Tópicos Avançados em Ciência da Computação I: Introdução à Teoria de Códigos para Criptografia Pós-quântica

Daniel Panario
School of Mathematics and Statistics
Carleton University

IC - Unicamp, Sala 351 do IC-3
das 13:30 as 18:30 em 16-17 de janeiro de 2020,
das 13:30 as 17:30 em 20-24 de janeiro de 2020

Introdução à Teoria de Códigos: Part II

Daniel Panario
School of Mathematics and Statistics
Carleton University

16-24 de janeiro de 2020

Conteúdo da aula

- Códigos low density parity check (LDPC)
- Grafo de Tanner
- Algoritmo de decodificação bit-flipping
- Códigos cíclicos e polinômios sobre corpos finitos

Texto: Introducing Low-Density Parity-Check Codes, Sarah J. Johnson, 2006.

Códigos atingindo a capacidade do canal

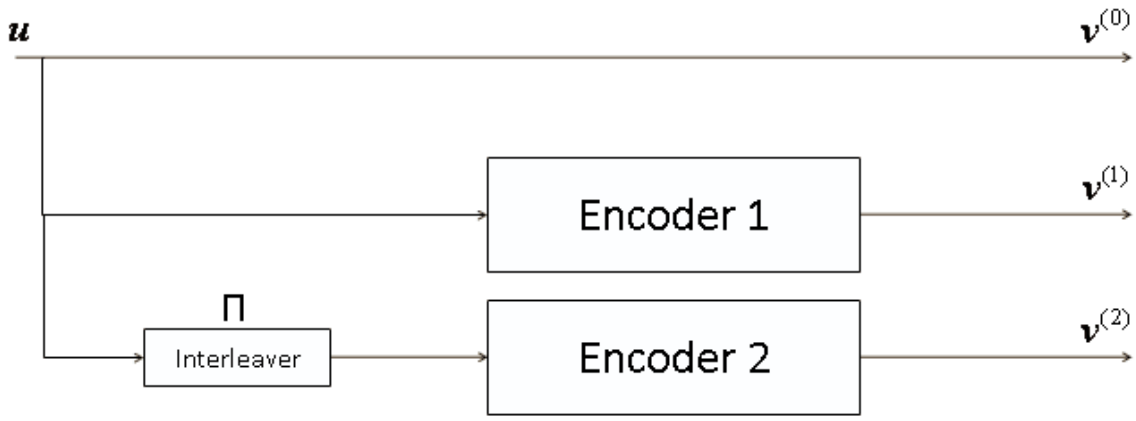
Até os anos 1990, os códigos algébricos introduzidos eram eficientes mas ficavam muito aquém da capacidade do canal estabelecida teoricamente pelo limite de Shannon (1948).

A partir dos anos 1990 surgiram várias classes de códigos com essa capacidade de canal. Os primeiros foram os códigos turbo, introduzidos por Claude Berrou (1993).

Códigos turbo são usados em comunicação 3G/4G e em comunicação de satélites no espaço profundo.

Códigos turbo

A estrutura básica de um codificador de código turbo para uma sequência de entrada, dois codificadores e um "interleaver", denotado por Π, é representada abaixo:



Normalmente, os codificadores são códigos de convolução.

Interleavers e permutações

O interleaver permuta um bloco de informação $\mathbf{x} = (x_0, \ldots, x_N)$; assim, o segundo codificador recebe como entrada uma sequência permutada do mesmo tamanho, denotada por $\widetilde{\mathbf{x}} = (x_{\Pi(0)}, \ldots, x_{\Pi(N)})$. Desta forma, o uso de interleavers leva naturalmente ao estudo de polinômios de permutação.

A função inversa Π^{-1} pode ser necessária no processo de decodificação quando um "de-interleaver" é implementado. Porém, alguns algoritmos de decodificação não requerem de-interleavers.

Um interleaver Π é auto-inverso se $\Pi = \Pi^{-1}$. Essas permutações se decompõem em ciclos de tamanho 1 ou 2, e assim são involuções, como por exemplo as funções nas caixas-S do AES.

