第 4 章

QR コードを読む

76 回生 watcol

4.1　はじめに

はじめまして。76 回生 (高 1) の watcol です。初めて部誌を書きます。拙い文章ですが、最後まで目を通していただけると幸いです。それでは、早速本題に移りましょう。

4.2　QR コードとは

さて、みなさんは下のようなものを見たことがあるでしょうか。

図 4.1: QR コード

近年見かけることの多い (この文化祭の会場でもよく見かけますね) この形、ご存知の方も多いと思いますが、QR コードと呼ばれるものです。

ここに URL などの情報を入れておくと、読み取った時に Web サイトへアクセスできたり、LINE などでは友達が追加できたりもします。(ちなみに、上の QR コードを読み取ると NPCA の公式 Web サイト[*1]にアクセスできます。)

この QR コードは「マトリックス型 2 次元コード」と呼ばれるものの一つで、一種のバーコードのようなものです。しかし、一般的なバーコードに比べて多くの情報を詰め込める[*2]ほか、汚れや破損に強く、少しなら壊れていても読み取ることができます (経験がある方もいるのではないでしょうか)。

[*1] https://www.npca.jp

[*2] 数字のみだと 7,089 文字、英語だと 4,926 文字、日本語でも 1,817 文字詰め込めます。

4.3 QR コードを読む

しかし、この QR コードには一つ致命的な欠点があります。それは**人間には理解できない**ということです。これでは手元に携帯やスマートフォンがなければ何が書いてあるか全くわかりません。皆さんも読めるようになりたいですよね??

そこで、この記事では、QR コードの読み方について紹介していこうと思います。

4.4 データの解釈の仕方

QR コードでは、2 進数の 1 を黒いセル (1 つ 1 つのブロックのこと) に、0 を白いセルに置き換え、決まった順番で並べることによってデータを数値の列として表現しています。なので、どんなデータがどのように並べられているか、そしてそれをどのように解釈すればいいのかがわかれば、(理論上は) 我々人間にも QR コードの内容を理解することができます。ということで、まずは QR コードの大まかな構造を見てみましょう。

4.5 QR コードの構造

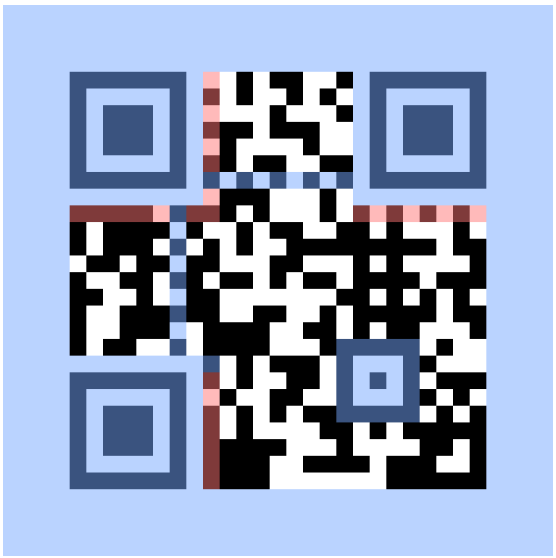

図 4.2: QR コードの構造

図 4.2 を見てください。これは、先程の QR コードをデータの役割ごとに色付けしたものです。早速、それぞれの部分について見ていきましょう。

形が決まっている部分 (水色)

図 4.3: 定常部の詳解図

どの QR コードでも同じ形をしている部分です。次のように分類されます。

①余白領域

「ただの余白??」思われるかもしれませんが、たかが余白、されど余白。この部分がないと正しく読み取ることができません。公式の仕様では上下左右共に最低で 4 セル分必要とされています。

②切り出しシンボル

おなじみの形ですね。機械はこれを見て QR コードを認識しています。ファインダーパターンとも呼ばれます。注意しなければいけないのは、周りの白い部分を含めて切り出しシンボルだということです。

