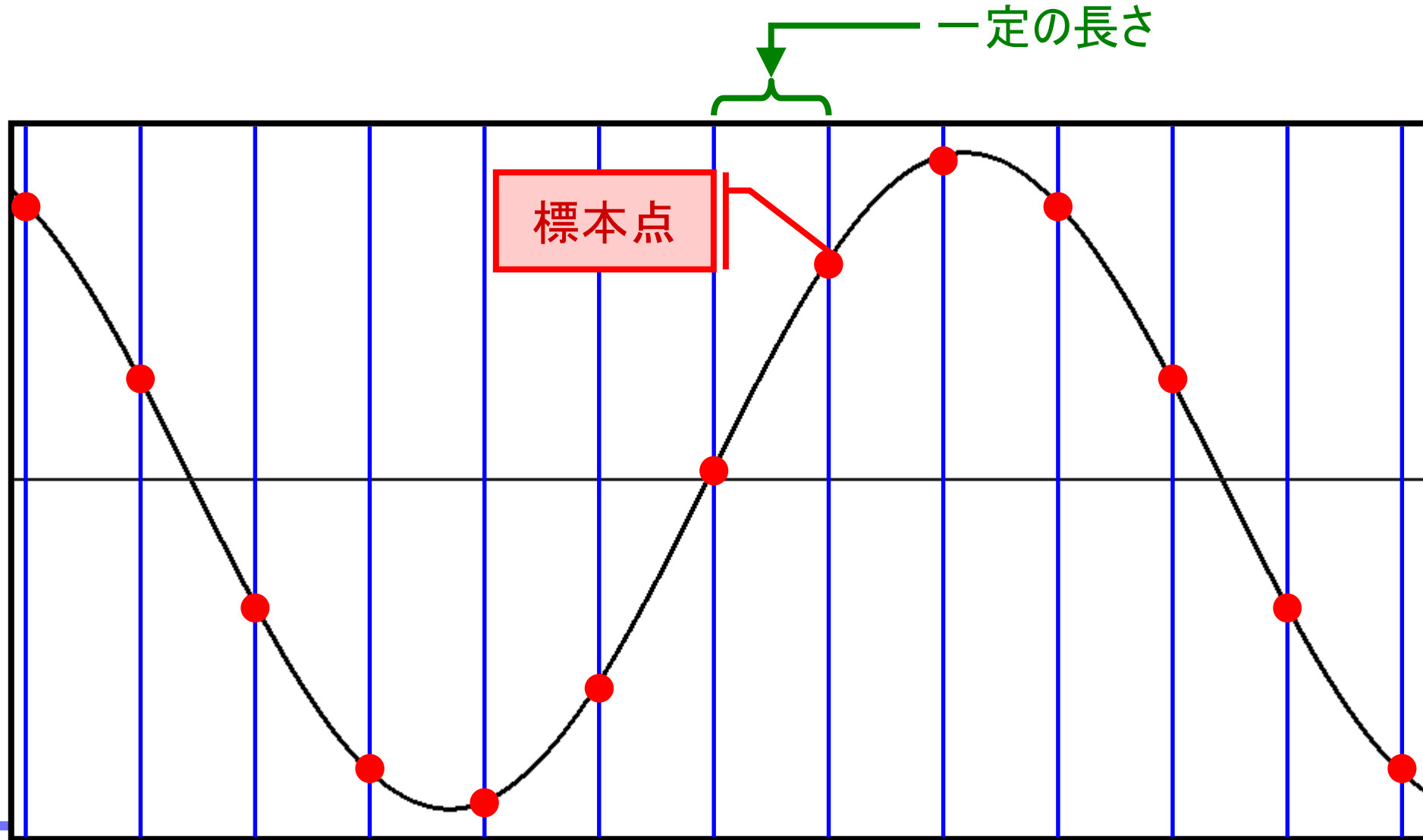
標本化[1](p. 20)

- **標本化**: 横軸の一定の長さごとに、縦軸の値を調べること
 - 調べた点を「**標本点**」と呼ぶ
- 調べた以外のところの値は使用しない
 - 標本点の間隔を十分に細かくとれば、もとのアナログ信号の情報を完全に保持できる

どのくらいの間隔で標本化をすれば良いか???

標本化[2](p. 20)

- 標本化: 横軸を一定の長さで区切って縦軸の値を調べること

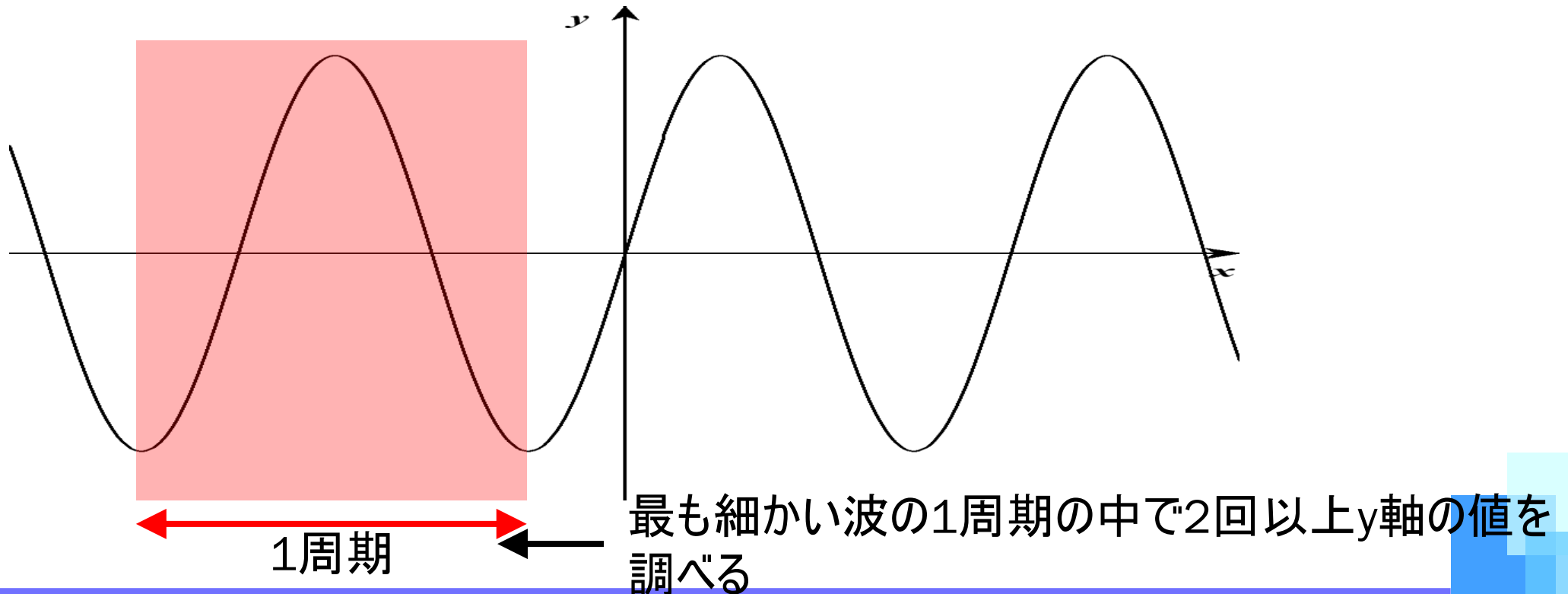


標本化[4](p. 20)