Códigos LDPC

Códigos LDPC (low-density parity-check) foram inventados por Robert Gallager na sua tese de doutorado (1960). Esses códigos foram redescobertos por McKay e Neal (1996), e são amplamente usados na prática, atualmente.

Códigos LDPC são códigos lineares construídos usando um grafo bipartido esparso. Esses códigos também se aproximam da capacidade do canal dada pelo limite de Shannon.

Após isso, outros códigos, como o código polar, que também tem um comportamento similar em termos de capacidade do canal, foram desenvolvidos .

Códigos LDPC

Definição

Um código LDPC (low-density parity-check) é um código linear cuja matriz de paridade é esparsa.

A matriz de paridade de um código LDPC é regular se todas as suas linhas tem o mesmo peso (número de elementos não nulos), e se todas as suas colunas também têm o mesmo peso (não precisa ser o mesmo número para linhas e para colunas).

Códigos quasi-cyclic low-density parity-check (QC-LDPC) são outra categoria importante de códigos LDPC. Eles também têm uma performance muito boa e implementação simples.

Grafo de Tanner

Os códigos LDPC são códigos lineares. Eles são construídos usando um grafo bipartido esparso: o grafo de Tanner.

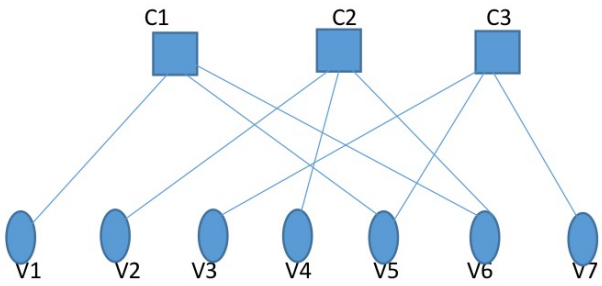
Definição

O grafo de Tanner é um grafo bipartido, cujos vértices são o conjunto de variáveis do código e o conjunto de equações de checagem ("check equations").

A matriz de adjacência do grafo de Tanner é a matriz de paridade do código.

A seguir mostramos um exemplo de grafo de Tanner e matriz de paridade associada, assim como as equações de checagem e variáveis.

Exemplo de Grafo de Tanner



Matriz H associada:

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

Equações de checagem

Considere um código de comprimento 7 com palavras código $(c_1, c_2, c_3, c_4, c_5, c_6, c_7)$, que satisfazem as seguintes três equações de checagem:

$$\begin{aligned} c_1 \oplus c_5 \oplus c_6 &= 0 \\ c_2 \oplus c_4 \oplus c_6 &= 0 \\ c_3 \oplus c_5 \oplus c_7 &= 0 \end{aligned}$$

Se escrevermos esse sistema em forma matricial, obtemos a matrix H anterior tal que as palavras-código satisfazem $Hc^T = 0$, onde operamos em $\mathbb{F}_2$ (somas são ou-exclusivos).

Daí o nome de parity-check matrix para a matriz H.

Equações de checagem (cont)

Vamos supor que recebemos $y = (1, 1, 1, 1, 0, 0, 1)$. Quando calculamos a síndrome $S(y) = Hy^T$, obtemos

$$S(y) = Hy^T = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

Isto indica que a primeira equação não é satisfeita. Portanto, isto implica que o erro está em algumas das variáveis c_1, c_5, c_6 da primeira equação.

Observe que se a síndrome é 0 e os erros transformam a palavra código enviada numa outra palavra código recebida, não temos como detectar esse erro.

Texto sobre códigos LDPC

Vamos rever esses conceitos junto com construções de códigos LDPC em detalhe usando o texto: Introducing Low-Density Parity-Check Codes, de Sarah J. Johnson, 2006. Usamos a seção 1.3, páginas 10-20:

https://pdfs.semanticscholar.org/657c/5be8e9af37dea31e357cda636e123ba03411.pdf

Veremos as construções clássicas de Gallager, de MacKay-Neal e "repeat-accumulate", e a codificação desses códigos usando versões sistemáticas.

Algoritmo de Decodificação Bit-flipping

Algoritmos de decodificação iterativa

Os algoritmos de decodificação de códigos LDPC são chamados, em geral, de algoritmos que passam mensagem (message-passing algorithms).