③アライメントパターン

皆さんは②の形が右下にだけ無いのを不思議に思ったことはありませんか？ これは機械的に向きを判定できるようにする機構です。機械でこの形が右下に来るように QR コードを回転させればいいので、私たちは逆さのまま QR コードを読み取ることができるというわけです。

④タイミングパターン

すべての QR コードにこの形があることを知っていた方は少ないのではないのでしょうか? これは QR コードが歪んでいるときやセルの位置がずれているときに、読み取るデータがそのままずれてしまうのを防いでいます。

⑤ダークモジュール

忘れられがちですが、このセルは必ず黒になっています。後述のフォーマット情報領域でセル数を調整するためにあります。

フォーマット情報領域 (赤色)

フォーマット情報領域は、その QR コードについての情報を表す部分です。この情報によって後述するデータ本体の読み方が変わるため、注意して読む必要があります。

図 4.4: フォーマット情報領域の読み方

さて、この領域の読み方ですが、まず上の図のような順番で黒を `1`、白を `0` として 1 セルずつ読んでいきます。このとき①と②では同じ情報になっているはずです。これはどちらかがかくれていたり、汚れていたりする場合でも読み取れるようにするための工夫なのですが、今回はどちらかを無視してもらって構いません。

その後、このデータ (上の QR コードでは `111110110101010` になります。) をそれぞれの桁について `101010000010010` と比べて、次のようなルールで処理します。(このような処理を **XOR 演算**といって、競技プログラミングなどでよくつかわれます。)

- それぞれの数が同じならば、その桁を `0` とします。
- それぞれの数が異なる、その桁を `1` とします。

この例では次のようになります。

```
  111110110101010
  101010000010010
-----------------
  010100110111000
```

`010100110111000` となりましたね。このうち最初の 2 桁は「誤り訂正レベル」、その次の 3 桁は「マスクパターン」と呼ばれる情報です。残りの 10 桁は「誤り訂正符号」と呼ばれる、読み取りが間違っていないか調べるためだけの情報なので無視して構いません。では、この 2 つの情報について詳しく見ていきましょう。

誤り訂正レベル

QR コードがどのくらいの汚れを許容するかという情報が入っていて、低い方から「L」(7% の誤りを許容)、「M」(15% の誤りを許容)、「Q」(25% の誤りを許容)、「H」(30% の誤りを許容) です (ここでのパーセンテージは後述するデータ領域を 100 としたときのもの)。上で求めた数との対応は以下の通りです。

表 4.1: 誤り訂正レベル

コード	誤り訂正レベル
01	L
00	M
11	Q
10	H

誤り訂正レベルが高くなるほど汚れには強くなりますが、その代わりに QR コードが大きくなったり、読み取りに時間がかかったりするので、シチュエーションに合ったものを選ぶ必要があります。

ここまで言っておいてなんですが、正直誤り訂正レベルは QR コードを読むときには関係無いので、今回は無視してもらって構いません。

マスクパターン

QR コードでは、コード内の白と黒のバランスを良くしたり、コード内に誤ってファインダーパターンが出て来たりしないようにするために、後述のデータ領域を加工するのですが、そのために使われるのがこのマスクパターンです。上で求めた数との対応は以下の通りです。

表 4.2: マスクパターン

コード	マスクパターン
000	$(i+j) \bmod 2 = 0$
001	$i \bmod 2 = 0$
010	$j \bmod 3 = 0$
011	$(i+j) \bmod 3 = 0$
100	$((i \div 2) + (j \div 3)) \bmod 2 = 0$
101	$ij \bmod 2 + ij \bmod 3 = 0$
110	$(ij \bmod 3 + ij) \bmod 2 = 0$
111	$(ij \bmod 3 + i + j) \bmod 2 = 0$

ここで、i は一番上の行を 0 としたときの行番号、j は一番左の列を 0 としたときの行番号です。この数式が成り立つようなセルを反転 (白を黒に、黒を白にする) して読む必要があります。

