プロジェクト実習1 D02回
誤り訂正符号，RS符号

山本担当分

通信ネットワーク基礎実験（テーマC山本）の教材資料です．
1限分と2限分の最後の2箇所に課題があります．Teamsの課題で提出してください．

ラテン方格（数独パズルの基礎）

2

各記号が各行および各列に1回だけ現れるように
並べたもの

1	2	3
2	3	1
3	1	2

これは数独パズルのもとになっているラテン方格というものです．

数字の並び方に「各記号が各行および各列に1回だけ現れる」という規則が設定されています．数独パズルはこれをさらに 3x3 に配置して数字も 1 から 9 に増やして複雑化し，パズルとしたものです．

ラテン方格パズル

3

以下のラテン方格を完成させよ

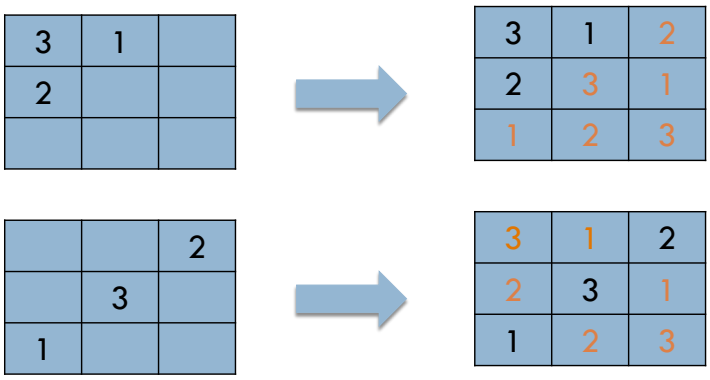
3	1	
2		

3	1	2
2	3	1
1	2	3

解き方は,例えば左下は列(縦)の上に 3,2 があるので 1 しかない,真ん中は上に 1 があるので 1 はダメ,左に 2 があるので 2 もダメなので 3,というように空欄を埋めていくことができます.

冗長性

4



いくつか消えても復元できる

ラテン方格の性質として元の表からいろいろな消え方で空欄を作ってももとに復元できるということです.

ラテン方格は数学の一分野で,与えられた大きさのラテン方格について何個の数字を残せば一意に復元できるか,というようなことが研究されています.

いくつか消えても復元できる,という性質は,通信中のノイズの修復に利用することができます.

ディジタル通信

5

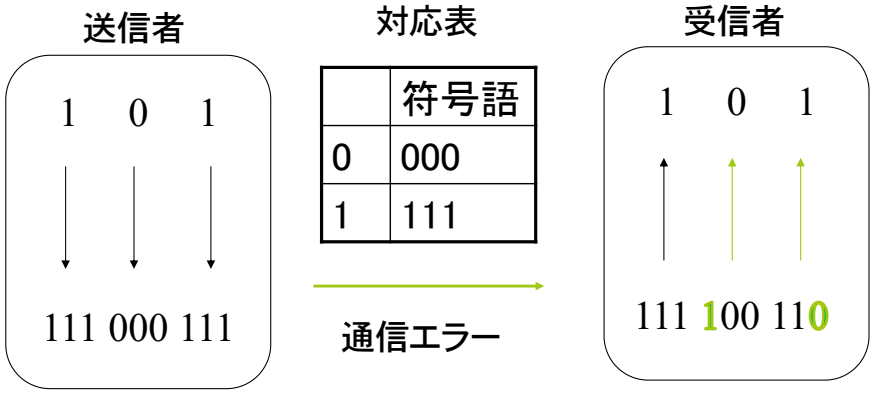
- すべての情報は 2 種類の記号（通常は 0 と 1で表す）の連続で表される（例：A→0100 0001）
  - インターネット
    - 0 と 1 を連続して送信し，受信者がそれを受信する
    - いくつかが間違って受信される
  - CD-ROM
    - 0 と 1 が連続してディスクに書き込まれているものを読み取る
    - いくつかが間違って読み取られる
- 両者ともに符号理論の技術が使われている。

デジタル通信では全ての情報は 0, 1 の系列で表されます.インターネットなどで使われる通信とCD-ROM の読み出しなどの記憶媒体の利用は別の分野ですが,いくつかの文字が誤ったときにそれを訂正する技術が必要,という点で同じ技術が使われます.

この誤りがあったときにそれを訂正するために符号理論の技術が使われます.

単純な符号（3回繰り返し符号）

6



通信エラーは,通信の途中でビットが反転する（0が1に化ける,1が0に化ける）形で現れるとします.

通信エラーを訂正するための 3 回繰り返し符号とよばれる簡単な符号を説明します.

0 を送りたいときには 0 を 3 個,1を送りたい時には 1 を 3 個送ります.