Esses algoritmos operam passando mensagens pelas arestas do grafo de Tanner. Cada vértice do grafo de Tanner trabalha localmente; a informação que esse vértice tem são os seus vértices vizinhos.

Outro nome comum para esses algoritmos de decodificação de códigos LDPC é algoritmos de decodificação iterativa, já que as mensagens são passadas entre os vértices de checagem e variáveis iterativemente, até atingir a decodificação ou atingir o numero máximo de iterações permitido.

Algoritmos de decodificação iterativa (cont)

Há diversos algoritmos de decodificação iterativa. Eles dependem do tipo de operação executada nos vértices do grafo.

O algoritmo de interesse nesse curso é bit-flipping, onde as mensagens são bits, e as operações se baseiam em verificar as equações de checagem.

Em outros algoritmos as mensagens sao probabilidades indicando o nível de confiança no resultado desse bit. Esses algoritmos são chamados de belief propagation. Um algoritmo importante desse tipo em teoria de códigos é o algoritmo de soma-produto (sum-product algorithm); para mais informação sobre este algoritmo ver a seção 2.3, p. 31-39 do artigo da Sarah Johnson.

Algoritmo de decodificação para códigos LDPC e MDPC

- BIKE: Bit Flipping Key Encapsulation:
  - QC-MDPC
  - Bit flipping
- LEDAcrypt:
  - QC-LDPC
  - Bit flipping

- Bit flipping:
  - Hard decision: decodifica cada bit considerando que ele é definitivamente 0 ou 1
- Soma-produto:
  - Soft decision: a entrada para cada bit é uma probabilidade

Algoritmo de decodificação bit-flipping

Idéia: se um bit de uma palavra de código está envolvido num número grande de equações não satisfeitas, há uma alta chance desse bit estar errado.

É um algoritmo de tipo hard decision: as variáveis tem valor 0 ou 1. Usa o grafo de Tanner como representação da matriz de paridade.

- O algoritmo calcula cada equação de paridade.
- O bit (ou bits) usado(s) no maior número de equações não satisfeitas é (são) "flipado(s)" (se era 0 passa a ter valor 1 e reciprocamente).
- As equações são recomputadas.

O algoritmo pára quando todas as equações estão satisfeitas ou se o máximo número de iterações é atingido.

Algoritmo de decodificação bit-flipping

Idéia: se um bit de uma palavra de código está envolvido num número grande de equações não satisfeitas, há uma alta chance desse bit estar errado.

É um algoritmo de tipo hard decision: as variáveis tem valor 0 ou 1.
Usa o grafo de Tanner como representação da matriz de paridade.

- O algoritmo calcula cada equação de paridade.
- O bit (ou bits) usado(s) no maior número de equações não satisfeitas é (são) "flipado(s)" (se era 0 passa a ter valor 1 e reciprocamente).
- As equações são recomputadas.

O algoritmo pára quando todas as equações estão satisfeitas ou se o máximo número de iterações é atingido.

Exemplo de decodificação por bit-flipping

Seja H a matriz de paridade de um código LDPC:

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

Desenhar o grafo de Tanner no quadro negro.
Seja a mensagem original $m = [001011]$ recebida como $y = m + e = [101011]$.

Da matriz H temos as seguintes equações de paridade:

$$\begin{aligned} c_1 &= v_1 \oplus v_2 \oplus v_4 = 0 \\ c_2 &= v_2 \oplus v_3 \oplus v_5 = 0 \\ c_3 &= v_1 \oplus v_5 \oplus v_6 = 0 \\ c_4 &= v_3 \oplus v_4 \oplus v_6 = 0 \end{aligned}$$

Exemplo de decodificação por bit-flipping (cont)

Substituimos $y = [101011]$ nas equações de paridade

$$\begin{aligned} c_1 &= 1 \oplus 0 \oplus 0 = 1 \\ c_2 &= 0 \oplus 1 \oplus 1 = 0 \\ c_3 &= 1 \oplus 1 \oplus 1 = 1 \\ c_4 &= 1 \oplus 0 \oplus 1 = 0 \end{aligned}$$

As equações $c_1 = v_1 \oplus v_2 \oplus v_4$ e $c_3 = v_1 \oplus v_5 \oplus v_6$ não são satisfeitas.

