*QR コードを作って学ぶ情報数理入門

2025 年 6 月 21 日　@岐阜県先端技術体験センター

執筆: 土本幸多[1)]
協力: 小口将平[2)]

目次

1　はじめに

この資料は富山大学理学部数学教室の幸山直人先生の公開講義資料に着想を得て，岐阜県先端技術体験センターで行われるサイエンスフェア２０２５のために作成したものです．今日ご参加いただいた皆さんに少しでも数学や情報学の魅力が伝われば嬉しく思います．

1)　所属：富山大学理学部数理情報学プログラム２年
2)　所属：岐阜大学社会システム経営学環２年

2 大まかな手順 （変換プログラムを使う方は手順８から）

1. 漢字を **シフト JIS コード**[3] に変換する
2. シフト JIS コードを **13 ビット**[4] の **2 進数**[5] に変換する
3. **モード指示子**[6] (漢字モード：1000)，**文字数指示子**[7] (文字数を８ビットで表現) を先ほど変換したデータの先頭に付ける
4. 手順 3 で作ったデータの末尾に**終端パターン**[8] (0000) を付ける
5. 手順 4 で作ったデータを８ビットずつに分ける．もし最後に数字が足りない場合（例えば５ビットだけ余るときなど）は末尾に 0 を付けて８ビットにする．[9]
6. 手順 5 で作ったデータが**１９バイト**[10] になるまで，**埋め草コード**[11] 11101100 および 00010001 を交互に付け足す
7. （**上級者向けの内容です**第４章の詳しい手順を参照）手順 6 で求めたデータから**誤り訂正コード語**[12] を計算し，元のデータと合わせ，**RS 符号**[13] を計算する
8. 手順７で求めたデータ（もしくは変換プログラムで求めたデータを）を３バイトずつに分けて，３バイト毎それぞれ交互に 10011001 と 01100110 と**論理的排他和**[14] をとる
9. 手順８で求めたデータを QR コードの対応するマスに 0 は白（塗らない）1 は黒で配置する．

3） 漢字を 1 文字 1 文字を漢字に書き換える決まり事
4） 0 もしくは 1 が二つ合計 13 個並んだ数字のこと．詳しくは第 2 章を参照
5） 0 と 1 のみで表現した数字．詳しくは第 2 章を参照
6） 機械に何を表現した数字であるかを伝える数字
7） 機械に文字数を教える数字のこと
8） 機械に文字列の終わりを伝える数字
9） 3 文字の人は 0 を 5 文字の人は 0000000 をそれぞれ補って 8 ビットにする
10） １ビットが１９個並んだ形．詳しくは第２章を参照
11） 意味を持たない数字列のこと
12） 伝達ミスがあっても，言いたいことが伝わる魔法の言葉．詳細は第３章を参照
13） リードさんとソロモンさんが開発した暗号．これのお陰で*QR コードが成り立つ．詳細は第４章
14） 0+0=0,1+0=1,1+1=0 として和をとること．

3 用語のちょっとだけ詳しい説明

3.1 シフト JIS コード

コンピュータは「あ」や「A」といった文字をそのまま理解することはできません. コンピュータが理解できるのは, 究極的には「ON」と「OFF」のような電気信号の状態だけです. そこで, 人間が使う文字とコンピュータが理解する数字を結びつけるための「対応表」が作られました. シフト JIS コードとは, その対応表の一種です.
例えば, 「あ」という文字には「82A0」, 「A」という文字には「41」というように, 世界中の文字や記号の一つ一つに”ユニーク”（ただ一通り）な番号が割り振られています. この番号を使うことで, コンピュータは文字を区別し, 記憶したり表示したりすることができるのです.QR コードで文字を送るためには, まずこの対応表を使って文字を数字に変換する必要があります.

3.2 2 進数

私たちは普段, 0 から 9 までの 10 個の数字を使って数を数えます（10 進数）. しかし, コンピュータは「ON」と「OFF」の 2 つの状態しか持たないため, 0 と 1 の 2 つの数字だけで数を表現する「2 進数」を使います. これは, 電球のスイッチに似ています. スイッチが OFF なら「0」, ON なら「1」です.

