Universidade do Estado do Rio de Janeiro

Centro Tecnologia e Ciências

Instituto de Matemática e Estatistica

Isaque Damião Siqueira Costa

Código de Hamming: detecção e correção de erros

Rio de Janeiro

2018

Isaque Damião Siqueira Costa

Código de Hamming: detecção e correção de erros

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-graduação em Matemática, da Universidade do Estado do Rio de Janeiro.

Orientador: Prof. Dr. Roberto Alfonso Olivares Jara

Rio de Janeiro

2018

CATALOGAÇÃO NA FONTE
UERJ / REDE SIRIUS / BIBLIOTECA CTC-A

C837 Costa, Isaque Damião Siqueira.
Código de Hamming: detecção e correção de erros / Isaque Damião Siqueira Costa – 2018.
65 f. : il.

Orientador: Roberto Alfonso Olivares Jara.
Dissertação (Mestrado Profissional em Matemática em Rede Nacional) - Universidade do Estado do Rio de Janeiro, Instituto de Matemática e Estatística.

1. Análise numérica – Processamento de dados – Teses. I. Jara, Roberto Alfonso Olivares. II. Universidade do Estado do Rio de Janeiro. PROFMAT. III. Título.

CDU 519.6

Patricia Bello Meijinhos – CRB-7/5217 – Responsável pela elaboração da ficha catalográfica

_____ _____

Assinatura Data

Isaque Damião Siqueira Costa

Código de Hamming: detecção e correção de erros

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Mestrado Profissional em Matemática, da Universidade do Estado do Rio de Janeiro.

Aprovada em 27 de fevereiro de 2018.

Banca Examinadora

_____
Prof. Dr. Roberto Alfonso Olivares Jara (Orientador)
Instituto de Matemática e Estatística - UERJ

_____
Profª. Dra. Patrícia Nunes da Silva
Instituto de Matemática e Estatística - UERJ

_____
Prof. Dr. Johnny Villavicencio Tafur
Centro Universitário Carioca

Rio de Janeiro

2018

DEDICATÓRIA

À memória do prof. Elon Lages Lima.

AGRADECIMENTOS

Não posso deixar de agradecer ao meu orientador, pelo comprometimento, seriedade, apoio e todo esforço que dedicou, a despeito dos problemas médicos que teve (desejo melhoras!). E pelo seu olhar rigoroso, que escrutinou cada detalhe deste trabalho, acusando com propriedade desde os meus erros matemáticos até os de estilo e gramática – apesar de o português não ser sua língua materna.

Agradeço aos meus pais, que sempre se orgulharam de mim, mesmo talvez não entendendo muito bem o que é Matemática, nem para que serve exatamente um curso de Mestrado. Nem mesmo "a quem terá puxado esse garoto para escolher fazer uma coisas dessas da sua vida? Puxa, criei com tanto carinho!".

Agradeço aos Mestres que tive, colegas professores e excelentes alunos com quem aprendi e continuo aprendendo.

Mas o meu maior agradecimento tem de ser para minha grande companheira de vida, Josineia Mendes da Costa, que está tão presente e de modo tão impactante há tanto tempo em minha história que chega a ser impossível dimensionar sua importância. Eu não seria professor de Matemática se não fosse pela sua influência. Mas não é só isso. Eu não teria ido aos lugares que eu fui, tomado as decisões que tomei, conhecido as músicas que ouço, abraçado as ideias que defendo, não moraria onde moro, não usaria as roupas que visto, não seria feliz como eu sou. Eu simplesmente não existiria, teria sido uma pessoa completamente diferente. Irreconhecivel. Um estranho pra mim mesmo. Mas por um capricho do cosmos, do destino, do universo, das leis da probabilidade quântica – o que seja – nos encontramos. E por isso, para o bem ou para o mal, eu sou quem eu sou. E ela é quem é. E nós somos um.