O vértice variável v_1 é parte de 2 equações que falharam e v_2, v_4, v_5 e v_6 são cada uma parte de 1 equação não satisfeita. Então, o bit v_1 é trocado de valor:

$$y = [101011] \rightarrow y' = [001011].$$

Exemplo de decodificação por bit-flipping

Novamente, substituimos $y' = [001011]$ nas equações de paridade:

$$\begin{aligned} c_1 &= 0 \oplus 0 \oplus 0 = 0 \\ c_2 &= 0 \oplus 1 \oplus 1 = 0 \\ c_3 &= 0 \oplus 1 \oplus 1 = 0 \\ c_4 &= 1 \oplus 0 \oplus 1 = 0 \end{aligned}$$

Todas as equações de paridade são satisfeitas!

Como recebemos $y = m + e = [101011]$, o erro é $e = [100000]$.

Decodificação por bit-flipping

No exemplo anterior, a matriz H era pequena e de fato o número de zeros nao era tão grande assim. Na prática, o tamanho das matrizes LDPC é da ordem dos milhares de linhas e colunas, tornando o método de bit flipping muito eficiente.

Um dos problemas maiores desse método é quando o grafo de Tanner apresenta ciclos de tamanho pequenos. Ciclos de tamanho pequeno no grafo de Tanner de um código reduzem a eficiência de algoritmos iterativos.

Requer-se, portanto, que a cintura (girth) do grafo, ou seja, o tamanho dos ciclos menores, seja o maior possível.

Um problema importante na prática é encontrar construções de matrizes LDPC tais que a cintura do grafo de Tanner seja pelo menos 6 (com cintura 4, em geral, o funcionamento desse algoritmo não é adequado).

Exemplo de problema na decodificação por bit-flipping

Vamos considerar agora a seguinte matriz de paridade e seu grafo de Tanner, que contém ciclos de comprimento 4:

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

Desenhar o grafo de Tanner no quadro negro.

Temos vários ciclos de tamanho 4, por exemplo:

$$c_1 \to v_1 \to c_2 \to v_2 \to c_1,$$

$$c_3 \to v_3 \to c_4 \to v_6 \to c_3.$$

Exemplo de problema na decodificação por bit-flipping

Seja a mensagem original $m = [001001]$ recebida como $y = m + e = [101001]$.

Da matriz H temos as seguintes equações de paridade:

$$\begin{aligned} c_1 &= v_1 \oplus v_2 \oplus v_4 = 0 \\ c_2 &= v_1 \oplus v_2 \oplus v_5 = 0 \\ c_3 &= v_3 \oplus v_5 \oplus v_6 = 0 \\ c_4 &= v_3 \oplus v_4 \oplus v_6 = 0 \end{aligned}$$

Exemplo de problema na decodificação por bit-flipping

Substituimos $y = [001001]$ nas equações de paridade

$$\begin{aligned} c_1 &= 1 \oplus 0 \oplus 0 = 1 \\ c_2 &= 1 \oplus 0 \oplus 0 = 1 \\ c_3 &= 1 \oplus 0 \oplus 1 = 0 \\ c_4 &= 1 \oplus 0 \oplus 1 = 0 \end{aligned}$$

As equações $c_1 = v_1 \oplus v_2 \oplus v_4$ e $c_3 = v_1 \oplus v_2 \oplus v_5$ não são satisfeitas.

