コンピュータ・サイエンス1

第14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14回の内容

- ☑ データ圧縮

設問1

- ☑ 下記の中で正しい説明をすべて選びなさい。
  1. ある写真を300dpiと50dpi、どちらも24bitカラーで取り込んだとき、300dpiで取り込んだ方がファイルサイズが大きい
     - ☑ 300dpiの方がよりきめ細かく取り込むのでファイルサイズが大きい
  2. ある写真を72dpiと48dpi、どちらも8bitグレーで取り込んだとき、48dpiで取り込んだ方がファイルサイズが大きい
     - ☑ 48dpiの方が粗く取り込むのでファイルサイズは小さい
  3. ある写真を8bitカラーと24bitカラー、どちらも300dpiで取り込んだとき、8bitカラーで取り込んだ方がファイルサイズが大きい
     - ☑ 8bitカラーの方が1つの点を表現する色のビット数が小さいのでファイルサイズは小さい
  4. ある写真を8bitグレーと24bitカラー、どちらも50dpiで取り込んだとき、24bitカラーで取り込んだ方がファイルサイズが大きい
     - ☑ 24bitカラーの方が1つの点を表現する色のビット数が大きいのでファイルサイズが大きい

解答: 1, 4

設問2

- ☑ 下記は、20日分の朝食のメニューを表している。ハフマン符号化で暗号化するとき、割り当てるビット列が少ないメニュー順に並べなさい。

1日	トースト
2日	おにぎり
3日	おにぎり
4日	なし
5日	卵焼き
6日	卵焼き
7日	トースト
8日	トースト
9日	トースト
10日	おにぎり

11日	コーンフレーク
12日	おにぎり
13日	トースト
14日	卵焼き
15日	コーンフレーク
16日	コーンフレーク
17日	卵焼き
18日	トースト
19日	おにぎり
20日	トースト

ハフマン符号化: 出現率の大きい情報ほど少ないビット列で情報を表現

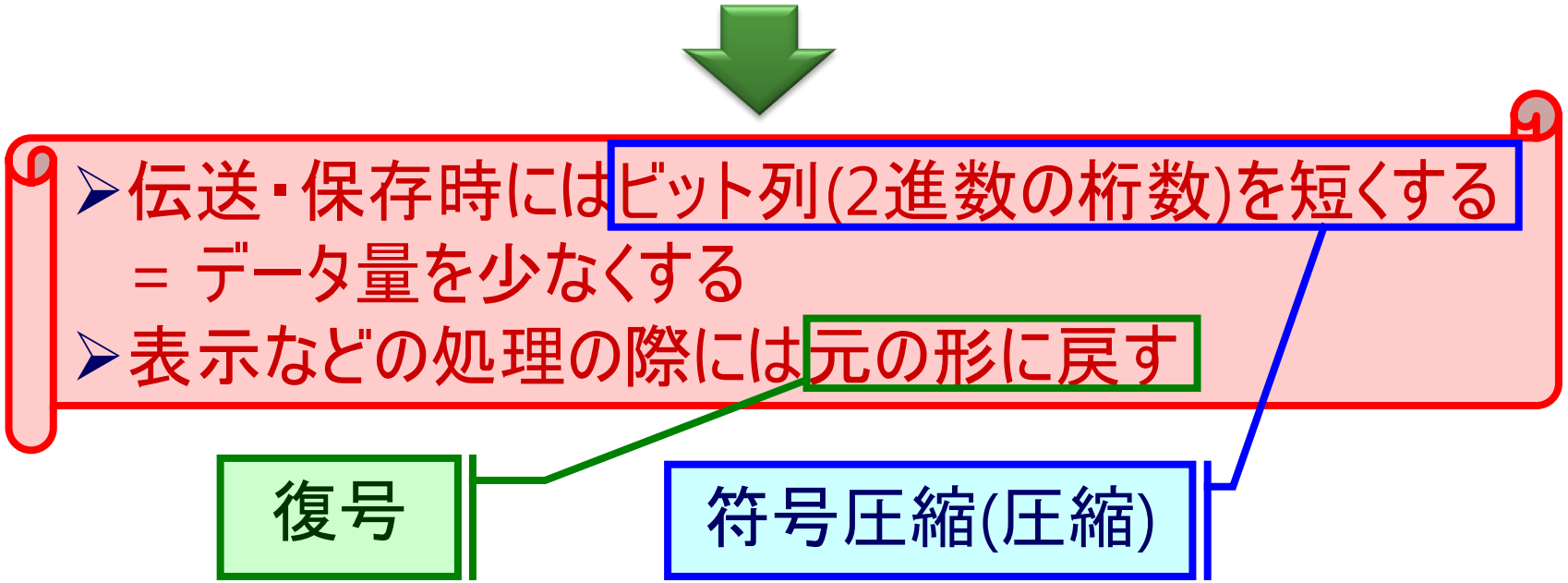
メニュー	出現回数
トースト	7
おにぎり	5
卵焼き	4
コーンフレーク	3
なし	1