Um problema que vale a pena ser atacado prova seu valor contra-atacando

Piet Hein

RESUMO

COSTA, Isaque Damião Siqueira. *Código de Hamming: detecção e correção de erros*. 2018. 65 f. Dissertação (Mestrado em Matemática) – PROFMAT, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2018.

Neste trabalho, vamos estudar o uso de códigos para transmissões mecânicas e eletrônicas de dados, e os métodos matemáticos usados para a detecção e correção de erros de envio. A detecção de erros será feita por dígitos de checagem, exemplificadas nos seguintes códigos: CPF, UPC e ISBN. Para correção de erros, veremos os códigos de Hamming.

Palavras-chave: Código de Hamming. Correção de erros.

ABSTRACT

COSTA, Isaque Damião Siqueira. *Hamming's code: error detection and correction.* 2018. 65 f. Dissertação (Mestrado em Matemática) – PROFMAT, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2018.

In this work, we will study the use of codes for mechanical and electronic data transmissions, and the mathematical methods used for the detection and correction of sending errors. Error detection will be made by check digits, exemplified in the following codes: CPF, UPC and ISBN. For error correction, we will see the Hamming codes.

Keywords: Hamming codes. Error correction.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ASCII	American Standard Code for Information Interchange
CPF	Cadastro de Pessoas Físicas
ISBN	International Standard Book Number
LD	Linearmente dependente
LI	Linearmente independente
UPC	Universal Product Code
SDR	Sistema de direcionamento remote

SUMÁRIO

INTRODUÇÃO

Neste trabalho, vamos estudar o uso de códigos lineares para o envio de transmissões eletrônicas, e os métodos matemáticos usados para a detecção e correção de erros nessas transmissões. Deste modo, teremos a oportunidade de explorar a aplicação de conceitos como vetores e matrizes em problemas reais, com a possibilidade de justificar para o aluno do ensino médio a importância desses conteúdos para além da sala de aula.

A ubiquidade da informática em nossos dias e o exercício da cidadania nos oferecem uma variedade de exemplos, acessíveis pela experiência quotidiana como os dígitos verificadores dos documentos oficiais de identificação (CPF), os códigos de barras dos produtos (UPC), o sistema ISBN de catalogação de livros, e etc.

Em seguida, nos concentraremos nos códigos lineares – em particular nos códigos Hamming para correção de erros de transmissão, tomando como base principalmente Poole (2014).

1. **RESULTADOS PRELIMINARES**

1.1. **Aritmética modular**

Apresentamos nesta seção, de maneira resumida, uma construção e as principais propriedades do conjunto das classes de congruências módulo um número inteiro. Tais resultados serão utilizados por todo o texto. O leitor interessado em maiores esclarecimentos, ou em aprofundar os resultados apresentados aqui, pode consultar Hefez (2002).

Definição 1.1.1.

Sejam a, b, k inteiros com $k > 0$. Dizemos que ***a* é congruente a *b* módulo *k***, e escrevemos "$a \equiv b \bmod k$", quando k divide $a - b$.

Exemplo: $25 \equiv 13 \bmod 6$, pois 6 divide $25 - 13 = 12$. Analogamente, $-10 \equiv 4 \bmod 7$, pois 7 divide $-10 - 4 = -14$.

Definição 1.1.2.

Sejam a e k inteiros com $k > 0$. A **classe de congruência de *a* módulo *k*,** denotada por $[a]$, consiste em todos os números inteiros que são congruentes a a módulo k, isto é,

$$[a] = \{b: b \in \mathbb{Z} \; e \; b \equiv a \; mod \; k\}.$$

Dizer que $b \equiv a \bmod k$ significa que $b - a = nk$ para algum inteiro n ou, equivalentemente, que $b = a + nk$. Assim

$$[a] = \{a + nk : n \in \mathbb{Z}\}.$$

Exemplo: Na congruência módulo 5 temos

$$[11] = \{11 + 5n: n \in \mathbb{Z}\} = \{11, 11 \pm 5, 11 \pm 10, 11 \pm 15, \dots\}$$
$$= \{\dots, -14, -9, -4, 1, 6, 11, 16, 21, 24, \dots\}.$$

Um importante resultado sobre as classes de congruências é o seguinte.