Os vértices variáveis v_1 e v_2 são parte de 2 equações não satisfeitas, e v_4 e v_5 são cada uma parte de 1 equação não satisfeita. Então, os bits v_1 e v_2 são trocados de valor:

$$y = [101001] \rightarrow y' = [011001].$$

Exemplo de problema na decodificação por bit-flipping

Novamente, substituimos $y' = [011001]$ nas equações de paridade:

$$\begin{aligned} c_1 &= 0 \oplus 1 \oplus 0 = 1 \\ c_2 &= 0 \oplus 1 \oplus 0 = 1 \\ c_3 &= 1 \oplus 0 \oplus 1 = 0 \\ c_4 &= 1 \oplus 0 \oplus 1 = 0 \end{aligned}$$

Temos que trocar os bits v_1 e v_2 de novo!

$$y' = [011001] \rightarrow y = [101001].$$

As variáveis v_1 e v_2 sempre vão estar envolvidas em equações de paridade não satisfeitas! Não é possível determinar quais bits estão errados.

O algoritmo não converge.

Ciclos do grafo de Tanner

Ciclos do grafo de Tanner afetam a convergência do algoritmo de decodificação. Quanto menor a cintura do grafo, maior é a influência no resultado. Então, queremos construções que evitem ciclos pequenos.

A performance melhora consideravelmente evitando 4-ciclos e 6-ciclos dos grafos de Tanner de códigos LDPC, mas o retorno tende a diminuir com a remoção de cinturas maiores.

Quando consideramos propriedades da matriz de paridade de um código LDPC, em geral, estamos falando de uma matriz específica. Diferentes matrizes de um mesmo código podem ter comportamentos diferentes.

Exemplo de matrizes do mesmo código

Essas duas matrizes de paridade correspondem ao mesmo código mas o comportamento é diferente, já que a primeira tem um ciclo de comprimento 4:

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \end{pmatrix},$$

$$H = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

Vamos rever esses conceitos anteriores junto com um pseudo-código do algoritmo de bit-flipping; seção 2.2, páginas 26-30 e na seção 4.1, páginas 56-57 do texto da Sarah Johnson.

Códigos Cíclicos, Quase-Cíclicos (QC) e QC-LDPC

Códigos cíclicos

Definição

Um código linear $C(n,k)$ sobre $\mathbb{F}_q$ é cíclico, se $(c_0,\ldots,c_{n-1}) \in C$ implica que $(c_{n-1},c_0,\ldots,c_{n-2}) \in C$.

Exemplo: $C = \{000, 110, 101, 011\}$ é um código cíclico.

Códigos cíclicos podem ser caracterizados com polinômios.

Teorema

Um código linear $C(n,k)$ sobre $\mathbb{F}_q$ é cíclico se e somente se C é um ideal de $\mathbb{F}_q[x]/(x^n-1)$.

Retornamos a esse teorema após ter visto a álgebra de polinômios sobre corpos finitos necessária.

Teorema

Um código linear $C(n,k)$ sobre $\mathbb{F}_q$ é cíclico se e somente se C é um ideal de $\mathbb{F}_q[x]/(x^n-1)$.

Demonstração.

Se C é um código cíclico e $c(x) \in C$, temos que $xc(x), x^2c(x), x^3c(x), \ldots$ também pertencem a C. Seja $a(x) = \sum_i a_i x^i \in \mathbb{F}_q[x]/(x^n-1)$. Como $a(x)c(x) = \sum_i a_i(x^i c(x))$ e C é um subespaço vetorial sobre $\mathbb{F}_q$, temos que C é um ideal.

Reciprocamente, se C é um ideal de $\mathbb{F}_q[x]/(x^n-1)$ e $c(x) = \sum_{i=0}^{n-1} c_i x^i$ é uma palavra-código, então $xc(x)$ também é uma palavra-código. Logo, C é cíclico. □

Teorema

Um código linear $C(n,k)$ sobre $\mathbb{F}_q$ é cíclico se e somente se C é um ideal de $\mathbb{F}_q[x]/(x^n-1)$.

Demonstração.

Se C é um código cíclico e $c(x) \in C$, temos que $xc(x)$, $x^2c(x)$, $x^3c(x), \ldots$ também pertencem a C. Seja $a(x) = \sum_i a_i x^i \in \mathbb{F}_q[x]/(x^n-1)$. Como $a(x)c(x) = \sum_i a_i(x^i c(x))$ e C é um subespaço vetorial sobre $\mathbb{F}_q$, temos que C é um ideal.