解答: トースト, おにぎり, 卵焼き, コーンフレーク, なし

データ圧縮と情報量

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ならない
  - ☑ 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、ほとんどの場合非可逆圧縮が使われる
- ➢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(=たくさん出てくる)情報を短いビット列で表現
  - 出現率の低い(=あまり出てこない)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

一番たくさん出てくる情報なので
一番短いビット列で表現

2番目にたくさん出てくる情報なので
2番目に短いビット列で表現

一番少なく出てくる情報なので
一番長いビット列で表現

※各情報をどのような0と1で表すかはややこしいので割愛

ハフマン符号化[4](p. 29)

- 8種類の天気を普通のやり方(圧縮なし)で現すと…?
  - 1種類を3ビットで表現: 320日では、320×3=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)

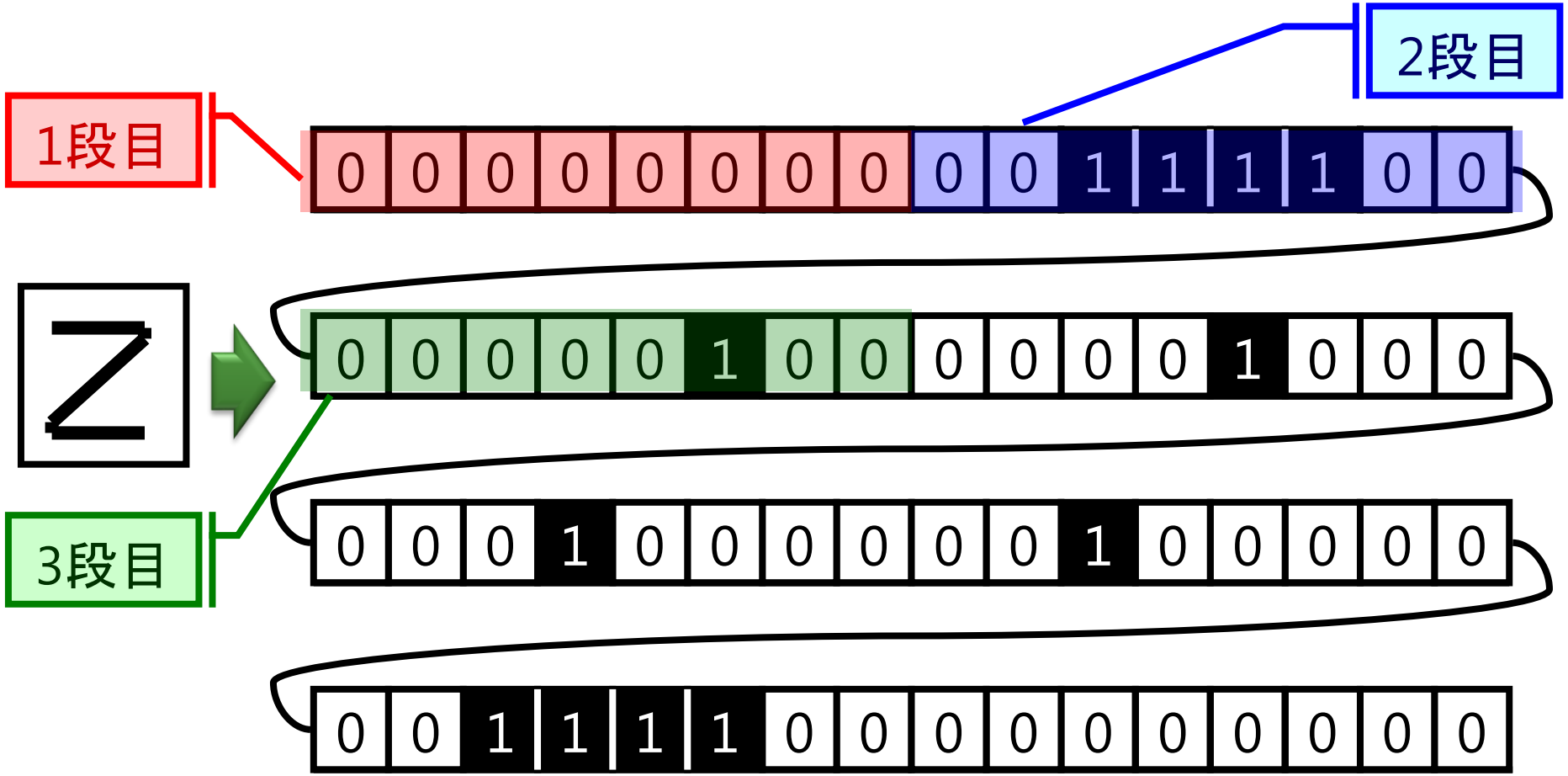
- ランレングス符号化: 白黒2値画像(灰色のない、白と黒のみの画像)の圧縮方法の1つ
  - 主にFAXで使われている