- 電球が 1 個なら, 0 と 1 の 2 通りを表現できます.
- 電球が 2 個なら, 00(OFF,OFF), 01(OFF,ON), 10(ON,OFF), 11(ON,ON) の 4 通りを表現できます.

このように, スイッチ（数字の桁）を増やすことで, どんな大きな数でも 0 と 1 の組み合わせだけで表現できるのです.*QR コードの黒い点と白い点は, まさにこの 2 進数の 1 と 0 に対応しており, 膨大な情報を小さな絵の中に詰め込むことを可能にしています.

ちょっと挑戦！

それでは,2 進数の考え方に慣れるために, 簡単なクイズに挑戦してみましょう！

1. 私たちが普段使う 10 進数の「10」は,2 進数で表すとどうなるでしょうか？
2. 2 進数の「101」は,10 進数で表すといくつになるでしょうか？

☑ こたえと解説

問題 1 の答え: 1010

解説: 10 進数を 2 進数に変換するには,「2 で割って余りを記録する」という作業を, 商が 0 になるまで繰り返します.

- $10 \div 2 = 5$ ... **余り 0**
- $5 \div 2 = 2$ ... **余り 1**
- $2 \div 2 = 1$ ... **余り 0**
- $1 \div 2 = 0$ ... **余り 1**

出てきた余りを**下から順に**並べると,「1010」になります.

問題 2 の答え: 5

解説: 2 進数を 10 進数に直すには, 各桁の「位の重み」を考えます. 右の桁から順に 1 の位, 2 の位, 4 の位, 8 の位... と, 位が一つ上がるごとに重みが 2 倍になっていきます. 2 進数「101」は,

- (1 の位が **1** 個) + (2 の位が **0** 個) + (4 の位が **1** 個)

ということを意味しているので,$(1 \times \mathbf{1}) + (2 \times \mathbf{0}) + (4 \times \mathbf{1}) = 1 + 0 + 4 = 5$ となります.

3.3 ビット・バイト

「ビット」と「バイト」は, コンピュータが扱う情報の量を表す単位です.

- **ビット (bit)**: 2 進数の 1 桁分の情報量のことです. つまり, 0 か 1 かのどちらかを表す, 情報の最小単位です. 電球の例で言えば, 電球 1 個が 1 ビットに相当します.
- **バイト (byte)**: ビットが 8 個集まったものを 1 バイトと呼びます.1 ビットでは 2 通りしか表現できませんが, 8 ビット（=1 バイト）あれば, $2^8 = 256$ 通りの情報を表現できます. これにより, アルファベットや数字, 簡単な記号などを十分に表現することができます.

日本語の漢字のようにたくさんの種類の文字を扱うには, さらに 2 バイト（=16 ビット）を組み合わせて使います. 今回の QR コード作成手順で「8 ビットごとに区切る」という作業が出てくるのは, このバイトという単位に情報を整理し直しているためです.

ちょっと挑戦！

ビットとバイトの関係を確認してみましょう.

問題: スマートフォンのアプリのサイズが「2 メガバイト (MB)」と表示されていました. これは, およそ何ビット (何個の 0,1 で作られているでしょう？) の情報量になるでしょうか？（※ヒント：1 メガバイトは 100 万バイトとして計算してみましょう）

【余白】[15)]

15) フェルマーの最終定理で有名な数学者ピエール・ド・フェルマーは, 自分の本に「私は真に驚くべき証明を見出したが, それを書くには余白が狭すぎる」という言葉を残したそうです. 皆さんもテストで余白が狭かった時には, この言葉を思い出してみてください. くれぐれも解答に書かないように！

☑ こたえと解説

答え: 約 1600 万ビット

解説:

1. まず, メガバイトをバイトに直します.1 メガバイトを 100 万バイトとすると,2 メガバイトは
   $2 \times 1,000,000 = 2,000,000$ (200 万) バイトです.
2. 次に, バイトをビットに直します.1 バイトは 8 ビットでしたね.
3. したがって,200 万バイトは,
   $2,000,000$ バイト $\times 8 = 16,000,000$ ビット
   となり, 答えは約 **1600 万ビット**となります.