こうすることで 3 個のうち 1 個までなら通信途中でエラーにあっても0 か 1 の多い方に復元することで正しく復元されます.

エラーには強くなったのですが,送りたい情報の 3 倍のデータを送らなければいけないので通信効率が悪いというデメリットがあります.

符号化率

7

- 符号化率 $R = \dfrac{\text{送りたい情報のビット数}}{\text{符号化後のデータのビット数}}$
- 3回繰り返し符号の符号化率は 1/3
- 符号化率が高いほうが通信効率がよい

情報を符号語に変換することを符号化といいます.符号化後のデータ,すなわち符号語のビット数に対する送りたい元の情報のビット数の割合を符号化率といい,R で表します.

符号化率は高いことが望ましいのですが,先ほどの 3 回繰り返し符号の符号化率は 1/3 で,送られた記号の個数の 1/3 しか情報が入っていない,ということになります.

3回繰り返し符号，受信ベクトル

8

- 全体空間
  - 3 ビットのあらゆるパターン
  - {000,001,010,011,100,101,110,111}
- 符号語
  - {000,111}
- 送信される可能性があるのは符号語のみ．受信ベクトルは全体空間のどれか
  - 可能性が低くてもノイズすべてを想定する

誤り訂正符号によってどれだけ誤り確率を低下させられるのかを具体的に説明します．

計算が簡単な 3 回繰り返し符号にもどります．3回繰り返し符号をあらためて定義するとこのようになります．

ポイントは符号語が全体空間の部分集合であること,送信される可能性があるのは符号語のみですが,受信される可能性があるのは全体空間のすべてのベクトルであることです．

確率が低くても全てのノイズを想定するからです．

送信語

9

全体空間

000, 100
001, 101
010, 110
011, 111

- 送信される可能性があるのは符号語のみ
- 受信される可能性があるのは全体空間の全てのベクトル

これは全体空間にあたる全 8 個のベクトルを図示したものです.

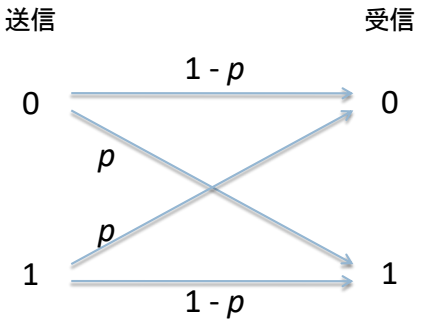
符号語の集合はこの中の部分空間で図では緑色で示されています.

送信される可能性があるのは全体空間の一部である一方,受信される可能性があるのは全体空間の任意の要素です.

2元対称通信路

10

- 最も一般的に仮定される
  0 が 1 に化ける確率も 1 が 0 に化ける確率も p



誤り確率を求めるには通信路の誤りの起こり方をモデル化しなければなりません.

これは通信路の誤りモデルとして最も単純な2元対称通信路です.

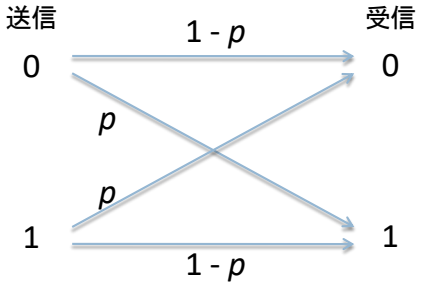
「対称」とは0が1に化ける確率と,逆に1が0に化ける確率が等しいことからきています.

誤りが起こる(化ける)確率を p とします.一般的に誤りが起こらない確率 1-p は誤りが起こる確率 p よりもずっと大きな値で 1 に近いと考えて下さい.

2元対称通信路の例

11

- 000 を送って 000 に化ける確率 : $(1-p) * (1-p) * (1-p)$
- 000 を送って 100 に化ける確率 : $p * (1-p) * (1-p)$
- 000 を送って 110 に化ける確率 : $p * p * (1-p)$
- 000 を送って 111 に化ける確率 : $p * p * p$



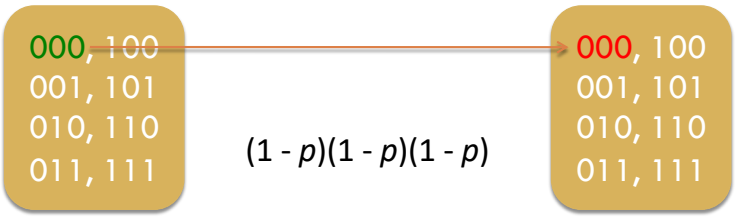
2元対称通信路を仮定すると,誤りのパターンに応じて発生する確率を計算することができます.

上の4つの誤りパターンでは誤りが発生する順番でpを,発生しない順番で1-pを掛けることでそのパターンの文字化けが起こる確率を求めることができます.

