# データ量を減らすしくみと
# データの誤り検出・訂正

情報Ⅰ　第13回授業

03情報のデジタル化

対応ファイル：23exp13.xls

# データの圧縮

データを内容や意味を保ったまま別のデータに変換し、データ量を減らす処理を「圧縮」という。

- 可逆圧縮
  - 完全に元の情報を復元する圧縮方法
    - GIF、PNGなどの圧縮、ファイルの圧縮
- 非可逆圧縮
  - 完全には元の情報に戻らない圧縮方法
    - JPEG、MP3などの圧縮

圧縮と逆の処理を「展開（、伸長、解凍）」という。

※一般に、非可逆圧縮の方が可逆圧縮よりも圧縮率が高い

# 圧縮率の計算

圧縮率（%）：圧縮前のデータ量を基準とした圧縮後の割合（%）

→　圧縮「後」のデータ量　／　圧縮「前」のデータ量　×　100

一般的に、「基準となるもの（この場合は『圧縮前』）で割る！」

例）　5MB のデータを圧縮したら、256KBになった。　圧縮率は？

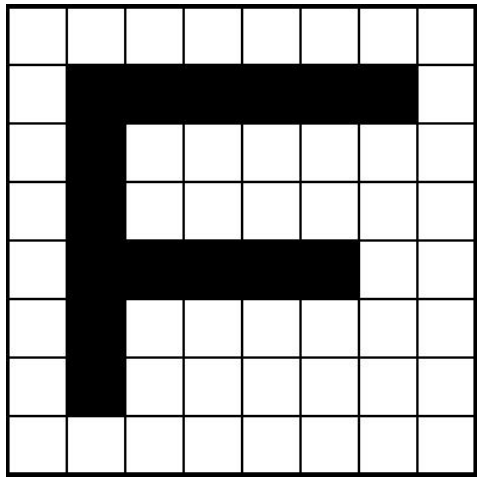
→　まずは単位をそろえる！

5MB　→　5120KB　だから　（← 1MB ＝ 1024KB ）

256　／　5120　×　100　＝　5　（%）

# 可逆圧縮のしくみ（1）

- ランレングス圧縮
  - 同じデータの繰り返しパターンに注目した方式



8×8＝64bit

| 0が9個 | 1が6個 | 0が2個 | 1が1個 | 0が7個 | 1が1個 | 0が7個 | 1が5個 | 0が3個 | 1が1個 | 0が7個 | 1が1個 | 0が14個 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|

4bitで表現可

| 9 | 6 | 2 | 1 | 7 | 1 | 7 | 5 | 3 | 1 | 7 | 1 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|

「0」と「1」の列で表すことを「符号化」という。

1001 0110 0010 0001 0111 0001 0111 0101 0011 0001 0111 0001 1110

12bit減少

52÷64×100
≒ 81%に圧縮

4×13＝52bit

※圧縮するものによって圧縮率は変わり、場合によっては、むしろ増えてしまうこともある！

# 可逆圧縮のしくみ（2）

例） AABAAACAABADABABBBAAC

→ A,B,C,D を、2ビットの二進法の数に符号化し置き換えてみると・・・

| 文字 | 符号 |
|---|---|
| A | 00 |
| B | 01 |
| C | 10 |
| D | 11 |

→ 00 00 01 00 00 00 10 00 00 01 00 11 00 01 00 01 01 00 00 10
→ 40 bit の情報量

→ もう少し効率の良い方法があるのでは？

データの桁数を固定しておく
→「固定長」とも言う

# 可逆圧縮のしくみ（2）

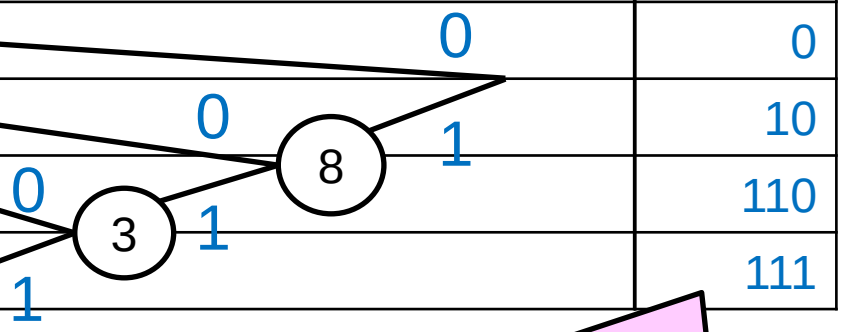
- ハフマン符号化
  - 出現頻度が高いデータを短いビット列に割り当てる。

例） AABAAACAABADABABBAAC

→ 0 0 10 0 0 0 110 0 0 10 111 0 10 0 10 10 0 0 110

→ 30 bit の情報量

| 文字 | 頻度 | | 符号 |
|---|---|---|---|
| A | 12 | | 0 |
| B | 5 | | 10 |
| C | 2 | | 110 |
| D | 1 | | 111 |