たった 2 メガバイトのアプリでも, これだけ膨大な数の 0 と 1 の集まりでできていると考えると, コンピュータのすごさが感じられますね！

3.4 誤り訂正コード語

QR コードの一部が汚れたり, かすれて読めなくなったりしても, スマートフォンで正しく読み取れることが多いのはなぜでしょうか？ それは「誤り訂正コード」という, いわば「魔法の呪文」が一緒に記録されているからです.
これは, 大事な伝言をするときに, 「A 君は公園に行ったよ. そう, 公園に行ったのは A 君なんだよ」というように, 表現を変えて（全く同じ内容の）情報を付け加えるのと似ています. もし一部分を聞き逃しても, 付け加えられた情報から元の内容を推測できます.
QR コードでは, 記録したい本来のデータ（名前など）から, ある数学的な計算（詳細は第 5 章で解説）によって, この「魔法の呪文」にあたる特別なデータ（誤り訂正コード語）を生成します. そして, 本来のデータと誤り訂正コード語の両方を QR コードに記録します. 読み取り時に一部のデータが欠けていても, この誤り訂正コード語を手がかりに, まるでパズルを解くように元のデータを復元することができるのです.

4 なぜ？なに？の説明

4.1 なぜ，数字だけで”伝わる”のか？

私たちが普段何気なく使っている「言葉」や「文字」は, 実は非常に複雑な概念です. コンピュータは, このような曖昧さを含むものを直接理解するのが苦手です. その代わり, コンピュータが得意なのは, 寸分の狂いもなく計算することです.

そこで，「あ」という文字を「82A0」という数字に置き換える，というように「ルール（約束事）」を決めておけば，コンピュータはそのルールに従って，文字を数字として扱うことができます.QR コードを読み取るスマホのカメラも，白黒のパターンを 0 と 1 の数字の列として読み取り，その数字の列を約束事に従って文字に再変換することで，私たちが読める情報として表示しているのです.
つまり，「文字→数字」と「数字→文字」という共通のルールがあるからこそ，数字だけのやり取りで情報が”伝わる”のです.
ご両親やお友達と以下のゲームに挑戦してみましょう.

ちょっと挑戦！

ゲーム：数字で伝える「１」と言われたら，座ってる人は立ち上がり，立っている人は座る.「０」と言われたら，座ってる人はそのまま座り，立っている人はそのまま立つ.[a]
数字しか言っていないのに，「立ってほしい」「座ってほしい」「そのままで」ということが伝わっていますね！
これは，数字が「立つ」「座る」という行動を表すルールを共有しているからです.

a なぜ，このような数字設定にしたと思いますか？それは，論理的排他和をとれば対応した行動になるからです！気になった方は気軽に質問してね！

4.2 なぜ，０と１だけで表現するのか？

コンピュータの内部では，無数の小さなスイッチが電気で ON になったり OFF になったりしています. この「ON」と「OFF」という 2 つの状態は，最も単純で，かつ間違いなく区別できる状態です. 例えば，電圧が高い状態を「1(ON)」，低い状態を「0(OFF)」と決めておけば，非常に高速かつ正確に動作することができます.
もし，0 から 9 までの 10 段階の状態を区別しようとすると，10 段階の微妙な電圧（もしくは電流）の違いを正確に検知する必要があり，回路は非常に複雑で高価になり，ちょっと電圧や電流がズレるだけで簡単にエラーを起こしてしまいます. 最もシンプルで信頼性の高い「ON/OFF」の 2 状態に絞り，それを 0 と 1 に対応させる「2 進数」を使うことが，今日のコンピュータ技術の根幹を支えているのです.

4.3 なぜ，モード指示子，文字数指示子，終端パターンが必要なのか？

QR コードのデータを読み解くのは，まるで外国語で書かれた巻物を解読するようなものです. 何もヒントがなければ，どこからどこまでが本文で，何語で書かれているのかすら分かりません. そこで，解読のためのヒントが必要になります.

- **モード指示子**: 巻物の冒頭にある「これは日本語で書かれている」という注意書きです.QR

コードでは「これから来るデータは漢字ですよ (1000)」「数字ですよ (0001)」といった種類を最初に宣言します. これによって, 読み取り側は正しい翻訳ルール（シフト JIS など）を準備できます.