ここで 1-p が p より大きいことから,文字化けの数が少ないほうが発生確率が高いことがわかります.

誤りがない場合

000 を送った場合

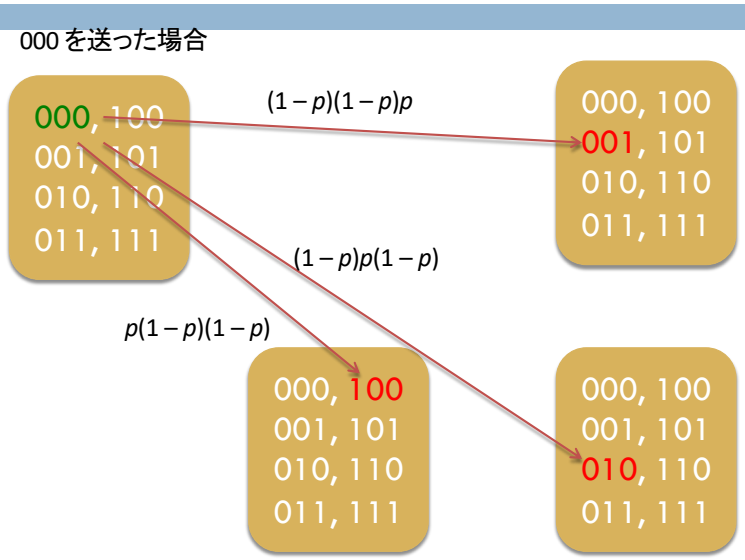


文字化けのパターンは 2 の 3 乗の 8 種類ありますが,全ての発生確率を求めることができます.

全ての文字化けパターンについて,文字化けの個数ごとに発生確率を説明します.

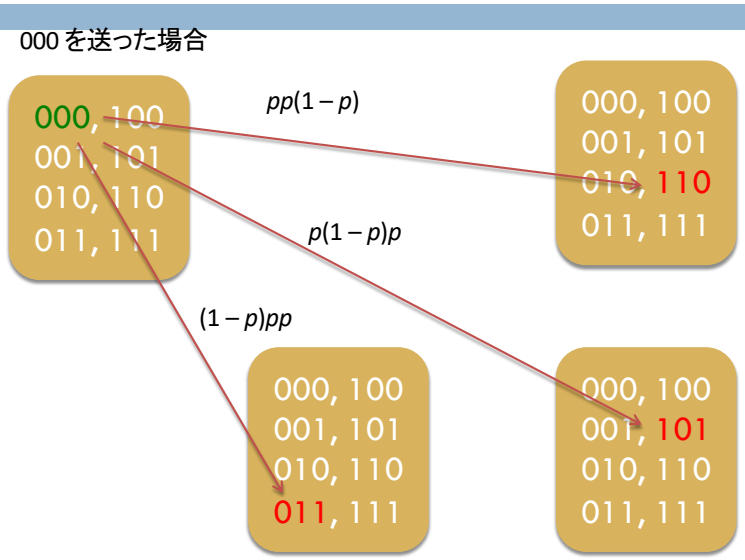
1個も誤りが発生しない場合は全ての順番で1-pを掛けるので 1-p の3乗になります.

1 文字誤り



誤りの数が 1 箇所のケースはこの3通りです.
掛ける順番は違いますが,結果的にどれもp掛ける(1-p)の2乗になります.

2 文字誤り



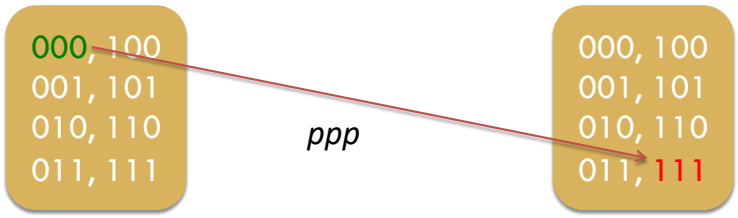
誤りの個数が 2 のケースはこの 3 つです.

これも掛け算の順番が違うだけで,発生する確率はいずれも p の2乗掛ける 1-p になります.

3文字誤り

15

000 を送った場合



3 ビット空間上の全てのベクトルが 受信される可能性があるが，それぞれ確率が異なる

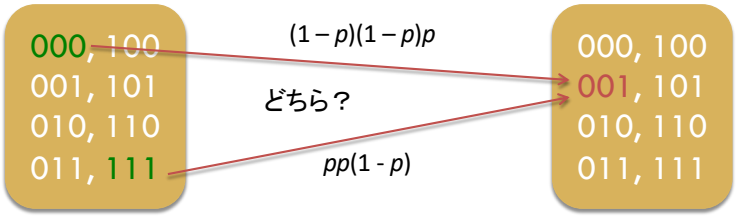
最後に,最も確率は低いですが 3 文字とも化けることもあり得ます.