■ 標本化定理

- 波の中のもっとも細かい正弦波の周期の半分よりも狭い間隔で標本化すれば、標本から元の波を再現可能

アナログ信号の最も細かい波の周期で2回以上値を調べれば良い



標本化[5](p. 20)

- 1周期の半分より小さい間隔で標本点をとれば、元の波形を正確に表現可能(=方程式を算出可能)
- 1周期の半分より大きい間隔で標本点をとれば、**元の波形とは異なった波形が出現**

エイリアシング

音や画像の情報: 波形の周期の長いもの・短いものを
重ね合わせることで表現可能



波形の中で最も短い周期の半分よりも
狭い間隔で標本化することが重要

量子化[1](p. 22)

■ **量子化**: 縦軸の一定の長さごとに、横軸の値を調べること

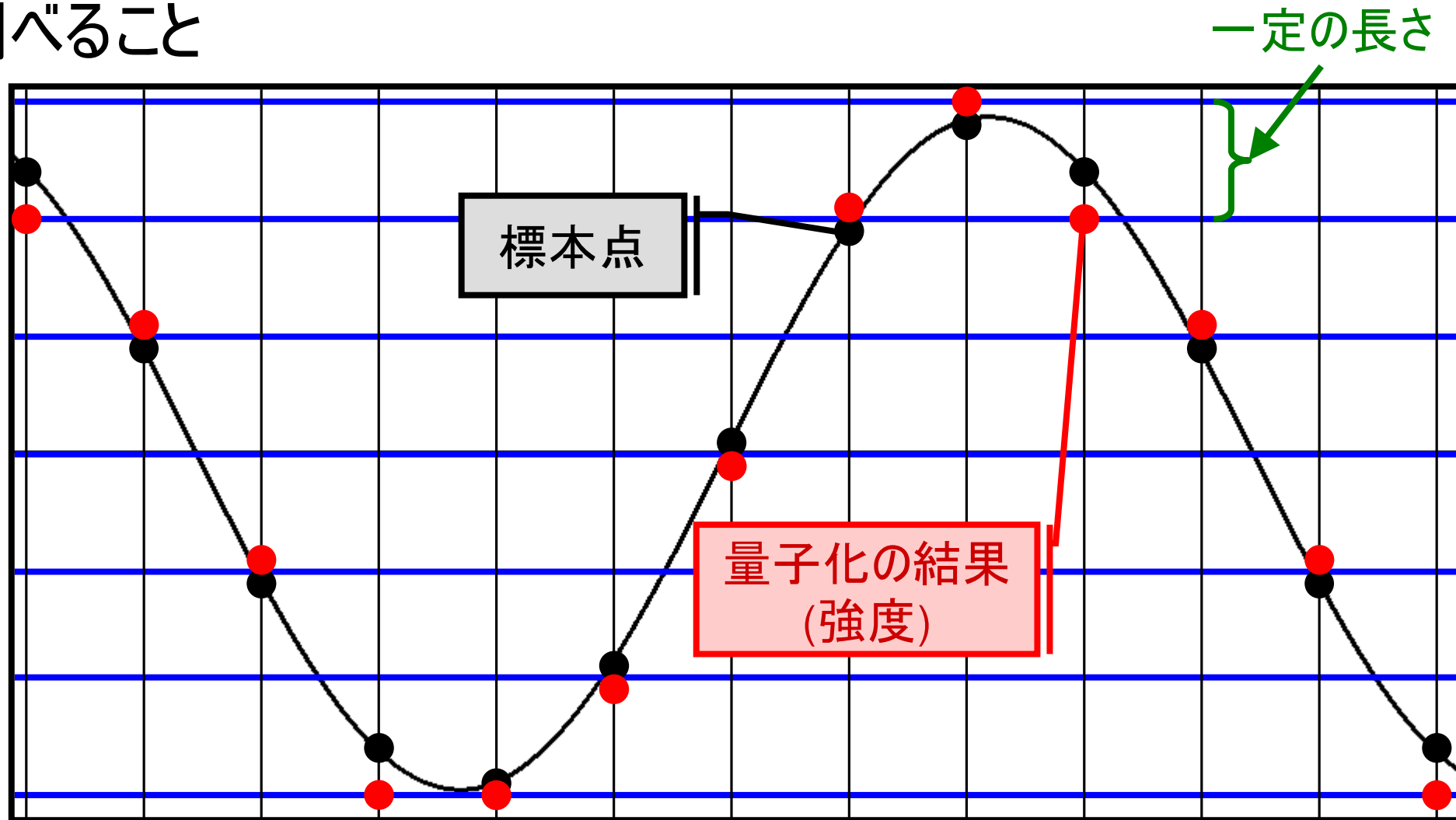
1. 横軸の値を調べるための縦軸の量(**量子化レベル**)をどの程度にするかを定める
2. 標本化で取り出された標本点のy軸の値(**強度**)を最も近い量子レベルに置き換える

アナログ情報のデジタル化が完了

- 最終的に、量子化の結果(**強度**)の値をコンピュータに取り込む
 - もとのアナログの値とは異なる値が取り込まれる

量子化[2](p. 22)

- 量子化: 縦軸を一定の長さで区切り、各標本点と最も近い縦軸の値を調べること



音の符号化[1](p. 23)

- 音: 空気などの分子の振動現象
- 人間の耳は、振動の中で20Hz～20kHzのものを聞き分ける
 - Hz(ヘルツ): 1秒間の振動の回数, または1秒間の標本化の回数
 - 正弦波の周期: 振動の間隔(1秒間の振動の回数)
 - 振動の回数が大きければ(多ければ)高い音
 - 振動の回数が小さければ(少なければ)低い音

音の符号化[2](p. 23)