数式を見ても分かりにくいと思うので、図にしてみました。

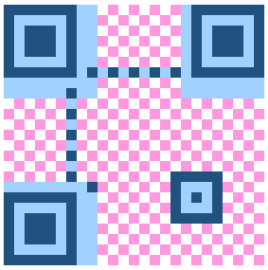

図 4.5: $(i+j) \bmod 2 = 0$

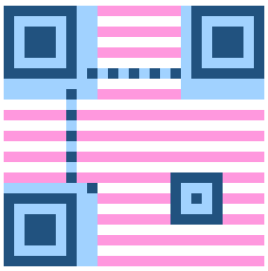

図 4.6: $i \bmod 2 = 0$

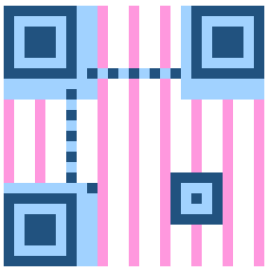

図 4.7: $j \bmod 3 = 0$

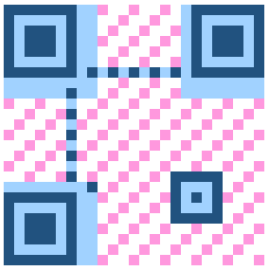

図 4.8: $(i + j) \bmod 3 = 0$

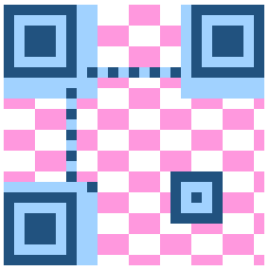

図 4.9: $((i \div 2) + (j \div 3)) \bmod 2 = 0$

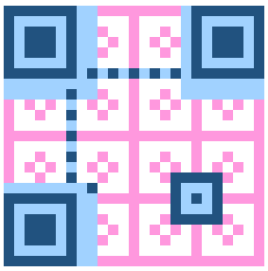

図 4.10: $ij \bmod 2 + ij \bmod 3 = 0$

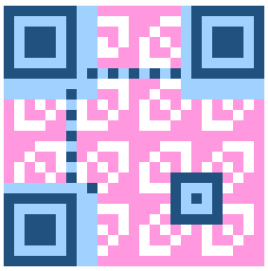

図 4.11: $(ij \bmod 3 + ij) \bmod 2 = 0$

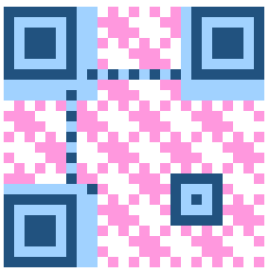

図 4.12: $(ij \bmod 3 + i + j) \bmod 2 = 0$

この図のピンクの部分を白黒反転させることになります。

早見表

これまでの作業、正直面倒ですよね。

ということでフォーマット情報から誤り訂正符号、マスクパターンが一瞬で分かる表を作りました。

表 4.3: 早見表

フォーマット情報	誤り訂正符号	マスクパターン
111011111000100	L	$(i+j) \bmod 2 = 0$
111001011110011	L	$i \bmod 2 = 0$
111110110101010	L	$j \bmod 3 = 0$
111100010011101	L	$(i+j) \bmod 3 = 0$
110011000101111	L	$((i \div 2) + (j \div 3)) \bmod 2 = 0$
110001100011000	L	$ij \bmod 2 + ij \bmod 3 = 0$
110110001000001	L	$(ij \bmod 3 + ij) \bmod 2 = 0$
110100101110110	L	$(ij \bmod 3 + i + j) \bmod 2 = 0$
101010000010010	M	$(i+j) \bmod 2 = 0$
101000100100101	M	$i \bmod 2 = 0$
101111001111100	M	$j \bmod 3 = 0$
101101101001011	M	$(i+j) \bmod 3 = 0$
100010111111001	M	$((i \div 2) + (j \div 3)) \bmod 2 = 0$
100000011001110	M	$ij \bmod 2 + ij \bmod 3 = 0$
100111110010111	M	$(ij \bmod 3 + ij) \bmod 2 = 0$
100101010100000	M	$(ij \bmod 3 + i + j) \bmod 2 = 0$
011010101011111	Q	$(i+j) \bmod 2 = 0$
011000001101000	Q	$i \bmod 2 = 0$
011111100110001	Q	$j \bmod 3 = 0$
011101000000110	Q	$(i+j) \bmod 3 = 0$
010010010110100	Q	$((i \div 2) + (j \div 3)) \bmod 2 = 0$
010000110000011	Q	$ij \bmod 2 + ij \bmod 3 = 0$
010111011011010	Q	$(ij \bmod 3 + ij) \bmod 2 = 0$
010101111101101	Q	$(ij \bmod 3 + i + j) \bmod 2 = 0$
001011010001001	H	$(i+j) \bmod 2 = 0$
001001110111110	H	$i \bmod 2 = 0$
001110011100111	H	$j \bmod 3 = 0$
001100111010000	H	$(i+j) \bmod 3 = 0$
000011101100010	H	$((i \div 2) + (j \div 3)) \bmod 2 = 0$
000001001010101	H	$ij \bmod 2 + ij \bmod 3 = 0$
000110100001100	H	$(ij \bmod 3 + ij) \bmod 2 = 0$
000100000111011	H	$(ij \bmod 3 + i + j) \bmod 2 = 0$