確率は p の 3 乗になります.

このように 3 ビット空間上のベクトル(3 ビットのビットパターン)全てが受信される可能性があり,それぞれの確率を算できることがわかります.

受信語だけ分っている場合

16



- 001 を受信，しかし 001 は送られないはず
  - 000 が 001 に化けた: 確率 $(1-p)(1-p)p$
  - 111 が 001 に化けた: 確率 $pp(1-p)$
- 確率の高いものに復号する
  - 最尤復号

復号とは何を行うことかを改めて説明すると,全空間の要素である受信語を入力として,部分空間である符号語の集合から送信されたであろう符号語を選ぶことです.

どの符号語を選ぶかの判断基準として誤りが起こる確率を使います.

上の例は 001 を受信したケースで,001は符号語でないので誤りがあったことがわかります.

送信される可能性があるのは符号語なのでそれぞれの符号語について,受信した 001 に「化ける」確率を調べます.

この確率が高いものに復号することを最尤（ゆう）復号といいます.

この例では p が 1-p より小さいことに注意すると 000 のほうが確率が高いので000 に復号します.

ハミング距離

17

- 異なるビット位置の数
  - $d(000, 001) = 1$
  - $d(111, 001) = 2$
- ハミング距離が近いもののほうが確率が高い.
- ハミング距離が近いものに復号する

どの符号語に復号すると最尤復号ができるかは,ハミング距離を考えるとわかりやすくなります.

ハミング距離は 2 つのベクトル間に定義されます.2つのベクトルを要素ごとに比較し,値の異なる位置の数をハミング距離といいます.

000 と 001 のハミング距離は 1, 111 と 001 のハミング距離は 2 です.要するに何文字違うかを表しています.

これまでに説明したように,ベクトルの文字化けは文字化け前と後のベクトルのハミング距離が小さいものほど確率が高くなります.

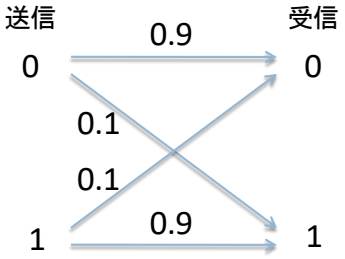
前の例で001に距離が近い符号語000に復号したことからもわかるようにハミング距離が近い符号語に復号すれば最尤復号を行うことができます.

3回繰り返し符号では単純に多数決をとって復号すれば最尤復号を行うことができます.

例 p = 0.1

18

- 0 を符号化せずに送る
  - 誤り確率は 0.1



- 3回繰り返し符号で送る

送信語 000

受信語	確率	
000	0.729	正
100	0.081	
010	0.081	
001	0.081	
110	0.009	0.028
011	0.009	誤
101	0.009	
111	0.001	

0.972

これは具体例として p が 0.1 のとき,3回繰り返し符号によってどれだけ誤り確率を下げることができるかを示したものです.

符号化せずに1文字を送った場合の誤り確率は当然 0.1 です.

3回繰り返し符号を使った場合,全ての可能性のある受信語についてその誤りが起こる確率をいままで説明した方法で計算したものが右の表です.

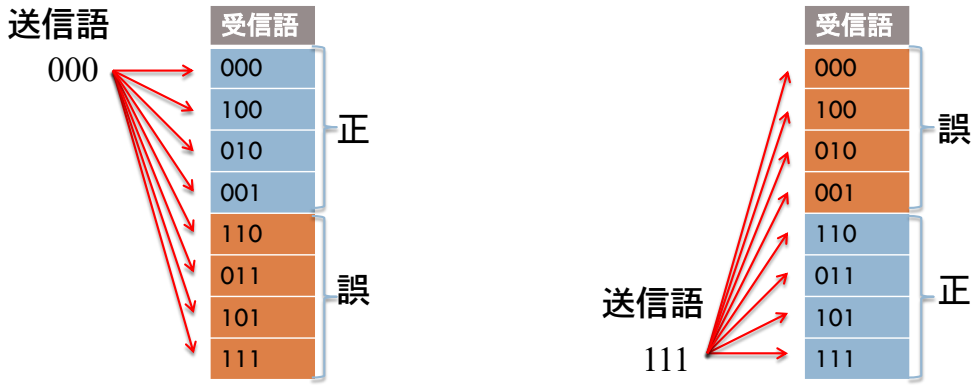
多数決で復号するので上の4つの受信語が正しく復号され,下の 4 つの符号語は誤って復号されます.

誤って復号されるような誤りが発生する確率の和 0.028 が 3 回繰り返し符号を使った場合の誤り確率になります.