データによって桁数が変わる
→「可変長」とも言う

5. 道順をたどっていった「0」と「1」の符号を文字に割り当てる

4. 新たにできた「節」の値を1つの文字の頻度に見立て、
   2, 3を繰り返す

3. 2つの頻度の和（この場合は1+2）を求め新たな「節」を作る

2. 一番頻度の少ない2つを取り出し、0と1に振り分ける
   （どちらを1とするかは任意だが、下を1とすると分かりやすい）

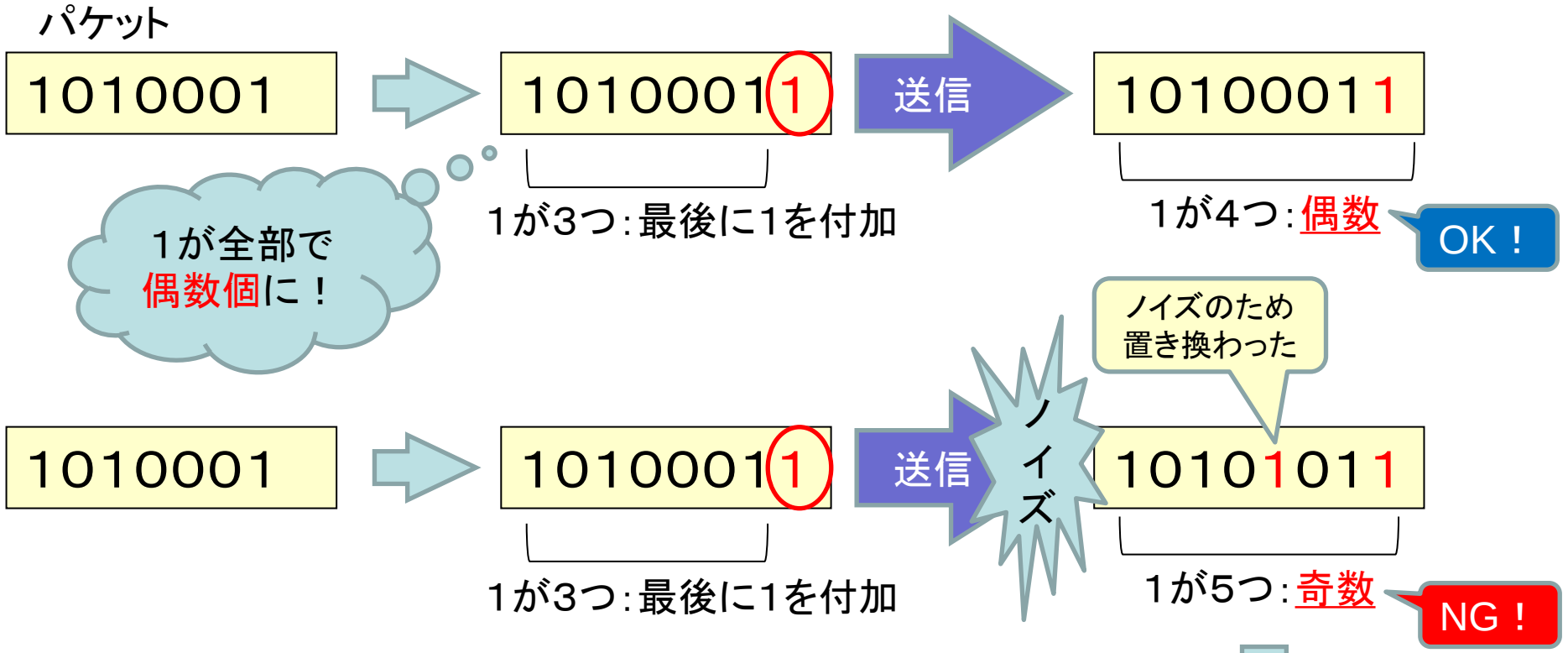
1. 頻度（確率）が高い順に並び替える

# ファイルの圧縮

- 圧縮ソフトウェアを用い、ファイル自体を圧縮
  - ZIP方式 ・・・ windows標準対応（右クリック）
  - LZH方式 ・・・ パソコン通信時代によく利用
  - RAR方式 ・・・ データ破損にある程度まで対応
- 学校のサーバにある「圧縮練習用１」と「圧縮練習用２」を、実際にzip形式で圧縮し、データ量を比較してみよう。
  - 圧縮したいファイルを右クリックし、「送る」から「zipフォルダ」で圧縮できます。