- 20kHzの波形を再現するには、40kHzで標本化(1秒間に40000回標本化)
 - 20kHzの波: 正弦波の周期は $1/20000$
 - 標本化は、 $1/2$ 周期で1回行う
 - = 20kHzの波は $1/40000$ 間隔で標本化
 - = 40kHzで標本化
 - 音楽では高い音も再現する必要: 音楽CDは44.1kHzで標本化
 - 固定電話では人間の声の高さ程度を再現: 8kHzで標本化

音の符号化[3](p. 23)

■ 正弦波の縦軸: 音の強さ

- 固定電話: 8ビット(256種類)の量子化レベル
 - 0～255の256段階の数値で音の強さを表現
- 音楽CD: 16ビット(65,536種類)の量子化レベル
 - 0～65,535段階の数値で音の強さを表現(音の強さをより細かく表現可能)

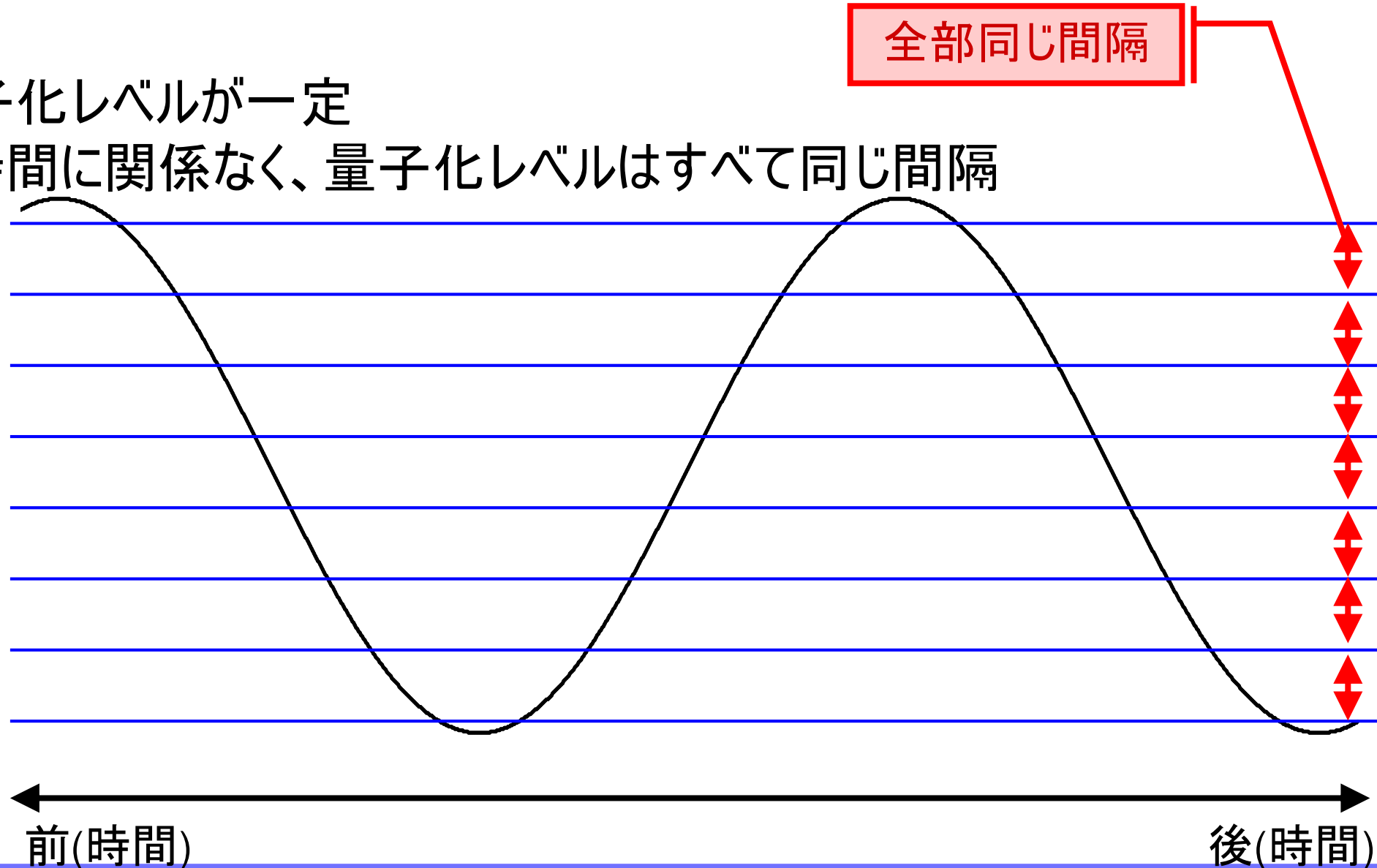
PCM(Pulse Code Modulation):

標本化したデータをそのまま量子化(量子化レベルが一定)

音の符号化[4](p. 23)

■ PCM

- 量子化レベルが一定
= 時間に関係なく、量子化レベルはすべて同じ間隔





アナログ情報とデジタル情報(続き)

画像の符号化[1](p. 23)

- 標本化した画像: 点の集まりと考えられる

- 点: 細かい正方形のマス目

画素(ピクセル, pixel)

- 画像の長方形のキャンバスは点の集まり
- 1つ1つの点の大きさによって画像の質が決定
 - 点が大きければ粗い画像
 - 点が小さければきめの細かい画像
- 1つ1つの点は何色かを記録しておくことで画像を表現

➡ 標本化の間隔により、点の大きさが決定

➡ 量子化の間隔により、画像中で利用可能な色の種類が決定
(どの程度、微妙な色合いを表現するか)

画像の符号化[2](p. 23)

- 白黒の2色の画像: 各画素を1ビットで表現
 - 点が白であれば「0」
 - 点が黒であれば「1」
- グレースケール(グレートーン, 白・黒・灰色の濃淡の画像): 1つの画素に0以上の整数を1つ割り当てて表現
 - 1つの画素を8ビットで表現: 0～255までの数を割り当て可能
 - 黒-灰色-白の濃淡を256段階で表現

画像の符号化[3](p. 23)

■ カラー画像

- コンピュータのディスプレイ: 赤(Red), 緑(Green), 青(Blue)の3つの光を利用
 - 3つの光にそれぞれ256段階の濃淡をつけ、3つの光を混ぜ合わせて色を作成
 - 256段階 = 8ビットで表現可能
 - 1つの色: $8\text{ビット} \times 3\text{つの光} = 24\text{ビットで表現}$
画像中の1つの点を24ビットで表現