符号化しなかったときと比較して3分の1以下に低減されていいることがわかります.

誤り訂正能力

19



1ビットまでの誤りなら訂正できる

3回繰り返し符号は1ビットまでの誤りであれば必ず訂正できます．
それは0を送った場合も1を送った場合も同じです．

最小距離

20

全体空間

000, 100
001, 101
010, 110
011, 111

- 符号語と符号語のハミング距離の最小値を最小距離という
- dで表されることが多い
- 3回繰り返し符号では $d = 3$

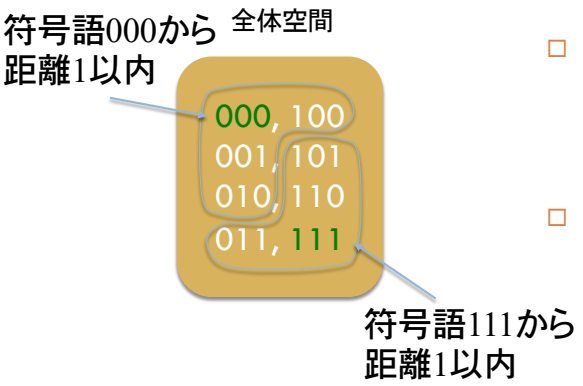
これは全体空間にあたる全 8 個のベクトルを図示したものです.

符号語の集合はこの中の部分空間で図では緑色で示されています.

送信される可能性があるのは全体空間の一部である一方,受信される可能性があるのは全体空間の任意の要素です.

誤り訂正能力

21



- $t = \left\lfloor \frac{d-1}{2} \right\rfloor$ 以下の誤りは必ず訂正できる（⌊　⌋は切り捨て記号）
- 3回繰り返し符号の場合は $t = 1$

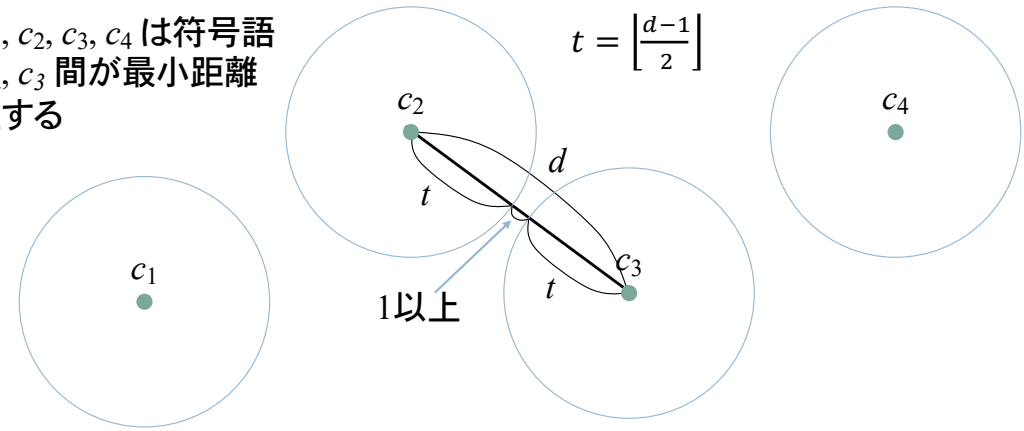
最小距離がdならd-1割る2を切り捨てた値までの誤りなら必ず訂正できます.
これを誤り訂正能力といい,tで示されます.
3回繰り返し符号の誤り訂正能力は1です.符号語を中心に
ハミング距離が1以下の範囲を符号語の勢力圏と考えると,
2つある符号語の勢力圏が重ならないため,復号すべき符号語を一位に選べるからです.

最小距離と誤り訂正能力

22

c_1, c_2, c_3, c_4 は符号語
c_2, c_3 間が最小距離
とする

$t = \left\lfloor \frac{d-1}{2} \right\rfloor$



t は最小距離 d から隙間として1を引いたものを2で割ったもの（以下）なので
符号語を中心としてハミング距離がt以下の勢力圏を考えると，
どの符号語の勢力圏も互いに重なることはありません．
このため,t 個以下の誤りであればその符号語の勢力圏から出ることはないので
最も近い符号語に復号することで正しく復号されることが保証されます．

最小距離 d が大きいほど誤り訂正能力は高くなるので符号にとって最小距離は
重要なパラメータです．

符号のパラメータ

23

- $[n, k, d]$ 符号：符号語長 n 情報記号数 k 最小距離 d の符号
- 3回繰り返し符号は $n = 3, k = 1, d = 3$ の $[3, 1, 3]$ 符号
- 符号化率 $R = \frac{k}{n}$ が大きく，最小距離 d が大きい符号が望ましい

符号は3つの重要なパラメータである符号語長n, 情報記号数k, 最小距離 d を表記して書かれることが多くありうます.