ご参考までに。

データ領域 (無色)

ところで、上の例でのフォーマット情報領域は `111110110101010` でしたね。これを上の早見表に照し合わせると、マスクパターンは「$j \bmod 3 = 0$」であることが分かります。ルールに従ってセルを反転させると以下のようになります。

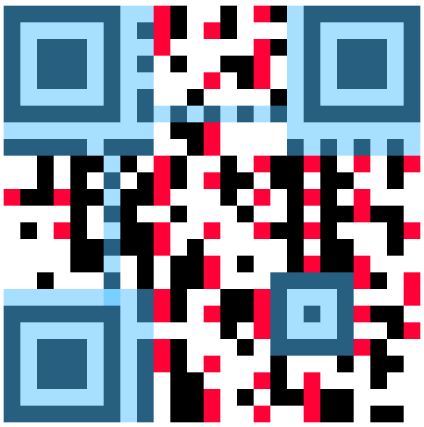

図 4.13: マスク処理後

さらに、これを色分けすると以下のとおりです。

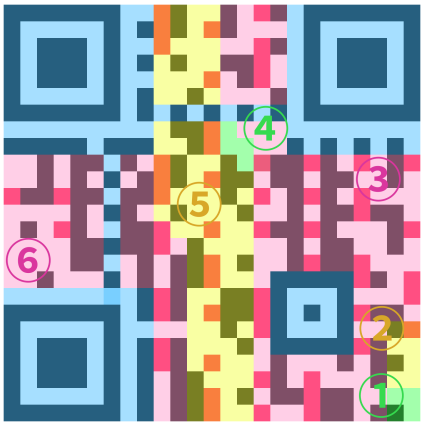

図 4.14: データ領域

この章 (節？) では、これらの領域を解読していきます。

はじめに、このデータ領域の読み進め方ですが、以下のように右下から 2 列ずつ、ギザギザに読んでいきます。この場合も白は `0`、黒 (と赤) は `1` に変換します。