カラー画像: 1つの画素を0～16,777,215までの数値で表現可能



データ圧縮と情報量

画像データの伝送[1](p. 25)

- 地上デジタル放送: 1440 × 1080の解像度でフレームレートが約30fps
 - fps(frame per second): 1秒あたりに切り替える画像の数

動画の世界では「フレーム」と呼ぶ

Ex. 1つの点を24ビット(3byte)で表現すると...

$$\begin{aligned} & (1440 \times 1080 \text{画素}) \times (3 \text{byte}) \times (30 \text{フレーム/秒}) \\ &= 139968000 \text{byte/秒} \\ &\doteq 140 \text{Mbyte/秒} \\ &(\text{ただし、音声なし}) \end{aligned}$$

画像データの伝送[2](p. 25)

- デジタル放送: 140Mバイト/秒の伝送が必要
 - テレビ局から家庭に映像を送るときに、1秒間に140Mバイト送れないと、なめらかな映像が見られない
 - Ex. 家庭のコンピュータのネットワーク環境
 - ADSL: 50Mビット/秒(bps) = 6.25Mバイト/秒
 - 光: 100Mビット/秒(bps) = 12.5Mバイト/秒

理論上の速さで、実際はもっと遅い



140Mバイト/秒は莫大な伝送量!!

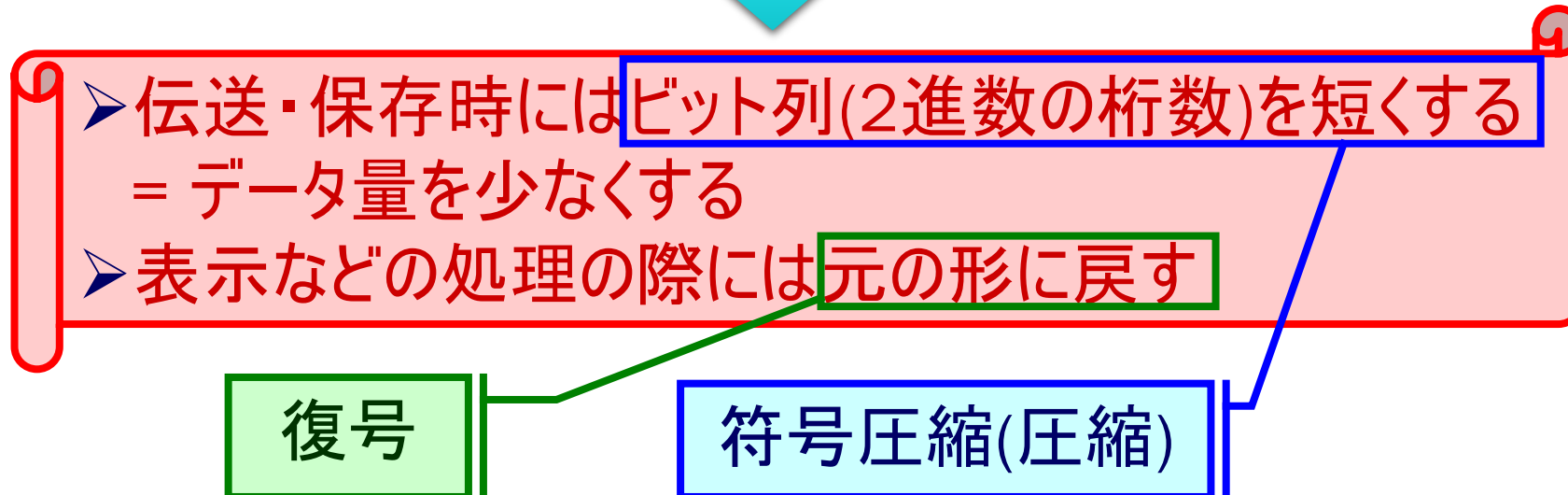
画像のデジタル表現[4](p. 25)

- 140Mバイト/秒は現実的に伝送不可能
→データを圧縮して伝送
 - **圧縮**: 決まった手順に従ってデータのサイズを小さくすること
- 静止画
 - 画像の中の特定の領域の色の変化は少ない
- 動画
 - 動画の中で実際に動く部分は大きくない
 - 動く部分を過去の様子から大体予測できる

デジタル放送では、この性質を使って圧縮
(圧縮技術: MPEG-2)

圧縮と復号(p. 26)

- 数値・文字・画像・音声: 符号化(2進数に変換)して処理
- 特に画像・音声はデータ量が多い
 - 伝送に時間が多く必要
 - 保存場所の容量が多く必要



可逆圧縮(p. 26)

■ 可逆圧縮

- 復号時に1ビットの違いもなく元のビット列が復元される圧縮法
 - 圧縮したファイルから、圧縮前のファイルを取り出すことができる
- 非可逆圧縮に比べて、ビット列をあまり短くすることはできない
 - 非可逆圧縮よりもデータ量の減量分は少ない

非可逆圧縮(p. 26)

■ 非可逆圧縮

- 復号時に元のビット列とは多少の違いが生じてしまう圧縮法
 - 圧縮したファイルから、圧縮前のファイルを取り出すことができない
- 可逆圧縮に比べて、ビット列がかなり短くなる
 - 可逆圧縮よりもデータ量の減量分が多い

数字・文字情報(p. 26)

- 復号したときに内容が変わってはいらない
 - 文書中の1文字が抜け落ちたら...?
 - 数値の小数点の桁数が少なくなったら...?
- 一般的に(画像・音声に比べて)データ量は少ない
 - 圧縮によるデータの減量分はそれほど(画像・音声ほど)多くなくて良い



可逆圧縮が使われる

画像・音声情報(p. 26)

- 復号したときに内容が多少変わっても良い
 - 画像: 表示したときに見た目が変わらなければ良い
 - 音声: 聞いたときに元と同じように聞こえれば良い
- 一般的に(数値・文字に比べて)データ量が多い
 - 圧縮によるデータの減量分が(数値・文字に比べて)多いことが必要



非可逆圧縮が使われる

- 動画・音声のデータ量は膨大なので非可逆圧縮
- 静止画のデータ量はそれほど多くないので可逆圧縮・非可逆圧縮のどちらも使われる

非可逆圧縮の圧縮法

- 余分な情報を取り除くことで圧縮

- 画像の場合、同じ色が集まっているところの色の情報など
- 音声の場合、人間の耳には聞こえないような音

人間の感覚ではわからないものを削るので、見た目・聞いた感じでは品質は変わらない

- 画像は拡大すると、画質が落ちているのがわかる
- 音声は、良いスピーカーを使うと音質が落ちているのがわかる
- 圧縮率を上げる(ファイルサイズをより小さくする代わりに、取り除くものを多くする)と、質が落ちているのがわかる

可逆圧縮の圧縮法(p. 28)

- ハフマン符号化
- ランレングス符号化
- etc.

