Reed-Solomon Codes

CS6025 Data Encoding

Yizong Cheng

4-9-15

GF(2^8) in QR Symbols

- The encoding region of a QR symbol is divided into 8-bit fields to hold codewords.
- Some of the codewords are data and others are for error correction.
- The codewords are elements of the field GF(2^8) with addition as bitwise exclusive OR and multiplication defined with the irreducible polynomial $x^8 + x^4 + x^3 + x^2 + 1$ (0x11d).
  - Compare with AES which uses the irreducible 0x11b.
  - α=2, instead of 3, is the chosen primitive element in GF(2^8).

```java
public class H22{

  static final int maxSize = 200;
  static final int formatLength = 15;
  static final int irreducible = 0x11d;
  static final int fieldSize = 256;
  static final int oneLessFieldSize = fieldSize - 1;
  static final int[] generator = new int[]{ 43, 139, 206, 78, 43,
     239, 123, 206, 214, 147, 24, 99, 150, 39, 243, 163, 136 };
  static final int capacity = 26;
  static final int correctionCapacity = 17;
  int[] G = new int[correctionCapacity + 1];
  int[] codewords = new int[capacity];
  String[] rawBitmap = new String[maxSize];
  int numberOfLines = 0;
  int version = 0;
  int width = 0;
  int height = 0;
  boolean[][] matrix = null;
  boolean[] format = new boolean[formatLength];
  boolean[] dataBitStream = null;
  int dataSpace = 0;
  int[] alog = new int[fieldSize];
  int[] log2 = new int[fieldSize];
```

Making GF(2^8)

```
void makeLog2(){
   alog[0] = 1;
   for (int i = 1; i < fieldSize; i++){
     alog[i] = (alog[i - 1] << 1);
     if ((alog[i] & 0x100) != 0) alog[i] ^= irreducible;
   }
   for (int i = 1; i < fieldSize; i++) log2[alog[i]] = i;
 }

 int inverse(int a){
   return alog[oneLessFieldSize - log2[a]];
 }

 int mul(int a, int b){
   if (a == 0 || b == 0) return 0;
   return alog[(log2[a] + log2[b]) % oneLessFieldSize];
 }
```

Error Correction Level (Format Info Bits)

Table 12 — Error correction level indicators for QR Code symbols

Error Correction Level	Binary indicator
L	01
M	00
Q	11
H	10

Table 9 — Error correction characteristics for QR Code 2005

Version	Total number of codewords	Error correction level	Number of error correction codewords	Value of p	Number of error correction blocks	Error correction code per block (c, k, r)[a]
M1	5	Error detection only	2	2	1	(5,3,0)[b]
M2	10	L M	5 6	3 2	1 1	(10,5,1)[b] (10,4,2)[b]
M3	17	L M	6 8	2	1 1	(17,11,2)[b] (17,9,4)
M4	24	L M Q	8 10 14	2 0 0	1 1 1	(24,16,3)[b] (24,14,5) (24,10,7)
1	26	L M Q H	7 10 13 17	3 2 1 1	1 1 1 1	(26,19,2)[b] (26,16,4)[b] (26,13,6)[b] (26,9,8)[b]
		L	10	2	1	(44,34,4)[b]

Table A.1 — Generator polynomials for Reed-Solomon error correction codewords

Number of error correction codewords	Generator polynomials
2	$x^2 + \alpha^{25}x + \alpha$
5	$x^5 + \alpha^{113}x^4 + \alpha^{164}x^3 + \alpha^{166}x^2 + \alpha^{119}x + \alpha^{10}$
6	$x^6 + \alpha^{116}x^5 + x^4 + \alpha^{134}x^3 + \alpha^{5}x^2 + \alpha^{176}x + \alpha^{15}$
7	$x^7 + \alpha^{87}x^6 + \alpha^{229}x^5 + \alpha^{146}x^4 + \alpha^{149}x^3 + \alpha^{238}x^2 + \alpha^{102}x + \alpha^{21}$
17	$x^{17} + \alpha^{43}x^{16} + \alpha^{139}x^{15} + \alpha^{206}x^{14} + \alpha^{78}x^{13} + \alpha^{43}x^{12} + \alpha^{239}x^{11} + \alpha^{123}x^{10}$ $+ \alpha^{206}x^9 + \alpha^{214}x^8 + \alpha^{147}x^7 + \alpha^{24}x^6 + \alpha^{99}x^5 + \alpha^{150}x^4 + \alpha^{39}x^3 + \alpha^{243}x^2$ $+ \alpha^{163}x + \alpha^{136}$