同じ送信記号数であればより多くの情報を送りたいので符号化率 k/n は大きく,かつ訂正能力を高めたいので

d も大きい符号が望ましい符号として探す研究がされてきました.

Reed Solomon 符号

- 最小距離が限界値を達成している最大距離分離符号
- 最小距離 $d = n - k + 1$ であることが証明できる
- 地上波デジタル放送で符号語長200程度のRS符号を使われている
- 0,1だけを使う 2 元符号ではない多元符号
- QRコードでは 2^8 元RS符号が使われる
- 8ビットを一つの記号として処理するので隣接するビット誤りに強い
  - 汚れてもデコードできる QR コードに適している

QRコード

- 格納できる情報量（バージョン），誤り訂正能力を選べる
- 今回はバージョン2，誤り訂正能力Mのコードを使用する
  - $[n, k, d] = [44, 28, 17], t = \left\lfloor \frac{17-1}{2} \right\rfloor = 8$
  - 28文字の英数字テキストを格納でき，8 シンボルまでの誤りを訂正できる

多項式の計算

- 多項式の掛け算や割り算を使う
- RS符号では係数が有限種類の世界の多項式を使う

有理数係数多項式の乗算

27

- x^2+2x-1 の係数
  - 2次: 1, 1次: 2, 0次(定数項): -1
- 有理数係数の多項式の計算ならなじみがある
  - 有理数:整数 m とゼロでない整数 n で m/n として表せる数

式の乗算: $(x^2+2x-1)(x+3)=$

	x^2	$+2x$	-1
$\times$		x	$+3$
	$3x^2$	$+6x$	-3
x^3	$+2x^2$	$-x$	
x^3	$+5x^2$	$-5x$	-3

係数は乗算と加算しかしないので
式どうしの乗算は有理数係数の
多項式になる

乗算について閉じている

有理数係数多項式の除算

28

係数は除算,減算を使う
有理数は割り算,引き算しても有理数

						$\frac{1}{1}x$	$\frac{6}{1}+6$	商
x^2	$-x$	$+1$	)	x^3	$+5x^2$	$+3x$	-1	
				x^3	$-x^2$	$+x$		除数x 1/1
					$6x^2$	$+2x$	-1	
					$6x^2$	$-6x$	$+6$	除数x 6/1
						$8x$	-7	剰余

除算についても閉じている

体（たい）

29

- おおざっぱに言うと加減乗除四則演算に対して閉じた集合
- 集合の要素が元（前の例では一つ一つの有理数）
- 減算は加法逆元の和，除算は乗法逆元の積
  - 3-2 は 3+(-2)，　2/3 は $2 * 3^{-1}$
- 加法単位元（0）：足しても変わらない元
- 乗法単位元（1）：掛けても変わらない元
- 元 a の加法逆元：aに足すと0になる数（-aと書く）
- 0以外の元 a の乗法逆元：aに掛けると1になる数（a^{-1}書く）

体で重要なことは

- 四則演算可能，というより加法単位元，乗法単位元があり，全ての元に加法逆元，0以外の全ての元に乗法逆元があることが重要
- 有理数体の場合は
  - 加法単位元は0
  - 乗法単位元は1
  - 1の加法逆元は-1, 2の加法逆元は -2,…
  - 1の乗法逆元は1/1, 2の乗法逆元は 1/2,…
- 有理数では存在することが簡単にわかる（分数の形だから）

有限体(素体)

- 要素数(位数)が有限の場合は体にするのは難しい(有限体)
- 要素数が素数の体がまず入門編(素体)
- 要素数が3であれば普通に計算して3で割った余りでいい

加算

＋	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

加法単位元 0

加法逆元 $-0 = 0, -1 = 2, -2 = 1$

$1 - 2$ は $1 + (-2) = 1 + 1 = 2$

乗算

×	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

乗法単位元 1

乗法逆元 $1^{-1} = 1, 2^{-1} = 2$

$1 \div 2$ は $1 \times 2^{-1} = 1 \times 2 = 2$

ガロア体

- 有限体の要素数（位数）は素数のべき乗
- p が素数の時有限体 $GF(p^n)$ をガロア体という
- 素体のように普通の加法乗法の結果を位数で割った余り，ではだめ

4で割った余りで体は作れない

×	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

2 に何を掛けても 1 にならない

2 の逆元が存在しない

ガロア体の作り方

- GF(p^n) の元は n-1 次以下の多項式（係数は位数pの素体）
  - そのような多項式は p^n 個ある
- どうやって加算，乗算とその単位元，逆元を定義するか？

例：GF(2^3)

35

ベクトル表現

0 0 0	0
0 0 1	1
0 1 0	x
0 1 1	x+1
1 0 0	x^2
1 0 1	x^2 +1
1 1 0	x^2+x
1 1 1	x^2+x+1