ハフマン符号化[1](p. 28)

- **ハフマン符号化**: データ内に出現する情報を統計的に処理し、**ビット列の長さを変えて情報を表現**

これまで: どの情報も同じ長さのビット列で表現

- 出現率の高い情報を短いビット列で表現
- 出現率の低い情報を長いビット列で表現

ハフマン符号化[2](p. 28)

Ex. ある地方の320日の天気

天気	天気である日数	ビット列での表現
雨が降っていない	160日(2日に1日)	0
小雨が降っている	40日(8日に1日)	100
適度な雨が降っている	20日(16日に1日)	1010
やや強い雨が降っている	20日(16日に1日)	1011
非常に強い雨が降っている	20日(16日に1日)	1100
強い雨が降っている	20日(16日に1日)	1101
弱い雪が降っている	20日(16日に1日)	1110
大雪が降っている	20日(16日に1日)	1111



「10100100...」は、何日分のどんな天気？

ハフマン符号化[3](p. 28)

■ 「10100100...」

- 最初のビットが「1」なので、「雨が降っていない」という天気ではない
- 2ビット目が「0」なので、「非常に強い雨」、「強い雨」、「弱い雪」、「大雪」という天気ではない
- 3ビット目が「1」なので、「小雨」という天気ではない
- 4ビット目が「0」なので、「やや強い雨」という天気ではない

1日目の天気は「適度な雨が降っている」



それぞれの情報を同じ長さのビット列にしなくても情報の表現は可能

※情報の種類ごとに割り当てるビット数を変える方式: 可変長符号

ハフマン符号化[4](p. 29)

- 8種類の天気を普通のやり方で現すと...?
 - 1種類を3ビットで表現: 320日では、 $320 \times 3 = 960$ ビット

ハフマン符号化では...

雨が降っていない	160日	0	1ビット × 160日
小雨が降っている	40日	100	3ビット × 40日
適度な雨が降っている	20日	1010	4ビット × 20日
やや強い雨が降っている	20日	1011	4ビット × 20日
非常に強い雨が降っている	20日	1100	4ビット × 20日
強い雨が降っている	20日	1101	4ビット × 20日
弱い雪が降っている	20日	1110	4ビット × 20日
大雪が降っている	20日	1111	4ビット × 20日

合計: 760ビット

➡ 200ビット少なくなっている

※どれだけ少なくなるかは扱う情報によって違う

ランレングス符号化[1](p. 29)

- **ランレングス符号化**: 白黒2値画像(灰色のない、白と黒のみの画像)の圧縮方法の1つ
 - 主にFAXで使われている

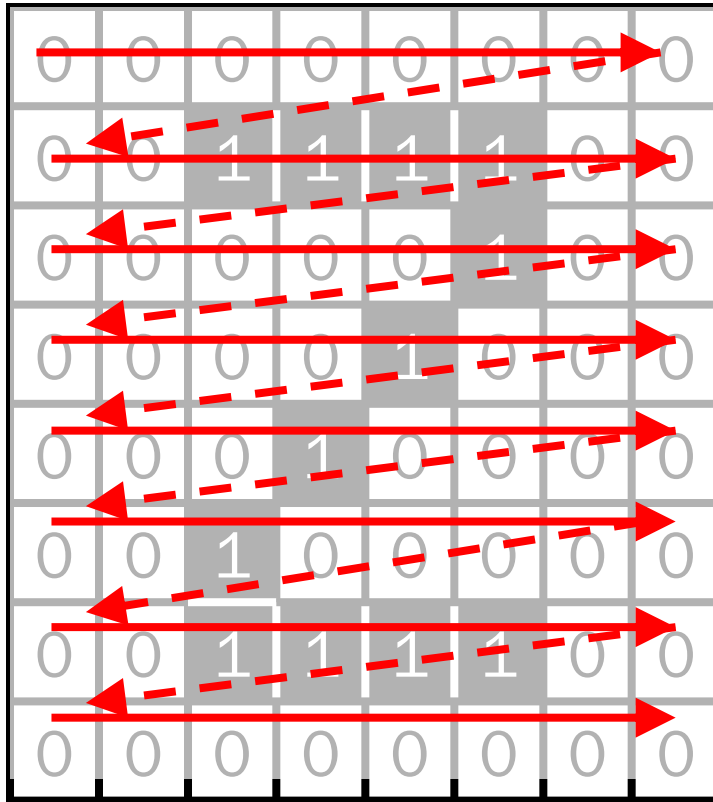
0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0
0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	1	1	0	0
0	0	0	0	0	0	0	0

8 × 8 = 64ビット

(2値画像は1つ1つの点を1ビットで表現するため)

ランレングス符号化[2](p. 29)

