目次

★3　パターン情報の表現 (2) — データ圧縮

★3.1　パターンのデータ量を見積もってみよう

パターンを表現するディジタルデータのデータ量がどれくらいになるのか見積もってみよう．CD に記録された音楽データの例，画像の例 (授業中に示します)．

★3.2　データ圧縮とは

上で説明した例からもわかるように，パターンを表現するディジタルデータのデータ量は非常に大きくなりがちである．そのようなデータをそのまま記録したり通信したりするのでは記憶容量や通信帯域が勿体ないので，**データ圧縮**を行うことが一般的である．

勿体ない: mottainai.
データ圧縮: data compression

データ圧縮とは，あるデータが与えられたときに，それが含む情報の性質を損なわないでよりデータ量の小さいデータに変換する処理のことである (☆ 1)．あるデータをより小さいデータ量のデータに変換することを「**圧縮**する」といい，圧縮されたデータを元のデータに戻すことを「**伸長**する (あるいは**展開**する)」という．「**符号化**する／**復号**する」ということもある (☆ 2)．元のデータと圧縮したデータのデータ量を比べて，前者より後者が十分小さければ「**圧縮率**が高い」といい，そうでなければ (あまり変わらなければ)「圧縮率が低い」という．

☆ 1) **データ圧縮の対象はパターンに限らないことに注意．多様な種類の情報に適用される，非常に重要な情報処理の方法である．**

☆ 2) 本来,「符号化」には標本化・量子化のような処理も含む広い意味がある (第 3 回「よだんだよん」も参照) が,「圧縮」の意味で使われることも多い．ちなみに,「復号」は動詞なので「化」はつけない．俗に「解凍する」ということも．

★3.3　データ圧縮手法の分類

データ圧縮の手法には様々なものがあり，いろいろな観点から分類することができる．例えば，全てのデータ圧縮手法は次の二つに分けられる．

可逆圧縮 逆が可能，つまり，圧縮したデータを伸長したら元のデータと完全に同じものが得られるようなデータ圧縮の方法．プログラムやテキストデータを対象とするなら一般にこちらを用いる．

不可逆圧縮 伸長しても元のデータを完全に再現できるとは限らない方法．その分，可逆圧縮よりも高い圧縮率を実現できる．音声や画像といったパターン等，情報の欠落や変化が多少あっても許容できる性質のデータに用いられる．

可逆圧縮: lossless compression, 不可逆圧縮: lossy compression

身近なところでは，コンピュータ上のファイルを圧縮するツール (☆ 3) は可逆圧縮を行なっている．静止画像のファイル形式では，PNG は可逆圧縮であるが，JPEG は不可逆圧縮である (☆ 4)．映像データの形式である MPEG や，オーディオデータの形式である MP3 なども不可逆なデータ圧縮方式を採用している (☆ 5)．

☆ 3) `compress`,`gzip`,`bzip2` といった UNIX 系 OS のコマンドや, MS-DOS や Windows で良く用いられる LHA などのソフトウェア，ZIP 形式を扱うツールなど．

☆ 4) PNG: Portable Network Graphics. JPEG: Joint Photographic Experts Group (この方式を作った組織の名前でもある)．JPEG は可逆手法と不可逆な手法を組み合わせているので全体として不可逆．ただし，可逆圧縮のみ行うことも可能．

☆ 5) MPEG: Moving Picture Experts Group. MPEG-1, MPEG-2 など様々な規格がある．DVD-Video は MPEG-2 をベースとしている．MP3 はもともと MPEG-1 のオーディオデータの規格．

可逆圧縮の手法は，次の二種類に分けることができる．

- データ値の関連性を利用する手法: データ中の値の並び方（同じ値が連続している，近い値が連続している，文章中の単語のように一定の並びが繰り返し出現する，など）を利用する．後述のランレングス圧縮や，LZ 符号化 (☆ 6) などが代表的．
- データ値の偏りを利用する手法: 次回登場予定のエントロピー符号化法（ハフマン符号化や算術符号化が代表的）など．

☆ 6) 1970 年代後半に Ziv と Lempel が提案した手法．様々に改良され，現在も幅広く使われている．有名な LHA もこれを用いる圧縮ツールの一つ．