Reciprocamente, se C é um ideal de $\mathbb{F}_q[x]/(x^n-1)$ e $c(x) = \sum_{i=0}^{n-1} c_i x^i$ é uma palavra-código, então $xc(x)$ também é uma palavra-código. Logo, C é cíclico. □

Definição

Seja $C = (g)$ um código cíclico. Dizemos que g é o polinômio gerador de C e $h = (x^n-1)/g$ é o polinômio verificador de C.

Teorema

Seja C um ideal não nulo em $\mathbb{F}_q[x]/(x^n-1)$, isto é, C é um código cíclico de comprimento n.

1. *O código C é gerado por um único polinômio mônico g de grau mínimo em C.*
2. *O polinômio gerador g de C é um fator de x^n-1.*
3. *Em $\mathbb{F}_q[x]$, qualquer $c \in C$ pode ser escrito unicamente como $c = fg$, onde $\mathrm{grau}(f) < n-r$ e $\mathrm{grau}(g) = r$. Além disso, a dimensão de C é $n-r$. (Assim, a mensagem f se torna a palavra-código fg.)*

Teorema

4. *Se* $g(x) = g_0 + g_1 x + \cdots + g_r x^r$, *então* C *é gerado como um subespaço de* $\mathbb{F}_q^n$ *pelas linhas da matriz geradora*

$$G = \begin{bmatrix} g_0 & g_1 & \cdot & \cdot & \cdot & g_r & 0 & 0 & 0 \\ 0 & g_0 & g_1 & \cdot & \cdot & \cdot & g_r & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & 0 & g_0 & g_1 & \cdot & \cdot & \cdot & g_r \end{bmatrix}$$

$$= \begin{bmatrix} g(x) & 0 & 0 & \cdot & \cdot & \cdot & \cdot & 0 & 0 \\ 0 & xg(x) & 0 & \cdot & \cdot & \cdot & \cdot & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & \cdot & \cdot & \cdot & 0 & x^{n-r-1}g(x) \end{bmatrix}.$$

Códigos de Hamming

Definição

Seja m *um inteiro maior ou igual a* 2. *Um código binário* C_m *de comprimento* $n = 2^m - 1$ *com uma matriz de paridade* H *de ordem* $m \times (2^m - 1)$ *é chamado de código binário de Hamming, se as colunas de* H *correspondem às representações binárias dos inteiros* $1, 2, \ldots, 2^m - 1$.

Exemplo: C_3 tem matriz de paridade

$$H = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

A dimensão de C_m é $2^m - 1 - m$. Quaisquer duas colunas são linearmente independentes, já que nenhuma coluna é múltipla de outra. Por outro lado, há três colunas que são linearmente dependentes. Por exemplo,

$$\begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ 1 \end{pmatrix}.$$

Concluímos que a distância mínima de C_m é $2 + 1 = 3$ e assim C_m corrige um erro.

Teorema

O código de Hamming com parâmetros $n = 2^m - 1$, $k = n - m$ e $d = 3$ é um código com matriz geradora

$$G = \begin{pmatrix} M^{(1)} & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & xM^{(1)} & \cdot & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & \cdot & \cdot & x^{n-m-1}M^{(1)} \end{pmatrix},$$

onde o polinômio $M^{(1)} = g$ (o polinômio gerador do código), é o polinômio minimal dos elementos na classe lateral ciclotômica $C_1 = \{1, 2, 4, \ldots\}$ módulo $n = 2^m - 1$.

A seguir veremos um exemplo de código de Hamming construído usando polinômios. Voltaremos aos códigos cíclicos para ver BCH codes e a proposta HQC de NIST após falar de polinômios minimais.

Exemplo

Para $n = 2^3 - 1 = 7$ e $m = 3$, temos que $M^{(1)}(x) = 1 + x + x^3$ é o polinômio minimal dos elementos na classe lateral ciclotômica $C_1 = \{1, 2, 4\}$ módulo 7 e, então, a matriz geradora é

$$G = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

Se não conhecemos a matriz de paridade H, podemos usar o fato de que $h(x) = (x^7 - 1)/(x^3 + x + 1) = x^4 + x^2 + x + 1$ e, assim, temos que

$$H = \begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}.$$