- 2値画像: 白画素と黒画素はある程度固まっている



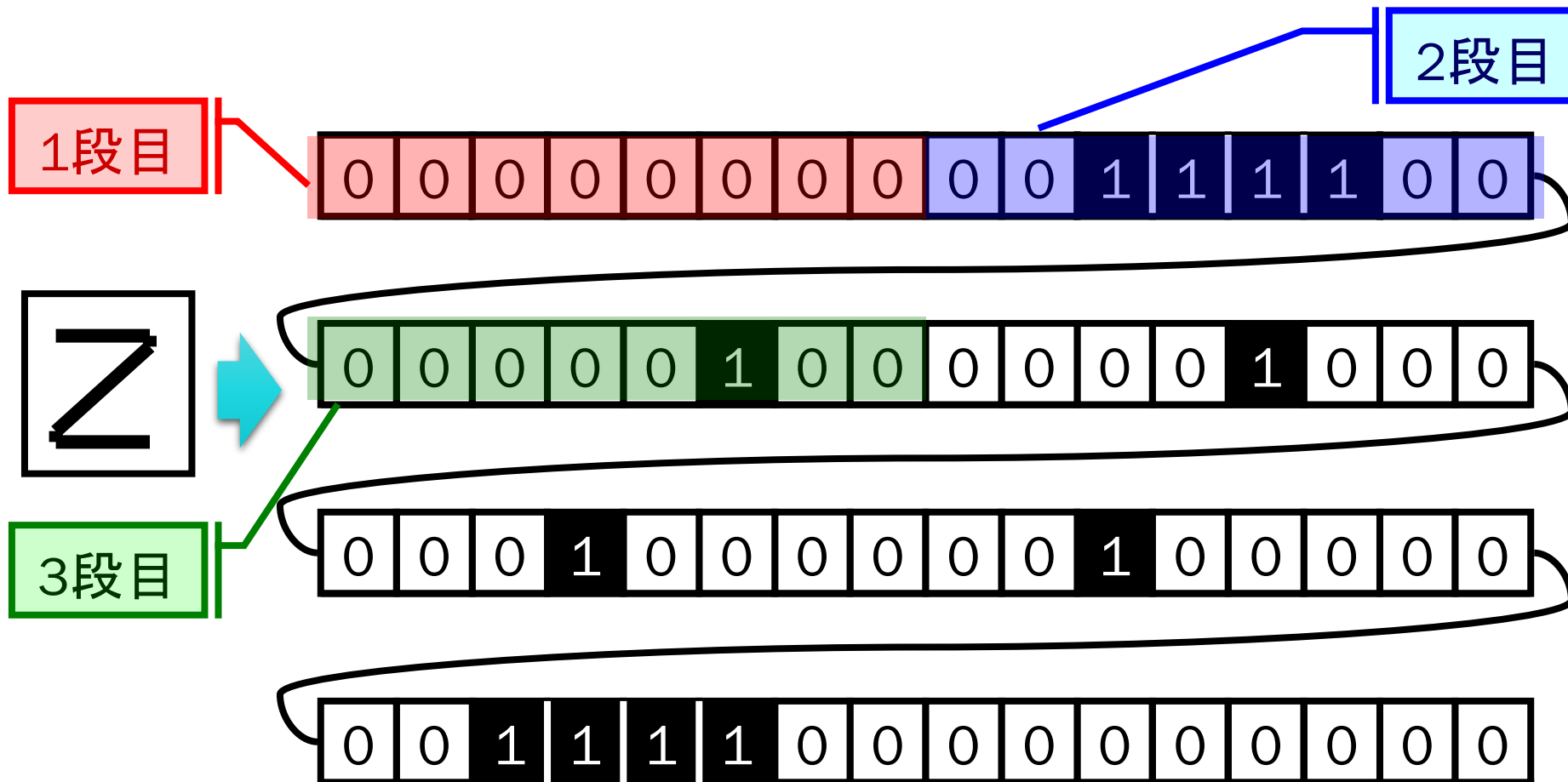
白画素が10個
黒画素が4個
白画素が7個

.....
黒画素が4個
白画素が10個

矢印の方向に画素を見ていったとき

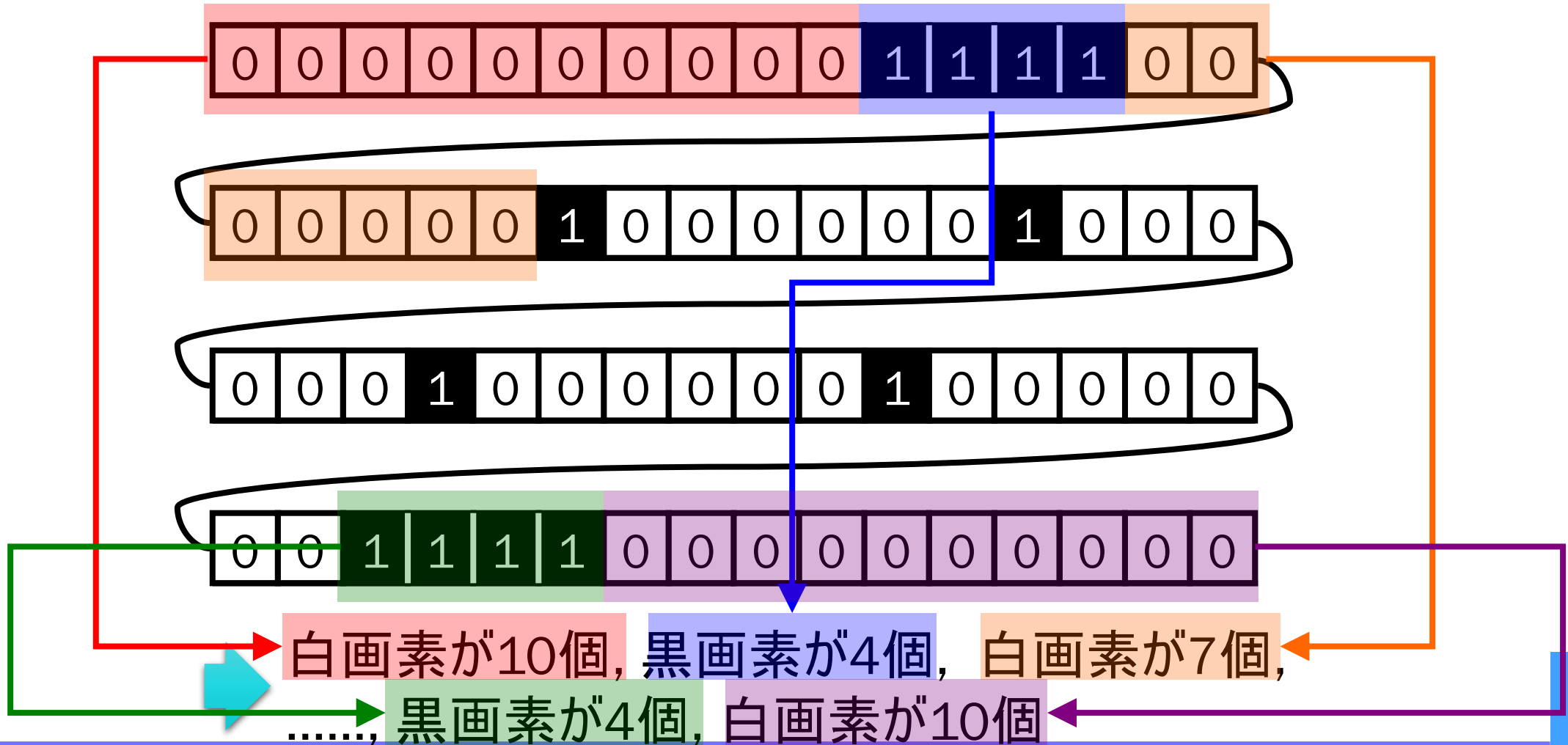
ランレングス符号化[圧縮][1]

- 画像の中の全ての画素を横一列に並べる
 - 横1段目の右に2段目、2段目の右に3段目...とつなげていく



ランレングス符号化[圧縮][2]

- 横一列に並べた画素について、左から順に、連続している白画素と黒画素の数を数えていく



ランレングス符号化[圧縮][3]

白画素が10個 黒画素が4個 白画素が7個
黒画素が4個 白画素が10個



2進数で表すと...