The Generator Polynomial

- In Reed-Solomon, $G(x) = (x - 1)(x - \alpha)(x- \alpha^2)\ldots(x- \alpha^{16})$
- Hence, $G(\alpha^i) = 0$ for $i = 0,\ldots,16$=correctionCapacity - 1

```
void makeG(){
   G[0] = 1;
   for (int i = 0; i < correctionCapacity; i++)
     G[i + 1] = alog[generator[i]];
}
```

Polynomial Evaluation

- $P(x) = c_n x^n + c_{n-1} x^{n-1} + \ldots + c_2 x^2 + c_1 x + c_0$ can be evaluated as
- $((\ldots((c_n x + c_{n-1})x + c_{n-2})x + \ldots)x + c_1)x + c_0$

```
int evaluatePolynomial(int[] coefficients, int x){
  int len = coefficients.length;
  int sum = coefficients[0];
  for (int i = 1; i < len; i++)
    sum = mul(sum, x) ^ coefficients[i];
  return sum;
}
```

Checking $G(\alpha^i) = 0$ for $i = 0,\ldots,16$

```
void checkG(){
   for (int i = 0; i < correctionCapacity; i++)
     System.out.println(evaluatePolynomial(G, alog[i]));
 }
```

Error Correction Codewords

- The 17 error correction codewords appended to the 9 data codewords D(x) should become a polynomial C(x) that is a multiple of the generaor polynomial, G(x).
- This can be done by dividing $x^{17}D(x)$ by G(x) and collect the remainder using long division.
- C(x) is now Q(x)G(x) and we also have $C(\alpha^i) = 0$ for i = 0...16.
- If there are errors, we receive C'(x) = C(x) + E(x).
- We have 17 syndromes $C'(\alpha^i) = E(\alpha^i)$ for i = 0...16.
- Non-zero syndromes indicate errors.

Computing and Comparing Error Correction Codewords

```
 void getRemainder(){
   for (int i = 0; i < dataCapacity; i++)
    codewords[i] = nextSymbol(i * 8, 8);
   for (int i = dataCapacity; i < capacity; i++) codewords[i] = 0;
   for (int i = 0; i < dataCapacity; i++) if (codewords[i] != 0){
    for (int j = 0; j < correctionCapacity; j++){
// your code for multiplying G(j + 1) with codewords[i] and then
// subtracting it at position i + j + 1
     }
   }
   for (int i = 0; i < correctionCapacity; i++)
    System.out.println(codewords[dataCapacity + i] + " " +
      nextSymbol((dataCapacity + i) * 8, 8));
 }
```

Syndromes are $C(\alpha^i) = 0$ for i = 0,…,16

```
void readCodewords(){
   for (int i = 0; i < capacity; i++)
    codewords[i] = nextSymbol(i * 8, 8);
}

void computeSyndromes(){
   for (int i = 0; i < correctionCapacity; i++)
     System.out.println(evaluatePolynomial(codewords, alog[i]));
}
```

Homework 22: due 4-15-15

- Complete polynomial division in getRemainder() in H22.java and run it on test21.txt.

Error Correction

- We may be able to correct errors in up to 8 codewords.
- Suppose there is error ej at position pj for j = 0,...,7.
- Syndrome Si = $E(\alpha^i) = \Sigma_{j=0}^{7}$ ej $\alpha^{ipj} = \Sigma_{j=0}^{7}$ ej $(\alpha^{pj})^i$ for i = 0,...16
- We have 17 equations and 16 unknowns ej and α^{pj} .
- We can solve these equations and correct the errors.