- **文字数指示子**: 「この巻物には 4 文字書かれている」という宣言です. データの長さをあらかじめ知ることで, 読み取り側はどこまで読めば本文が終わるのかを正確に把握できます.
- **終端パターン**: 「以上, 巻物おしまい」の印です. 文字数指示子と合わせて二重にチェックすることで, 確実にデータの終わりを認識し, 読み落としや読み過ぎを防ぐことができます.

これらの「ヒント」があるおかげで, QR コードは膨大な 0 と 1 の羅列の中から, 意味のある情報を正確に取り出すことができるのです.

4.4 なぜ，埋め草コードが必要なのか？

QR コードは, バージョン（大きさ）ごとに, データを入れられる「箱」のサイズが決まっています. 例えば今回の QR コードでは, 誤り訂正コードの分を除くと, ユーザーが使えるデータ領域は 19 バイトという決まった大きさの箱になっています. [14]
しかし, 私たちが QR コードにしたい文字数によっては, 作成したデータが 19 バイトに満たない場合があります. 例えば, 15 バイトにしかならなかったとします. 箱に 4 バイト分の隙間ができてしまいます. この隙間をそのままにしておくと, QR コードのルール全体が崩れてしまい, 正しく読み取れなくなります.
そこで, この隙間を埋めるためだけに使われるのが「埋め草コード」です. これは, 新聞紙を丸めて箱の隙間に詰めるようなもので, それ自体に意味はありませんが, 箱の中身をきっちり固定し, ルール通りのデータ長にするために不可欠な存在なのです.

4.5 なぜ，わざわざ 19 バイトにするのか？

この「19 バイト」という数字は, 我々が勝手に決めたものではなく, QR コードの規格によって定められています. 具体的には, 今回作成する QR コードの仕様である「型番:1, 誤り訂正レベル:L」の組み合わせによって決まっています. [10]
QR コードの規格書には, 型番（バージョンのこと, 1 から 40 まである）と, 誤り訂正レベル（L, M, Q, H の 4 段階）の全ての組み合わせについて,

- 全体で何バイト分のデータを格納できるか（総データ容量）
- そのうち, 誤り訂正コードに何バイト分を割り当てるか

が厳密にリストアップされています.「型番 1, 誤り訂正レベル L」の場合, 総データ容量は 26 バイト, そのうち誤り訂正コードに 7 バイトが割り当てられます. したがって, 残りの $26 - 7 = 19$ バイトが, 私たちが文字などの情報を格納するために使える領域となるのです. [14]

4.6 なぜ，四隅３箇所に四角形があるの？

QR コードをよく見ると, 必ず 3 つの隅に大きな「回」の字のような四角形がありますね. これは位置検出パターン（またはファインダーパターン）と呼ばれる,QR コードが自分自身の位置を正確に把握するための, 非常に重要な「目印」です .
スマートフォンなどのカメラで読み取る際, 私たちは QR コードを真上からぴったり撮影しているわけではありません. 斜めから撮ったり, 少し離れて撮ったり, 時には逆さまの向きで撮ろうとすることもあります. この位置検出パターンが 3 箇所にあるおかげで, 読み取りプログラムは

- QR コードが画像のどこにあるのか
- どれくらいの大きさで写っているのか
- どのように傾いているのか
- 上下左右の向きはどちらか

といった情報を,360 度どの角度からでも瞬時に, そして正確に計算することができます.
ちなみに, なぜ 4 つではなく 3 つなのでしょうか？ もし四隅すべてに同じ目印があると,180 度回転させたときに区別がつかなくなってしまいます.3 つにすることで, 向きが一意に定まり,「こちらが上」ということを確実に判断できるのです.

4.7 初めから塗ってあるマスは何？

四隅の大きな目印の他にも, よく見ると最初から白と黒の点が交互に並んだ線のような模様があることに気づくでしょう. これも, データを正しく読み取るための重要な部品です.