1010	0100	0111	0001	0110	0001	0110
0001	0110	0001	0111	0100	1010	

52ビット

※個数の中で「1010」(2進数)が最も大きな個数なので、他の個数も、
2進数で表現したときの桁数を「1010」(4桁)にあわせる

通常の方法で表す(1画素を1ビットで表す)と...

8 × 8 = 64ビット

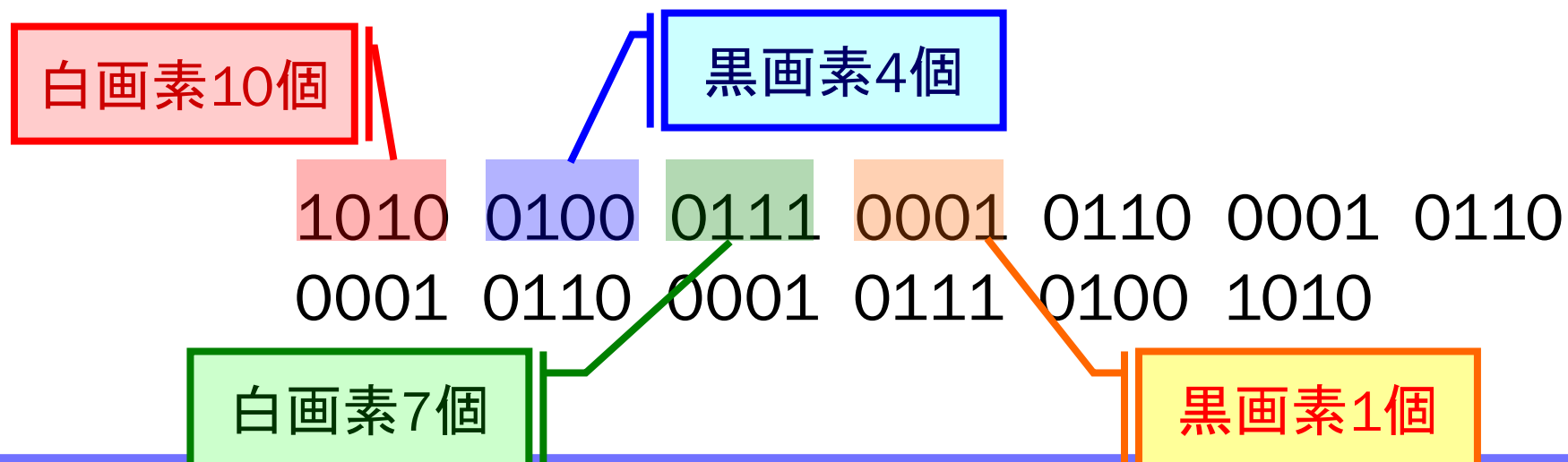


12ビット少なくなっている

※どれだけ少なくなるかは扱う画像の内容によって違う

ランレングス符号化[復号][1]

- 圧縮したものを復号化するには...?
 - 画像の画素の数だけを表したものの(色の情報はない)
 - 白画素の並びと黒画素の並びは必ず交互になる
 - 画像の最も左上隅の画素は白であることが多い
 - 先頭の個数は、白画素の個数として扱う
 - 最も左上隅の画素が黒の場合、圧縮時に、最初に出現する白画素の個数を0個としておく

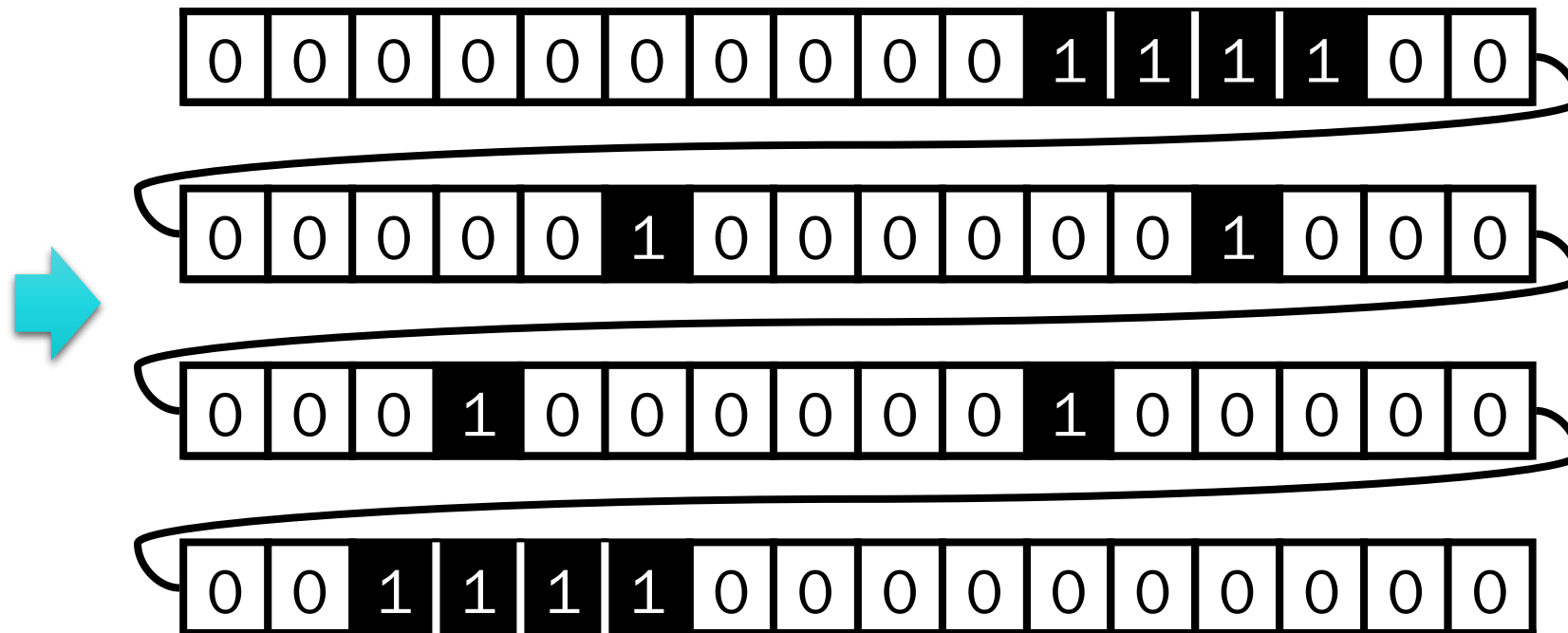


ランレングス符号化[復号][2]