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)

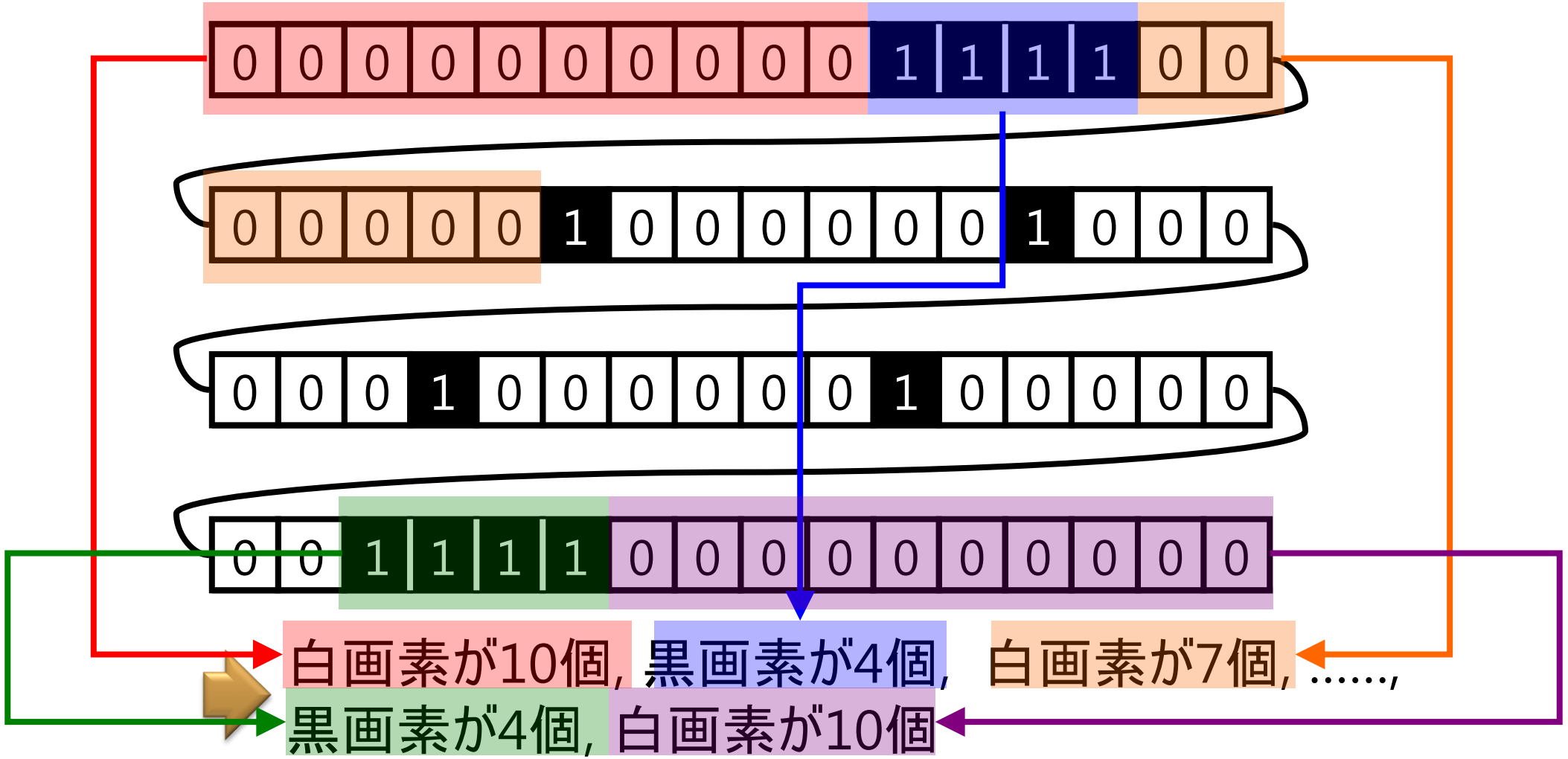
ランレングス符号化[圧縮][1]