# データの誤り検出、訂正

1の数が奇数個の場合は「1」を付加し、偶数個の場合は「0」を付加して、
1の数が全体で偶数個になるようにする。（垂直パリティチェック）



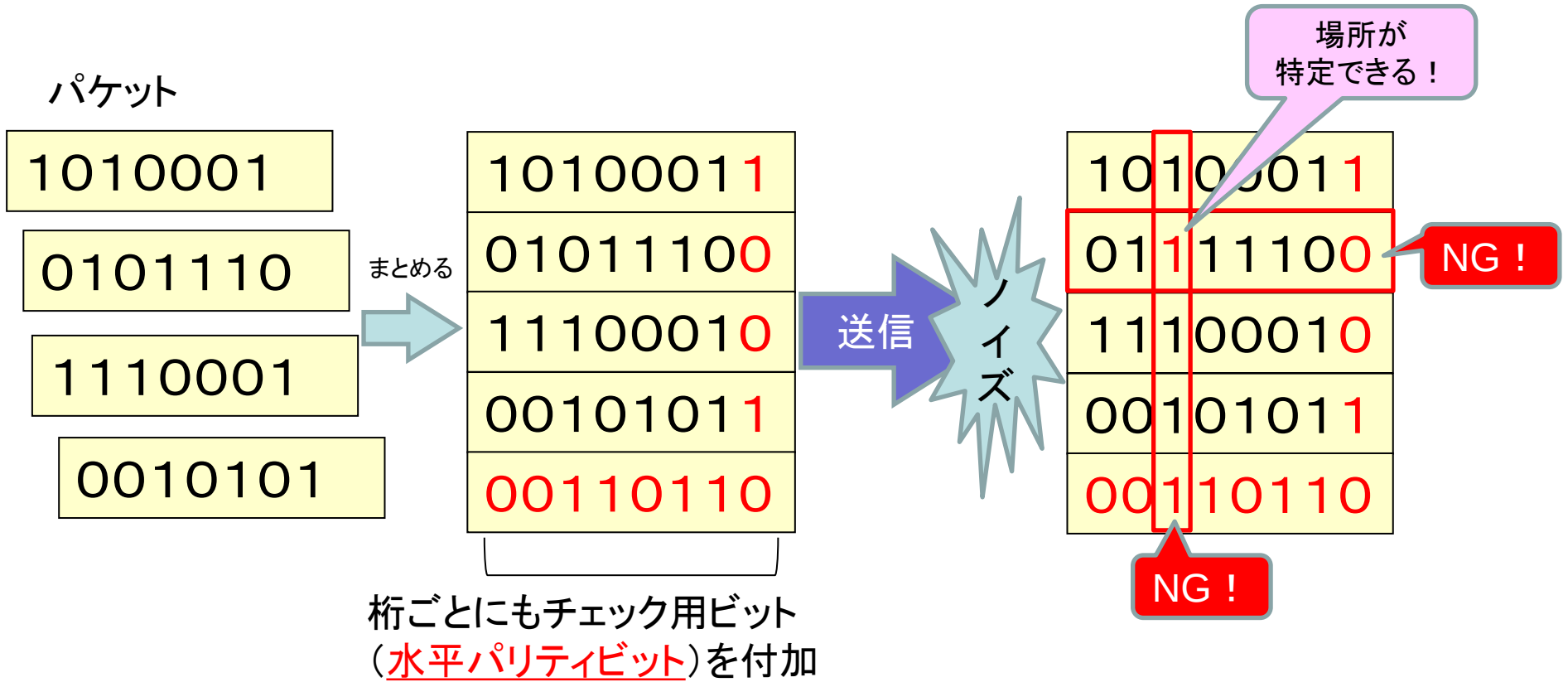

※偶数個にすることを「偶数パリティチェック」という。
（奇数個にする「奇数パリティチェック」もある）

※垂直パリティチェックでは、「誤り」があることはわかるが、
どこが誤っているかまではわからない！

破棄して再送信！

# データの誤り検出、訂正

複数のビット列をまとめ、データ列の桁ごとに、さらにチェック用ビットを追加することによって、誤りの箇所を特定し、訂正することができる。（水平パリティチェック）




※複数の誤りがある場合、それが違うビット列や桁であれば、ある程度絞り込めるが、同じビット列や同じ桁に複数の誤りがある場合、特定することはできない。