- 画素の個数から、横一列の画素の並びが復元

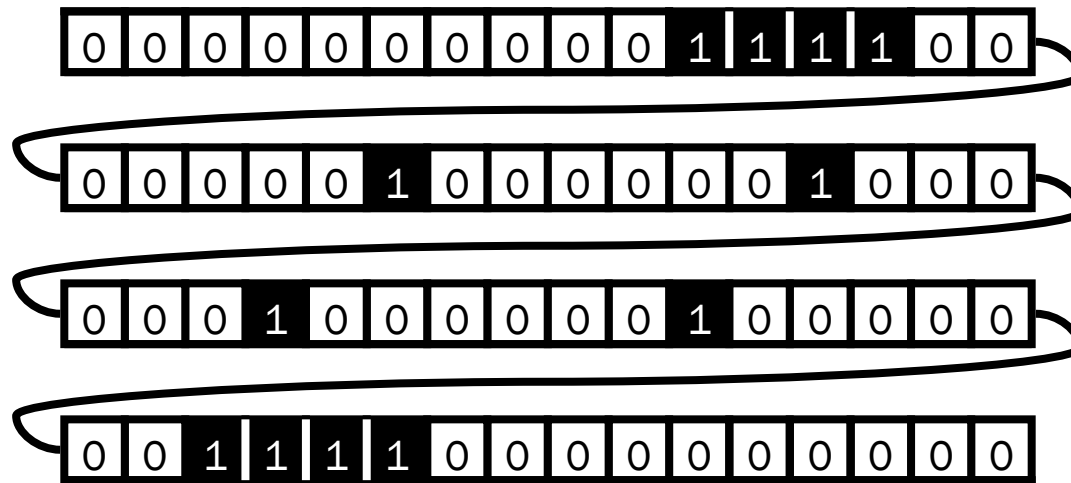
1010 0100 0111 0001 0110 0001 0110
0001 0110 0001 0111 0100 1010

➡ 白画素が10個, 黒画素が4個, 白画素が7個,
....., 黒画素が4個, 白画素が10個

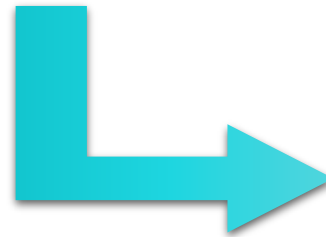


ランレングス符号化[復号][3]

- もとの画像の縦横の画素の数は記録されてある
 - 横一列の画素の並びが復元できると、もとの画像も復元できる



もとの画像は横8個、縦8個の画素



0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0
0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	1	1	0	0
0	0	0	0	0	0	0	0

先頭が黒画素のとき[1]

- 前のスライドの画像の、白と黒を逆にした画像を考えると...

1	1	1	1	1	1	1	1
1	1	0	0	0	0	1	1
1	1	1	1	1	0	1	1
1	1	1	1	0	1	1	1
1	1	1	0	1	1	1	1
1	1	0	1	1	1	1	1
1	1	0	0	0	0	1	1
1	1	1	1	1	1	1	1

白画素・黒画素の個数を数えると...(前のスライドの画像の白と黒を逆にした画像なので...)

黒画素10個

白画素4個

1010 0100 0111 0001 0110 0001 0110
0001 0110 0001 0111 0100 1010

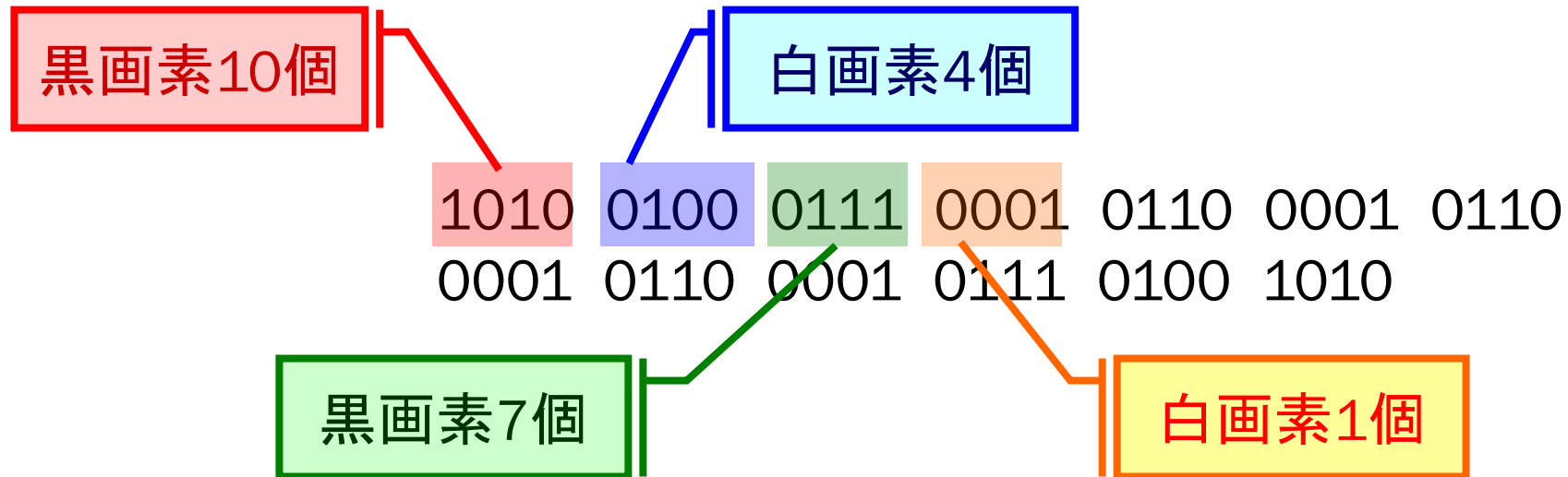
黒画素7個

白画素1個

先頭が黒画素のとき[2]

■ ランレングス符号化の考え方

- 白画素・黒画素の個数だけを並べたとき、先頭の個数は白画素の個数とみなす



先頭の個数は黒画素の個数! ...でもこれでは困る
→先頭の個数を「白画素0個」にすると良い

→ 0000 0100 0111 0001 0110 0001 0110
0001 0110 0001 0111 0100 1010

次回

- 実習をするので24102教室で授業
 - 写真をスキャナで取り込んで標本化・量子化のレベルの違いの確認を実習
- 紙媒体の写真を持ってくること
 - 電子データは不可
 - 電子データしかなければ印刷するなどして紙媒体で持ってくること