いずれの手法も，データの種類（画像であるとか音声であるとかテキストであるとか）によらず幅広く用いられる．不可逆圧縮したデータをさらに圧縮するのに用いることも多い．

一方，不可逆圧縮の手法は，対象とするデータの種類を（静止画像向け，音響信号向け，などのように）限定し，データの性質を利用して高い圧縮率を実現しているものが多い．可逆圧縮手法のように一般的な性質でもってそれらの手法を分類することもできるが，割愛する．

Q1. 自分の持っている画像データがどんな形式で保存されているか調べてみよう．ウェブ上の画像がどんな形式か調べてみよう．単純には，そのファイルの拡張子を見ればよい (☆ 7)．おそらく，JPEG や PNG が多いだろう．それぞれどのような画像に使われることが多いか，どのような画像に向いているかを調べてみよう．

☆ 7) 拡張子とファイル形式との対応は，ネットや書籍で調べてみよう．ただし，拡張子が間違っている／わざと変えてある（初歩的なコンピュータウィルスなどの手口）こともあるが．

Q2. 計算機室の Linux 環境で，適当なファイルを圧縮してみよう．

```
$ ls -l hoge      ← 圧縮前のサイズ（バイト単位）を調べる
$ gzip hoge       ← hoge を圧縮．hoge.gz というファイルができる
$ ls -l hoge.gz   ← 圧縮後のサイズを調べる
```

伸長の仕方は，

```
$ man gzip
```

して調べよう．ファイルの種類（プログラムのソース，実行形式，JPEG 画像，etc.）によって圧縮率が違ったりするだろうか．

Q3. 画像を扱える適当なソフトウェア（フリーで手に入るものも多い） を用いて，適当な画像を様々な形式で保存してみよう．圧縮率をいじれるならば，ファイルサイズと見た目がどのように変化するかいろいろ試してみよう．

計算機室の Linux 環境なら，ImageMagick (画像処理ツール群) が使える．ImageMagick の `display` コマンドを使って

```
$ display hoge.jpg
```

とすれば画像 `hoge.jpg` を画面に表示できるし，そこからマウスクリックしてメニューを開いて画像形式を選択して保存することもできる．ImageMagick には他にも画像の大きさや形式を変換する `convert` などがある．

★ 3.4 データ圧縮の例—ランレングス圧縮

ある地域の天気を 1 日毎に「晴」「曇」「雨」「雪」「霙」(☆ 8) などの 8 種類に分類したデータがあったとしよう．例えば，

晴 晴 晴 晴 晴 晴 晴 曇 曇 雨

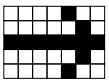
といったものである．このように同じ値が連続して現れることの多いデータの場合，値とその連続する数をペアにして

晴 7 曇 2 雨 1

のように表すと，データ量を減らせそうである．このようなアイデアに基づくデータ圧縮の手法を，**ランレングス圧縮**（ランレングス符号化）という．

「減らせそう」でごまかさないで具体的に考えてみよう．この例では，1 日の天気の値を 3bit で表現することができる．その場合，元のデータを表すためのデータ量は $3 \times 10 = 30$[bit] となる．一方，ランレングス圧縮したデータの方は，値だけでなくその連長も符号化する必要がある．例えば長さも 3bit で表現するとしたなら (☆ 9)，データ量は $(3 + 3) \times 3 = 18$[bit] ということになる．したがって，この例ではランレングス圧縮を行なうことでデータ量を元の 6 割にまで削減できている．

ランレングス圧縮は，**二値画像**（白と黒の二通りの画素で構成された画像）を扱うのに特に適しているので，ファクシミリ等に応用されている（例えば文書などを読み取って二値画像にすると，白画素ばかりになるから）．例えば右図の画像 の画素値を左上から右に向かって順に符号化していくと「白 4 黒 1 白 7 黒 1 白 1 黒 7 白 5 黒 1 白 5 黒 1 白 2」のように表せるが，二値かつ白から数えると仮定すれば，さらに省略して「4 1 7 1 1 7 5 1 5 1 2」としてもちゃんと復号できる．