- **タイミングパターン**: 位置検出パターンの間を結ぶように配置されている, 白黒の点線模様です . これは,QR コードがもし歪んで印刷されていたり, 曲面に貼られていたりしても, 一つ一つのマスの座標を正確に特定するための「ものさし」の役割を果たします. このものさしを基準にすることで, プログラムはズレることなくデータを読み進めることができます.
- **形式情報**: 位置検出パターンのすぐ近くに配置されている, とても大事な情報が書き込まれた領域です . ここには, この QR コードがどれくらいの誤り訂正能力を持っているか（誤り訂正レベル）や, どのようなマスクパターン（後で説明します）が使われているか, といった QR コードの「自己紹介」データが記録されています . 読み取りプログラムは, まずこの形式情報を解読し, そのルールに従ってデータ全体の解読作業に進みます.

これらのパターンは,QR コードという設計図の「基礎」となる部分です. そのため, ユーザーがどんなデータを入れるかに関わらず, あらかじめ決まった場所に固定で配置されているのです.

5 上級者向け内容 (詳しい手順)

第 2 章の手順 7「誤り訂正コード語の計算」は, **リード・ソロモン符号 (Reed-Solomon code)** と呼ばれる強力な誤り訂正符号の理論に基づいています. この計算は, ガロア体 $GF(2^8)$ という特殊な数の世界で行われます.

1. **データ多項式の生成**:
   まず, 手順 6 で作成した 19 バイトのデータ列を, 多項式の係数と見なします. 各バイトを $GF(2^8)$ の元（要素）に変換し, これらを係数とする 18 次の多項式 $I(x)$ を作ります.

   $$I(x) = d_{18}x^{18} + d_{17}x^{17} + \cdots + d_1x + d_0$$

   ここで, 式の係数 d_i は 19 バイトのデータを順に $GF(2^8)$ の元に変換したものです.
2. **生成多項式の準備**:
   次に, 誤り訂正コードを生成するための「種」となる生成多項式 $G(x)$ を用意します. 今回作成する QR コード (型番 1, レベル L) では,7 バイト分の誤り訂正能力が必要とされ, そのための生成多項式は $GF(2^8)$ の原始元 α を用いて次のように定義されています.

   $$G(x) = (x-\alpha^0)(x-\alpha^1)(x-\alpha^2)(x-\alpha^3)(x-\alpha^4)(x-\alpha^5)(x-\alpha^6)$$

   これは展開すると 7 次の多項式になります.
3. **剰余の計算**:
   誤り訂正コード語とは, 実は多項式の割り算の余りに他なりません. 具体的には, データ多項式 $I(x)$ に, 生成多項式の次数である 7 次分のスペースを空けるため x^7 を掛けたもの $(I(x)x^7)$ を, 生成多項式 $G(x)$ で割ります. このときの余り $R(x)$ が, 求める誤り訂正コード語の多項式表現です.

   $$R(x) = (I(x) \cdot x^7) \pmod{G(x)}$$

   この割り算の加減乗除は, 全て $GF(2^8)$ のルールに従って行われます.
4. **符号語の完成**:
   最後に, 元のデータと計算した誤り訂正コード語を合体させます. これは, 多項式で表現すると, $I(x)x^7$ から $R(x)$ を引く（$GF(2)$ 上では足し算と同じ）ことに対応します.

   $$X(x) = I(x) \cdot x^7 - R(x) = I(x) \cdot x^7 + R(x)$$

   この $X(x)$ の係数を並べたもの（元の 19 バイトのデータの後に,$R(x)$ の係数である 7 バイトの誤り訂正コード語を連結したもの）が, 最終的な 26 バイトの符号語となり,QR コードのマス目に配置されるデータとなります.

6 上級者向け説明 (一部，数学系の学生向け)

6.1 ガロア体とは？

ガロア体 (Galois Field), または有限体 (Finite Field) とは, その名の通り「要素の個数が有限である体」のことである. 体とは, 私たちがよく知る有理数体 $\mathbb{Q}$ や実数体 $\mathbb{R}$ のように, その集合の中で自在に四則演算（ただしゼロ除算は除く）が定義され, 矛盾なく閉じている代数系 (演算と数の集合) を指します. e.g. 有理数体 $\mathbb{Q}$ の任意の二元 a, b に対して, 加法 $a+b$, 減法 $a-b$, 乗法 $a \cdot b$, 除法 $a/b(b \neq 0)$ が定義され, その結果もまた有理数であることが簡単に分かります.