- 元は 2 次以下の（係数が{0,1}の素体の）多項式全て

この 8 個が GF(2^3)の元

3ビットの全パターン（2次の係数,1次の係数,0次の係数）

この説明は非常に重要です

GF(2^3)の加法

0	1	x	x+1	x^2	x^2+1	x^2+x	x^2+x+1

- 係数は位数2の素体の加算（2になると0にもどる）
- 例：(x+1) + (x^2+1) = (x^2+x)
- 加法単位元は多項式 0 （どの多項式に足しても変わらない）
- どの多項式のペアの和も 8 要素のどれかになる（閉じている）
- 自分自身を足すと0になるので自分自身が加法逆元
- 加算はベクトル表現のビットごとのXOR（排他的論理和）で計算
  - 0+0 = 0, 0+1 = 1, 1+0 = 1, 1+1 = 0

GF(2^3)の加算

+	0	1	x	x+1	x^2	x^2+1	x^2+x	x^2+x+1
0	0	1	x	x+1	x^2	x^2+1	x^2+x	x^2+x+1
1	1	0	x+1	x	x^2+1	x^2	x^2+x+1	x^2+x
x	x	x+1	0	1	x^2+x	x^2+x+1	x^2	x^2+1
x+1	x+1	x	1	0	x^2+x+1	x^2+x	x^2+1	x^2
x^2	x^2	x^2+1	x^2+x	x^2+x+1	0	1	x	x+1
x^2+1	x^2+1	x^2	x^2+x+1	x^2+x	1	0	x+1	x
x^2+x	x^2+x	x^2+x+1	x^2	x^2+1	x	x+1	0	1
x^2+x+1	x^2+x+1	x^2+x	x^2+1	x^2	x+1	x	1	0

加算はベクトル表現の排他的論理和

GF(2^3)の乗算

×	0	1	x	x+1	x^2	x^2+1	x^2+x	x^2+x+1
0	0	0	0	0	0	0	0	0
1	0	1	x	x+1	x^2	x^2+1	x^2+x	x^2+x+1
x	0	x	x^2	x^2+x	x+1	1	x^2+x+1	x^2+1
x+1	0	x+1	x^2+x	x^2+1	x^2+x+1	x^2	1	x
x^2	0	x^2	x+1	x^2+x+1	x^2+x	x	x^2+1	1
x^2+1	0	x^2+1	1	x^2	x	x^2+x+1	x+1	x^2+x
x^2+x	0	x^2+x	x^2+x+1	1	x^2+1	x+1	x	x^2
x^2+x+1	0	x^2+x+1	x^2+1	x	1	x^2+x	x^2	x+1

（多項式の乗算をして原始多項式 $p(x) = x^3 + x + 1$ で割った余り）

GF(2^3)の乗算（べき表現）

- $p(x) = x^3 + x + 1$ で割った余り（2次式）が元全てである世界
- α を多項式 x とする
- $\alpha^0 = 1, \alpha^1 = x, \alpha^2 = x^2$
- $\alpha^3 = x^3$ を $p(x)$ で割った余り $= x + 1$
- $\alpha^4 = \alpha^3 \times \alpha = x^2 + x$
- $\alpha^5 = \alpha^4 \times \alpha = x^3 + x^2$ を $p(x)$ で割った余り $= x^2 + x + 1$
- $\alpha^6 = \alpha^5 \times \alpha = x^3 + x^2 + x$ を $p(x)$ で割った余り $= x^2 + 1$
- $\alpha^7 = 1$（α^0に戻る）
  - α^0 の逆元は α^0, $1 \leq i \leq 6$の α^i の逆元は α^{7-i}

GF(2^3)の原始限 α

40

- $\alpha^0 = 1, \alpha^1 = x, \alpha^2 = x^2, \alpha^3 = x + 1$
- $\alpha^4 = x^2 + x, \alpha^5 = x^2 + x + 1, \alpha^6 = x^2 + 1$
- 0 以外の元は α^0 から α^6 で表せる（べき表現）
- α を原始元という
- 多項式 0 はαのべき乗ではできない
  - 多項式 0 の元のみべき表現できない
- 多項式0の元以外はベクトル表現とべき表現に一対一対応

べき→ベクトル

α^0	α^1	α^2	α^3	α^4	α^5	α^6
001	010	100	011	110	111	101
1	x	x2	x+1	x^2+x	x^2+x+1	x^2+1