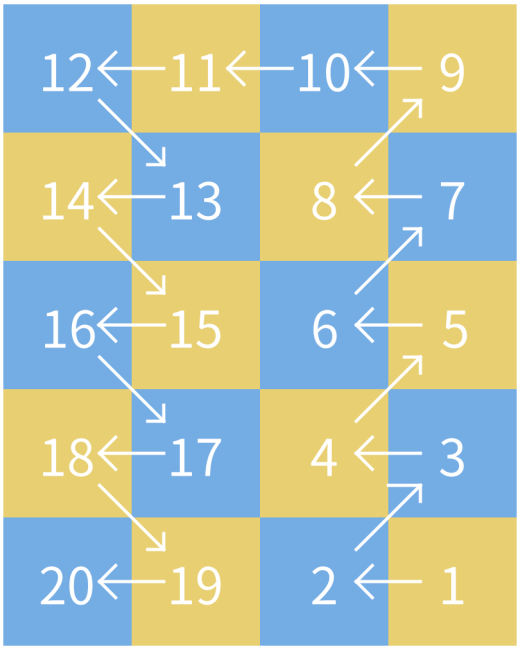

図 4.15: データ領域の読み方

例えば図 4.14 の右下の黄緑の部分は「0100」となります。それではこのことを頭に入れつつ、データ領域を解読していきましょう。

①モード指示子

QR コードでは扱える文字数を少しでも増やすために扱うデータの種類 (数字、英語、日本語など) に応じて符号化モードと呼ばれるものが存在します。このモード指示子はその QR コードで使われる符号化モードを示すためのものです。

表 4.4: モード指示子

モード指示子	符号化モード	扱える文字	データ密度
0001	数字モード	数字 (0~9)	10 セル/3 文字
0010	英数字モード	数字 (0~9)、大文字アルファベット (A-Z)、空白、$、%、*、+、-、.、/、:	11 セル/2 文字
0100	8 ビットモード	全てのデータ	8 セル/1 バイト
1000	漢字モード	漢字、ひらがな、カタカナ、全角英数字など	13 セル/1 文字

②文字数指示子

後述するデータコード語の文字数 (8 ビットモードの場合はバイト数) を表すものです。この領域の大きさは QR コードの大きさや符号化モードによって変わり、その対応は以下の通りです。(QR コードの大きさは「バージョン」というもので表現され、例の場合はバージョン 2 です。)

表 4.5: 文字数指示子の大きさ (セル数)

バージョン	数字モード	英数字モード	8 ビットモード	漢字モード
1~9	10	9	8	8
10~26	12	11	16	10
27~40	14	13	16	12

上の例ではバージョン 2、8 ビットモードなので 8 セル読み進めると、`00010011` で、10 進数に変換すると 19 になります。なので、下のデータコード語は 8 セル× 19 バイトで 152 セルあることが分かります。

③データコード語

やっと本題です。ここから QR コードの中身 (「https://・・・」といった情報) を読んでいきます。符号化モードによって読み方が大きく異なるので、それぞれ見ていきましょう。

■数字モード　まず 10 セル読み進めて、10 進数に直します。例えば `0001111011` の場合「123」です。このように、上で得た文字数と照し合わせながら、3 文字ずつ読み進めていきます。最後に 1 文字余ったら 4 セル、2 文字余ったら 7 セル読み進めて、10 進数に直します。これを順番に繋ぎ合わせると、数字が出来上がります。

■英数字モード　まず 11 セル読み進めて、10 進数に直します。そして、これを 45 で割って、商、余りをそれぞれ以下の表を見て英数字に直します。

表 4.6: 英数字モード

コード	文字
0	0
1	1
⋮	⋮
9	9
10	A
⋮	⋮
35	Z
36	空白
37	$
38	%
39	*
40	+
41	-
42	.
43	/
44	:

例えば `10000011000` の場合 10 進数に直すと 1048 で、45 で割ると商が 23、余りが 13 だから「ND」となります。このように 2 文字ずつ読み進め、1 文字余ったら 6 セル読んで 10 進数に直し、上の表で英数字に直します。これを繋ぎ合わせると、英数字の文字列が出来上がります。

■8 ビットモード　このモードは様々な種類のデータを表現できるのですが、英数字の文字列として読むときは、8 セルずつ読み進めて、10 進数に直し、以下の表 (Ascii コード表といいます。) を使って文字に変換します。

表 4.7: アスキーコード表

コード	文字
32	空白
33	!
34	"
35	#
36	$
37	%
38	&
39	’
40	(
41	)
42	*
43	+
44	,
45	-
46	.
47	/
48	0
49	1
⋮	⋮
57	9
58	:
59	;
60	<
61	=
62	>
63	?
64	@
65	A
⋮	⋮
90	Z
91	[
92	\
93	]
94	^
95	_
96	‘
97	a
⋮	⋮
122	Z
123	{
124	|
125	}
126	~