- 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

画素の個数を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

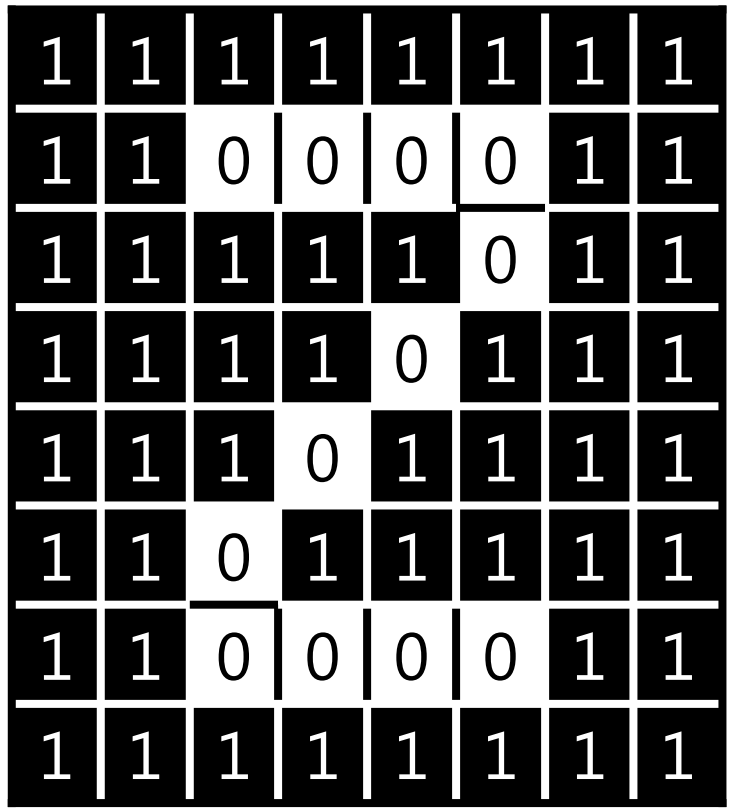
通常の方法で表す(1画素を1ビットで表す)と…
8×8=64ビット

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

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…



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

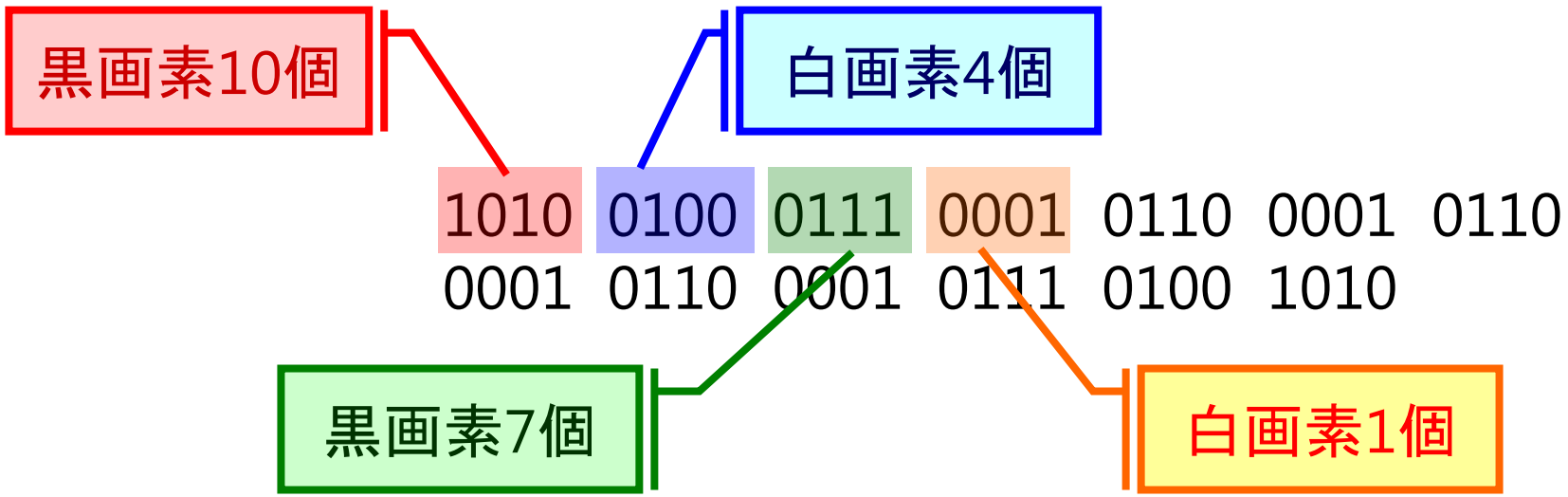
1010 0100 0111 0001 0110 0001 0110
0001 0110 0001 0111 0100 1010