べき表現によるGF(2^3)の乗算

41

×	(0)	α^0 (1)	α^1 (x)	α^2 (x^2)	α^3 (x+1)	α^4 (x^2+x)	α^5 (x^2+x+1)	α^6 (x^2+1)
多項式0	多項式0	多項式0	多項式0	多項式0	多項式0	多項式0	多項式0	多項式0
α^0	多項式0	α^0	α^1	α^2	α^3	α^4	α^5	α^6
α^1	多項式0	α^1	α^2	α^3	α^4	α^5	α^6	α^0
α^2	多項式0	α^2	α^3	α^4	α^5	α^6	α^0	α^1
α^3	多項式0	α^3	α^4	α^5	α^6	α^0	α^1	α^2
α^4	多項式0	α^4	α^5	α^6	α^0	α^1	α^2	α^3
α^5	多項式0	α^5	α^6	α^0	α^1	α^2	α^3	α^4
α^6	多項式0	α^6	α^0	α^1	α^2	α^3	α^4	α^5

指数の加算（mod 7）でできるので乗算はべき表現のほうが良い（多項式0は表現できない）

GF(2^3)の演算では

- 加減算はベクトル表現，乗除算はべき表現が向いている
- プログラムを書く場合，ベクトル表現，べき表現の互いの変換方法があればいい
- 配列でできる

べき→ベクトル

α^0	α^1	α^2	α^3	α^4	α^5	α^6
001	010	100	011	110	111	101

ベクトル→べき

000	001	010	011	100	101	110	111
	α^0	α^1	α^3	α^2	α^6	α^4	α^5

元をコンピュータに向いた数値で表現

べき表現の数値表現

本来の元	1	x	x^2	x+1	x^2+x	x^2+x+1	x^2+1
べき表現	α^0	α^1	α^2	α^3	α^4	α^5	α^6
数値表現	0	1	2	3	4	5	6

ベクトル表現の数値表現

本来の元	0	1	x	x+1	x^2	x^2+1	x^2+x	x^2+x+1
ベクトル表現	000	001	010	011	100	101	110	111
数値表現	0	1	2	3	4	5	6	7

数値表現で変換を配列で実装

べき→ベクトル

α^0	α^1	α^2	α^3	α^4	α^5	α^6
001	010	100	011	110	111	101

ベクトル→べき

000	001	010	011	100	101	110	111
	α^0	α^1	α^3	α^2	α^6	α^4	α^5

べき→ベクトル 配列 p2v

p2v[0]	p2v[1]	p2v[2]	p2v[3]	p2v[4]	p2v[5]	p2v[6]
1	2	4	3	6	7	5

ベクトル→べき 配列 v2p

v2p[0]	v2p[1]	v2p[2]	v2p[3]	v2p[4]	v2p[5]	v2p[6]	v2p[7]
	0	1	3	2	6	4	5

コーディング

- ベクトル→べきの変換配列 v2p とべき→ベクトルの変換配列 p2v のどちらが簡単に計算できるか
- 片方ができればそれを使って逆を計算できる
- $\alpha^0, \alpha^1, \ldots, \alpha^6$ の多項式を順に乗算で計算すれば p2v はできる
  - $\alpha^0 = 1, \alpha^1 = x, \alpha^2 = x^2, \alpha^3 = x^3 = x + 1, \alpha^4 = \alpha^3 \times \alpha = x^2 + x$
  - $\alpha^5 = \alpha^4 \times \alpha = x^3 + x^2 = x^2 + x + 1, \alpha^6 = \alpha^5 \times \alpha = x^3 + x^2 + x = x^2 + 1$
- v2p[p2v[0]]=0, v2p[p2v[1]]=1, v2p[p2v[2]]=2, … とすれば v2p を作れる

p2vの作り方

- α^0は1なのでp2v[0]=1（多項式1のベクトル表現の数値表現）
- α^1は$\alpha^0 \times \alpha^1$ つまり今の α^0の式にxを掛ける
  - 字数が一つ上がる $x \rightarrow x^2$
  - ベクトル表現では 010→100
  - x を掛けることはビット演算で左シフトに相当
  - ただし，3ビットに収まらなくなったときは 3 次式になったときである
    - $p(x) = x^3+x+1$ で割ったあまりにする
    - 4ビットでいうと 1011 と XOR
    - シフトする前に最上位ビットが1ならシフトして011とXORすればいい
    - そうでなければシフトするだけでいい

バージョン2，誤り訂正能力MのQRコード

- GF(2^8)を使う
  - p(x) = $x^8+x^4+x^3+x^2+1$ が指定されている
  - p(x) で割った余りは高々7次式→2^8通り，元は 256 個
- この元を係数とする多項式の加算減算乗算除算が必要
- サイズ 256 のべき→ベクトルの変換配列とベクトル→べきの変換配列を作る
- 多項式の係数（GF(2^8)）の加算乗除算のルーチンを作る
  - XORなので減算は加算と同じ
- 多項式の加減乗除算，剰余計算のルーチンを作る