これを繋ぎ合わせると、文字列ができます。コードがこの表に無い場合、諦めて別の方法で読む必要があります。(ここでは解説しません。)

■漢字モード　まず 13 セル読み進め、**16 進数** (123456789abcdef で表す) に直します。例えば 0000100100000 なら 120 になります。そしてこれを c0 で割った商について、00〜1e なら 81 を、1f〜2a なら c1 を足します。(例の場合は 120÷c0＝01 だから 01＋81＝82) また、余りについては 40 を足します。(例だと 120%c0＝60 だから 60＋40＝a0) そしてこれらをつなげたもの (82a0) をこちらの表を使って文字に変換します。(例だと「あ」) このように 13 セルずつ読んでいき、一文字ずつ繋げると文字列が出来上がります。

■図にすると　こんな感じ

図 4.16: QR コードを読む

ということでこの QR コードの中身は「https://www.npca.jp」(NPCA の公式サイト) であることが分かりました。ここからはおまけです。

④終端パターン・埋め草ビット

白を最低 4 つ、残りのセル数が 8 で割り切れるようになるまで続けます。入っているデータがギリギリの場合は省略できます。ちなみに「埋め草」とは元々穴を埋めるために使った草や雑木のことで、そこから転じて新聞や雑誌ではスペースを埋めるための記事のことを「埋め草」と言ったりします。ここでは空いたスペースを埋めるデータ列のことを言います。(話題不足)

⑤埋め草コード語

「11101100」、「00010001」が交互に並んでいるものです。なぜ「00000000」ではダメかというと白が極端に多くなるからです。ちなみに「埋め草」とは元々穴を埋めるために使った草や雑木のことで、そこから転じて新聞や雑誌ではスペースを埋めるための記事のことを「埋め草」と言ったりします。ここでは空いたスペースを埋めるデータ列のことを言います。(2 回目)(ホントに話題がない)

⑥誤り訂正符号

QR コードに汚れなどがあったり、真ん中に変なロゴが入っていたり、単純に読み取りミスをしたりしたときなど、それを確認・修正できないと読み取り精度が格段に下がってしまいます。そこで、誤り訂正という読み取り時のミスを訂正するための技術があり、そのためのデータがここに入っています。QR コードではリード・ソロモン符号という技術が使われているのですが、僕には何のことかさっぱり分かりません。数学などに詳しいという方は

調べてみてください。~~そして僕に教えてください。~~

4.6 まとめ

これで長い長い解説は終わりです。とはいえ、読み方自体の説明はごく少量でしたね。~~ほんまか？~~
ということで、これからは QR コードを**最速で**読む方法を解説します。

1. まずフォーマット情報の 3〜5 番目のセルを読みます。
2. 「早見表」で対応する 3〜5 番目の数字を見つけます (これがマスクパターンになります)
3. (以下、色の反転を意識しつつ、右下から) モード指示子を読みます。(「①モード指示子」参照)
4. 今読んでいる QR コードのバージョン、符号化モードを踏まえて、文字数指示子の長さを確認して、文字数を把握します。(「②文字数指示子」参照)
5. データコード語を 4 で得た文字数分読みます。(「③データコード語」参照)

ということで、暇なときに試してみてはいかがでしょうか。時間が溶けますよ。

4.7 おわりに

気付けばなかなかの大作になってしまいましたが、これでやっとおしまいです。軽い気持ちでページを開けて、後悔した方もいるのではないのでしょうか。ちなみに、この記事中に大量に存在する画像は「Krita」というソフトで作成しました。興味のある方は調べてみてください。非常にドット絵を作るのに向いています。(個人の感想)
最後に、この記事を最後まで読んでくださった奇特な方、本当にありがとうございました。

4.8 参考

- https://ja.wikipedia.org/wiki/QR%E3%82%B3%E3%83%BC%E3%83%89
- https://www.thonky.com/qr-code-tutorial
- https://qiita.com/Kta-M/items/6f7049a1e78b1fe7e883