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



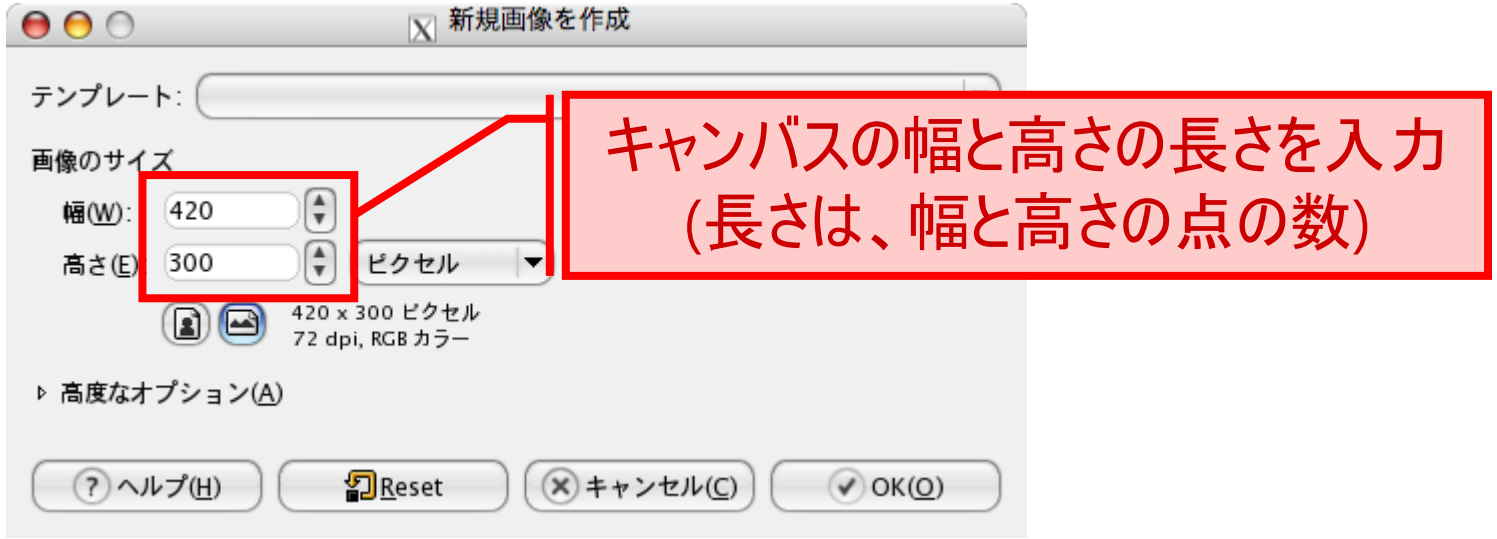
先頭の個数は黒画素の個数! …でもこれでは困る
→先頭の個数を「白画素0個」にすると良い

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

圧縮の実習～準備: GIMP～

GIMPの基本～起動～

- ☑ GIMP: 描画ソフトウェア
- ☑ Finder→「アプリケーション」→「GIMP」をダブルクリック
- ☑ 「ファイル」→「新しい画像」をクリック
- ☑ 絵を描くキャンバスの大きさを決め、「OK」をクリック



GIMPの基本～描画～

- ツールボックスの中のボタンをクリックし、キャンバスに絵を描く
  - 鉛筆やブラシ、インクツールであれば、マウスをドラッグ＆ドロップ
    - ツールを選択すると、ツールボックスの下に線の太さなどの設定ができるウィンドウが表示
  - 文字であれば、キャンバス上でクリックし、文字を入力

鉛筆・ブラシ・文字のツール

クリックして線や文字の色を選択

GIMPの基本～描いたものの保存～

- キャンバスのメニュー→「ファイル」→「エクスポート」をクリック
  - 「ファイル」→「保存」ではないことに注意!
    - GIMPならではの形式で保存されてしまって、他のソフトウェアで開けなくなる
- 「名前」欄にファイル名を入力
  - ファイル名は、拡張子をつけて入力すること(どの拡張子で保存するかは後で説明)
    - GIMPは、ファイルの拡張子に応じてファイルの保存形式を決定
    - ファイル名を「image」としたいと思い、拡張子を「tiff」と指定された場合:「image.tiff」と入力する
- 「他のフォルダを参照」を押すと、保存するフォルダを選択可能

GIMPの基本～保存してある画像を開く～

- ☑ Finder→「アプリケーション」→「GIMP」をダブルクリック
- ☑ 「ファイル」→「開く/インポート」をクリック
- ☑ 表示されたウィンドウで、開きたいファイルを選択し、「開く」をクリック