QR コードで用いられるのは, 位数 (何回同じ演算したら単位元 (足したり掛けても元の数字を変えない数字) に戻るのかという数)[16] p^m (p: 素数, m: 正整数) のガロア体 $GF(p^m)$ のうち, 特に $p=2$ の場合, すなわち $GF(2^m)$ であり, これはコンピュータの 2 進数の世界と非常に相性が良いためであると考えられます. 具体的には,$GF(2^8)$ という 256 個の元からなる体を利用する. この体の元は,

1. 係数が $\{0,1\}$ (つまり $GF(2)$) からなる,7 次以下の多項式
2. 8 ビットのバイナリデータ (1 バイト)

と同一視することができ, 例えば, 多項式 x^3+x+1 はバイナリデータ ‘00001011‘ に対応する. この対応関係により, バイト単位のデータを数学的な「数」として扱い, 強力な代数理論（符号理論）を適用することが可能になるのです.

6.2 ガロア体の演算の特徴

$GF(2^8)$ における演算は, 通常の数の演算とは異なる. 元を係数が $\{0,1\}$ の多項式と見なして定義されます.

- **加法 (+)**: 多項式の各次数の係数ごとに行われる排他的論理和 (XOR) であり, 通常の多項式の和から繰り上がりをなくしたもの, と考えることができる. この性質から,$GF(2^m)$ の世界では $a+a=0$, すなわち $a=-a$ が成り立ち, 加算と減算は全く同一の操作となるのです.
- **乗法 (×)**: 通常の多項式の乗算を行った後,QR コードの規格で定められた 8 次の原始既約多項式 $P(x)=x^8+x^4+x^3+x^2+1$ (16 進数で ‘11D‘) で割った余りを計算する.

$$A(x) \times B(x) = (A(x) \cdot B(x)) \pmod{P(x)}$$

この剰余演算により, 積が常に 7 次以下の多項式（つまり $GF(2^8)$ の元）に収まることが保証されます. 実用上, この多項式乗算は計算コストが高いため, 原始元 α のべき乗を用いた指数表

16) 実際, ガロア体の元は和を複数回取ることで 0 に戻ります.e.g. $GF(2)$ という体の元 1 は論理的排他和に対して 1+1=0 となり, 加法単位元 0 に戻る.

現（対数テーブル）[17]をあらかじめ作成しておき，乗算を指数の足し算に変換して高速化するのが一般的です．

$$\alpha^i \times \alpha^j = \alpha^{(i+j)} \pmod{255}$$

6.3 ガロア体の演算を使った誤り訂正コード語の計算

「上級者向け内容」で示した誤り訂正コード語の計算，すなわち剰余多項式 $R(x) = (I(x) \cdot x^7) \pmod{G(x)}$ は，ここで述べたガロア体の演算規則に基づいて実行される． この計算は，本質的には多項式の長除法（筆算）と同様であるが，計算過程で登場する係数の加減乗除はすべて $GF(2^8)$ のルールに従い，筆算で現れる「引き算」は，ガロア体の特徴により「足し算（XOR）」として実行される．このようにして得られた最終的な符号語 $X(x) = I(x) \cdot x^7 + R(x)$ は，その計算方法から，**必ず生成多項式 $G(x)$ で割り切れる**という重要な性質を持つことになります． QR コードを読み取る際は，この「$G(x)$ で割り切れるか」を検証することで誤りを検出することになります．

最後に

- 本書の内容は，富山大学理学部幸山研究室のホームページを参考にさせていただきました．
  `https://kouyama.sci.u-toyama.ac.jp/main/index.htm`
- また QR コードの仕様については，以下のサイトを参照しました．
  `http://ik1-316-18424.vs.sakura.ne.jp/category/QRCode/index.html`
- 拙作:QR コード データ符号化ツール (誤り訂正レベル L)
  WebApp:`https://mocho271828.github.io/rs_coding-compresser/`
  GitHub:`https://github.com/mocho271828/rs_coding-compresser`

最後に，この資料の執筆から当日の運営にいたるまで，多くの方々のご指導とご助力を賜りましたこと，心より感謝申し上げます．

17） 実際の対数テーブルをご覧になりたい方は以下の URL からご参照ください．