上述のことからわかるように，ランレングス圧縮のアルゴリズムは非常に単純である (☆ 10) しかし，単純なランレングス圧縮の方法には次のような問題点がある

- 同じ値があまり続かない場合には逆にデータ量を増やしてしまう
- 同じ値が長く続くからといってその長さをそのまま符号化すると，長さを表すために大きな bit 数を割り当てる必要が生じて圧縮率が落ちてしまう

そのため，実用的にはもう少し凝ったアルゴリズムを考える必要がある (☆ 11)．

Q4. 以下は，7×7 の格子状に並んだ画素の値（白か黒）をランレングス圧縮して得られるデータである．ただし，左上の画素から右に向かって符号化してある（最初は白の数）とする．これを伸長するとどんなパターンが得られるか．(^_-)
815111137151158

★ 3.5 補足

★ 3.5.1 データ圧縮の参考書

「圧縮アルゴリズム 符号化の原理と C 言語による実装」昌達 K'z，ソフトバンクパブリッシング，ISBN4-7973-2552-6

☆ 8) 霙：みぞれと読むそうです．

ランレングス: run-length. 連長ともいう．

☆ 9) 3bit なら 1 から 8 までの長さを表せる．

☆ 10) プログラミングの好きな人は，例えば，整数のならびをキーボードから入力するとそれをランレングス圧縮したものを表示するプログラムを考えてみると楽しいでしょう．手頃な練習問題になります．

☆ 11) 興味のある人は，データ圧縮関連の文献などを調べてみるとよい．下記の参考文献には，これらの問題点に対応するための改良版ランレングス圧縮を含め各種データ圧縮手法の C プログラムが載っている．

★ 3.5.2 ビットレート

画像や音声のように時間を変数とするパターン等のデータを何らかの（例えば USB ケーブルや無線 LAN 等）を通じて転送することを考えると，「1 秒分のデータを転送するには毎秒何 bit の転送速度が必要か」ということが気になる．使用する媒体の転送速度がその要求を上回っているなら，画像や音声等をリアルタイムに転送して再生できると想定できるからである．この，「1 秒分のデータに必要なデータ量」を**ビットレート** といい，ビット毎秒 (bps) という単位で測る．

ビットレート: bitrate

bps: bit par second

例えば，CD 品質の音響データの場合，1411200[bps] = 1411.2[kbps] となる (☆12)．一昔前の携帯電話の通話音声や MP3 の音楽データなどに必要なビットレートは，数十 kbps 程度である．

☆ 12) ビットレートの単位では，k や M といった接頭辞は 2 のベキではなく 10 のベキに基づくものであることに注意．

★ 3.6 まとめの問題

Q5. この授業のウェブページ上に掲載されている問題を参照してください．

Q6. 音声と動画像をディジタルデータとして取得できる情報機器 A がある．A は，音声を CD-DA の規格に従ってディジタル化する．一方，動画像は次の条件に従ってディジタル化する．

- 30 フレーム／秒（1 秒あたり 30 枚の静止画像から成る）
- 幅 1000 画素，高さ 1000 画素
- 画素値は赤緑青それぞれ 8bit ずつ

A をデータ転送装置 B を介して他の情報機器と接続し，取得した音声と同画像のデータを遅延させずに転送（つまり，1 秒分のデータを 1 秒以内で転送）したい．次の問に答えなさい．A でのデータ生成や A と B との間のやりとりなどに要する時間は無視して，データ転送にかかる時間のみを考えればよい．A が取得したデータは圧縮せずそのまま転送するものとする．

(1) A が生成する音声データのみを遅延させずに転送したい場合，B のデータ転送路のビットレートは 1411.2 kbps 以上あればよい．CD-DA 規格の数値を用いた計算によってその理由を説明しなさい．

(2) A が生成する動画像データもあわせて遅延させずに転送したい場合，このデータ転送路のビットレートは何 Mbps 以上あればよいか．

★ （ここは宿題の範囲外） 興味のあるひとは，USB，無線 LAN，LTE，小惑星探査機はやぶさがピンチのときに使用した低利得アンテナ，等の転送速度がどの程度か調べてみるとよい．ただし，細かい規格によって値は様々（例えば USB2.0/3.0/3.1 で異なる）であること，実際の性能ではなく理論上の最大値を示している場合が多いこと，に注意．