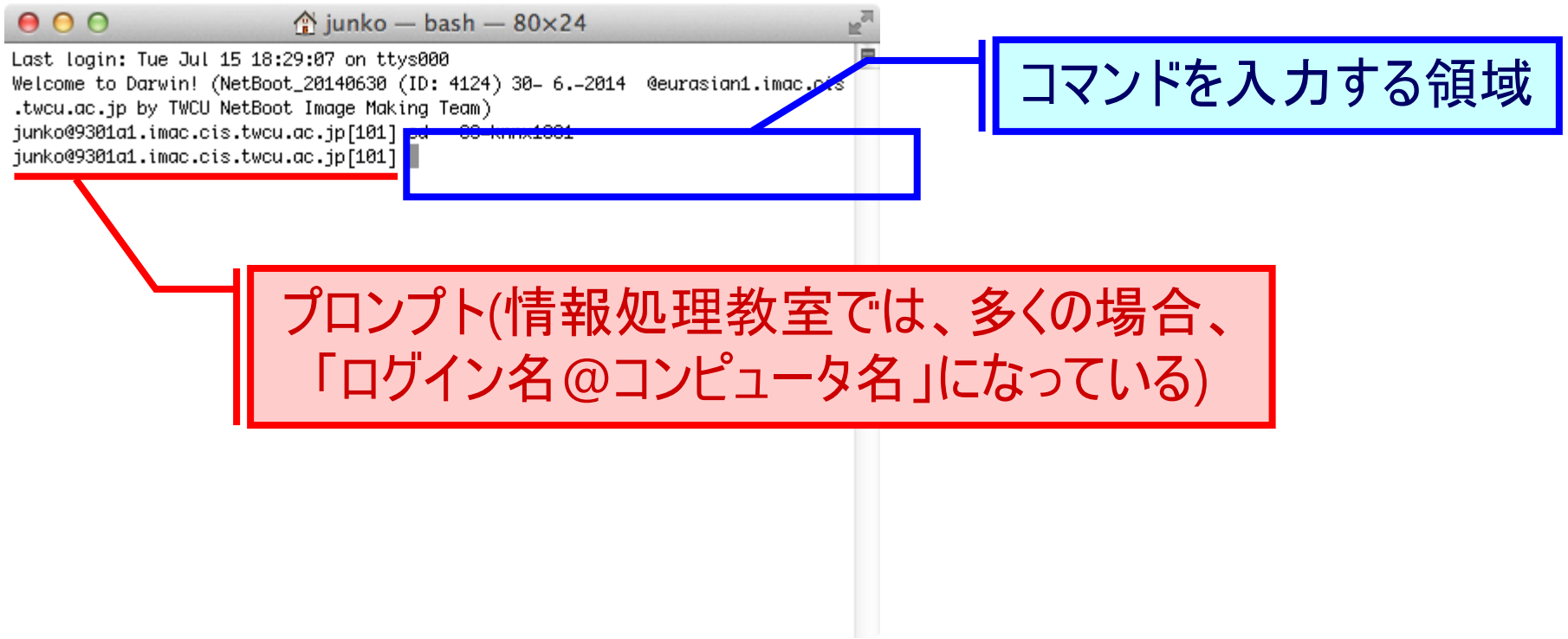
圧縮の実習～準備: ファイルサイズの確認方法～

ファイルサイズの確認方法

- Mac OS Xではリソースフォークがあるため、Finderからはファイルそのもののサイズを確認できない
  - リソースフォーク: アイコンの形や開くアプリケーションなど、ファイルに関する様々な情報を保存したファイル
    - Finderでは表示されないファイル
  - Finderで表示されるファイルサイズは、ファイルそのものサイズとリソースフォークのサイズをあわせたもの
    - Finderで表示されるファイルサイズは、圧縮の実習でのファイルサイズの比較には向かない

ファイルそのもののサイズを確認するには?(1)

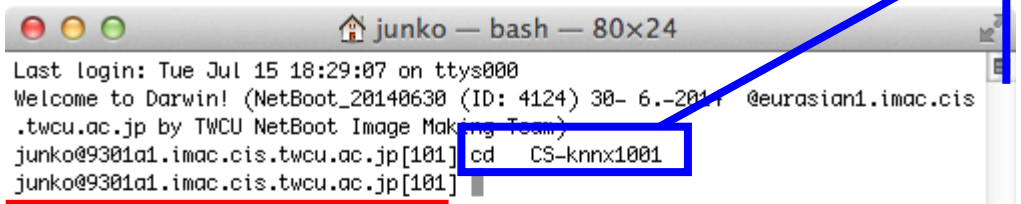
- Finder→「ユーティリティ」→「ターミナル」を起動
  - 起動したウィンドウに、文字を入力して様々な処理をするためのソフトウェア
    - 入力する文字(命令)を「コマンド」と呼ぶ
    - コマンドは「プロンプト」の後に入力し、「Return」キーを押すことで実行される



ファイルそのもののサイズを確認するには?(2)

- ターミナルでの作業場所を、ファイルを保存しているフォルダにあわせる
  - ターミナルでの作業場所は、前回ターミナルを使い終わった場所に設定されている
    - 初めて使った場合はホームフォルダ
  - ターミナルを起動したときに1度だけ行えば良い
- 作業場所を設定するには…
  cd　フォルダ名
  と、コマンド入力領域に入力し、「Return」キーを押す
  - フォルダ名は、「書類」は「Documents」、デスクトップは「Desktop」と入力
  - 「No such file or directory」など、何かのメッセージが表示されたら、やり直し
    - 以前に入力された内容は消せないので、そのまま追加で「cd　フォルダ名」を入力
  - 単に「cd」と入力して「Return」キーを押すと作業場所はホームフォルダに設定
    - どうしてもうまくいかない場合は、一度ホームフォルダに設定して再度やりなおし

ファイルそのもののサイズを確認するには?(3)



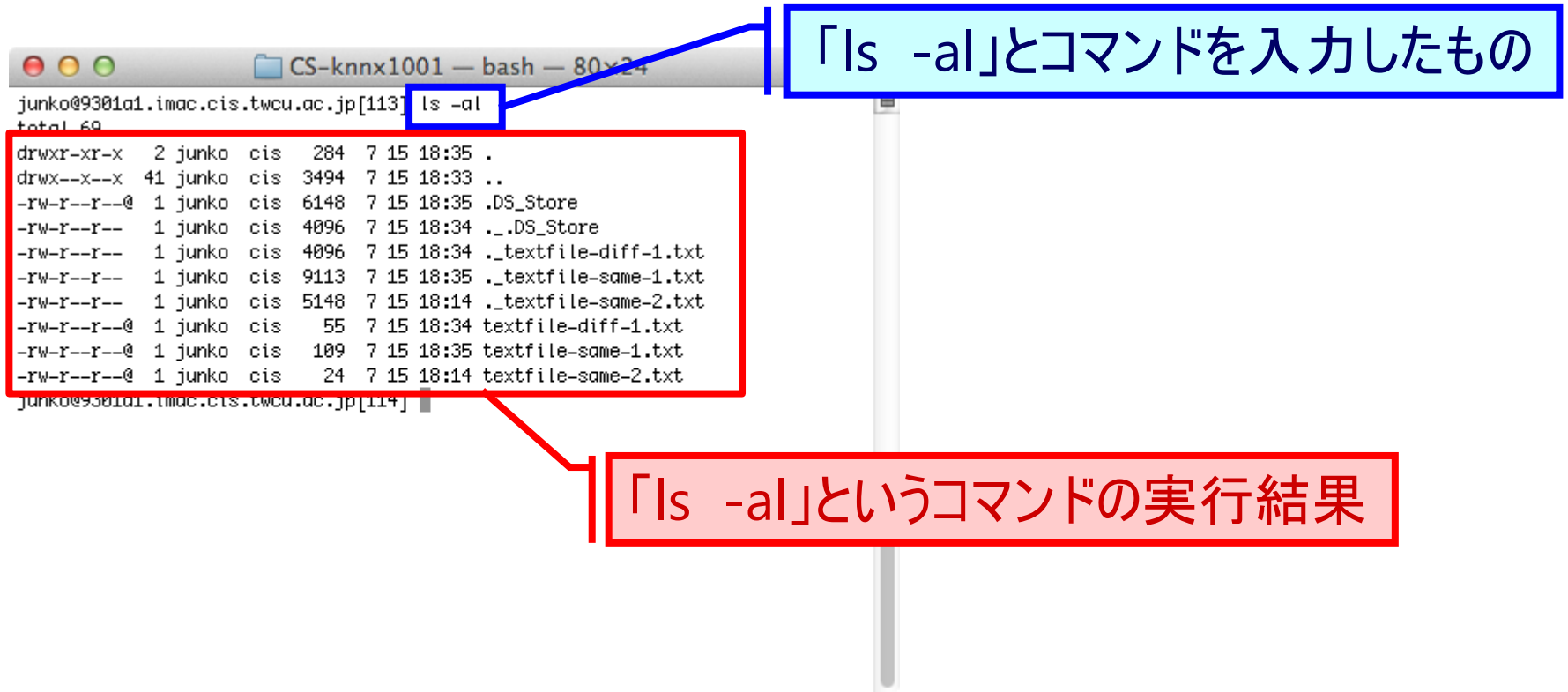
「cd　CS-knnx1001」と入力し、作業場所を「CS-knnx1001」フォルダにあわせている

コマンドが成功したら、何もメッセージはなく次のプロンプトが表示される

- コマンドの実行に失敗したら、何かメッセージが表示されて次のプロンプトが表示される
- 失敗した場合は、前の内容は消せないので、新しくコマンドを書き直してやり直す

ファイルそのもののサイズを確認するには?(4)

- ファイルサイズを確認するには…
  ls　-al
  とコマンド入力領域に入力し、「Return」キーを押す
  - 作業場所に設定したフォルダに保存されているすべてのファイルの情報が表示される



ファイルそのもののサイズを確認するには?(5)

- 「ls -al」コマンドで表示されたファイルサイズを確認
  - 「.」で始まるファイル名はリソースフォークなので関係なし

```
drwxr-xr-x   2 junko  cis   284  7 15 18:35 .
drwx--x--x  41 junko  cis  3494  7 15 18:33 ..
-rw-r--r--@  1 junko  cis  6148  7 15 18:35 .DS_Store
-rw-r--r--   1 junko  cis  4096  7 15 18:34 ._.DS_Store
-rw-r--r--   1 junko  cis  4096  7 15 18:34 ._textfile-diff-1.txt
-rw-r--r--   1 junko  cis  9113  7 15 18:35 ._textfile-same-1.txt
-rw-r--r--   1 junko  cis  5148  7 15 18:14 ._textfile-same-2.txt
-rw-r--r--@  1 junko  cis    55  7 15 18:34 textfile-diff-1.txt
-rw-r--r--@  1 junko  cis   109  7 15 18:35 textfile-same-1.txt
-rw-r--r--@  1 junko  cis    24  7 15 18:14 textfile-same-2.txt
```

ファイル名

ファイルサイズ

圧縮の実習～実習～

作成する画像(1)

- 画像1: キャンバス内に何も絵を描いていない画像(拡張子: tiff)
  - キャンバスの色は何色でもOK
- 画像2: 画像1と同じキャンバスサイズで、何か絵を描いている画像(拡張子: tiff)
  - 絵の内容は何でもOK
- 画像3: 画像1のファイル形式を「PNG」にしたもの(拡張子: png)
- 画像4: 画像2のファイルを「PNG」にしたもの(拡張子: png)

保存時に表示されるウィンドウは「OK」で進む

作成する画像(2)

- 画像5: 画像2のファイル形式を「JPG」にしたもの(拡張子: jpg)
  - 保存時に表示されるウィンドウの「品質」の数を「100」にして保存
  - キャンバスを閉じ、画像2を開きなおして画像5を作ること
- 画像6:画像2のファイル形式を「JPG」にしたもの(拡張子: jpg)
  - 保存時に表示されるウィンドウの「品質」の数を「0」にして保存
  - キャンバスを閉じ、画像2を開きなおして画像6を作ること

画像1～6が区別できるようにファイル名をつけておくこと

ファイルサイズの表示

- ターミナルで、「ls　-al」コマンドで表示されたファイルサイズを確認
  - 「.」で始まるファイル名はリソースフォークなので関係なし

ファイル形式

- ☑ TIFFの画像(拡張子が「.tiff」の画像): 圧縮なしの画像の形式
- ☑ PNGの画像(拡張子が「.png」の画像): 可逆圧縮の画像
- ☑ JPEGの画像(拡張子が「.jpg」の画像): 非可逆圧縮の画像

比較(1)

- 画像1と画像2
  - 画像1: 何も描いていない画像
  - 画像2: 画像1と同じ幅と高さの画像で、何か絵を描いている画像
  - 画像1と画像2のファイルサイズを比較

    - どちらが大きいか/小さいか?
    - その理由は何か?

比較(2)

- 画像2と画像4
  - 画像2: 何か絵を描いている、「TIFF」という形式の画像
  - 画像4: 画像2を「PNG」という形式で保存した画像
  - 画質の比較
    - 画像2と画像4をそれぞれダブルクリックで開く(「プレビュー」というソフトウェアで開かれる)
    - プレビューの「＋」(拡大)ボタンを何度か押して拡大して画質を見比べる
      (同じ大きさに拡大して比べること)
  - ファイルサイズの比較
    - 画像2と画像4のファイルサイズを比較
      - どちらが画質が良いか/悪いか?
      - どちらのファイルサイズが大きいか/小さいか?
      - その理由は何か?

比較(3)

- 画像3と画像4
  - 画像3: 何も描いていない画像を「PNG」の形式で保存したもの
  - 画像4: 絵を描いている画像で「PNG」の形式で保存したもの
    - 画像3と同じ幅と高さの画像
  - ファイルサイズの比較
    - 画像3と画像4のファイルサイズを比較

- どちらが大きいか/小さいか?
- 同じキャンバスサイズなのに、サイズの違いが出る理由は何か?

比較(4)

- 画像4と画像5
  - 画像4: 画像2を「PNG」という形式で保存したもの
  - 画像5: 画像2を「JPEG」という形式で、品質を「100」にして保存したもの
  - 画質の比較
    - 画像4と画像5をそれぞれダブルクリックで開く(「プレビュー」というソフトウェアで開かれる)
    - プレビューの「＋」(拡大)ボタンを何度か押して拡大して画質を見比べる
      (同じ大きさに拡大して比べること)

- どちらが画質が良いか/悪いか?
- その理由は何か?

比較(5)

- 画像5と画像6
  - 画像5: 画像2を「JPEG」という形式で、品質を「100」にして保存したもの
  - 画像6: 画像2を「JPEG」という形式で、品質を「0」にして保存したもの
  - 画質の比較
    - 画像5と画像6をそれぞれダブルクリックで開く(「プレビュー」というソフトウェアで開かれる)
  - ファイルサイズの比較
    - 画像5と画像6のファイルサイズを比較

      - どちらが画質が良いか/悪いか?
      - どちらのファイルサイズが大きいか/小さいか?
      - その理由は何か?

Question!