Teorema 1.1.3.

Seja k um número inteiro maior do que 1.

i) Se a é um número inteiro e r é seu resto $(0 \leq r < k)$ quando dividido por k, então $[a] = [r]$.

ii) Existem exatamente k distintas classes de congruências módulo k, a saber $[0], [1], [2], \ldots, [k-1]$.

O conjunto de todas as classes de congruências modulo k é denotado por $\mathbb{Z}_k$, isto é,

$$\mathbb{Z}_k = \{[0], [1], [2], \ldots, [k-1]\}.$$

Em $\mathbb{Z}_k$ são definidas para quaisquer $[a]$ e $[b]$ em $\mathbb{Z}_k$ as seguintes duas operações:

$$[a] + [b] = [a + b]$$
$$[a] \times [b] = [a \times b].$$

Exemplo: Em $\mathbb{Z}_6$ temos $[3] + [5] = [3 + 5] = [8] = [2]$ e $[2] \times [5] = [2 \times 5] = [10] = [4]$.

De agora em diante toda classe de congruência $[a]$ será denotada apenas por a.

As operações de adição e multiplicação no conjunto $\mathbb{Z}_k$ possuem as seguintes propriedades para quaisquer a, b, c em $\mathbb{Z}_k$:

Proposição 1.1.4.

Propriedades de $\mathbb{Z}_k$.

P1) Comutatividade: $a + b = b + a$ $\quad a \cdot b = b \cdot a$

P2) Associatividade: $(a + b) + c = a + (b + c)$ $\quad (a \cdot b) \cdot c = a \cdot (b \cdot c)$

P3) Existência do elemento neutro: $a + 0 = a$ $\quad a \cdot 1 = a$.

P4) Existência do elemento oposto: Para cada $a \in \mathbb{Z}_k$ existe um único elemento $-a \in \mathbb{Z}_k$ tal que $a + (-a) = 0 = -a + a$. E $-a$ é chamado o "oposto" de a.

P5) Distributividade: $(a + b) \cdot c = a \cdot c + b \cdot c$.

Em geral, qualquer conjunto munido de duas operações (adição e multiplicação) satisfazendo as cinco propriedades acima é chamado de **anel comutativo com identidade**. Os Principais modelos para essa estrutura algébrica são o conjunto $\mathbb{Z}$ dos números inteiros e o conjunto $\mathbb{Z}_k$. Um diferencial entre anéis, e em particular entre os anéis $\mathbb{Z}$ e $\mathbb{Z}_k$, é o seguinte:

Definição 1.1.5.

Um **domínio de integridade** é um anel comutativo com identidade que satisfaz a propriedade:

P6) Se $a \cdot b = 0$, então $a = 0$ ou $b = 0$.

Equivalentemente, num domínio de integridade, o produto de dois elementos não nulos é não nulo. Sabe-se que $\mathbb{Z}$ goza dessa propriedade e $\mathbb{Z}_k$ não. De fato, em $\mathbb{Z}_{10}$ por exemplo, tem-se que

$$2 \neq 0 \text{ e } 5 \neq 0, \text{ mas } 2 \cdot 5 = 10 = 0.$$

Um caso muito importante, tratando-se da estrutura algébrica de $\mathbb{Z}_k$, acontece quando $k = p$ é um número primo. Observe primeiramente que em $\mathbb{Z}$ os únicos elementos que possuem inverso (multiplicativo) são 1 e -1 enquanto que em $\mathbb{Z}_p$ todos os elementos não nulos possuem inverso (lembre que um elemento $a \in \mathbb{Z}_k$ é invertível se, e somente se, $mdc(a, k) = 1$)) . Daí segue que $\mathbb{Z}_p$ com p primo se enquadra na seguinte estrutura.

Definição 1.1.6.

Um **corpo** é um anel comutativo com identidade que satisfaz a propriedade:

P7) Para cada $a \neq 0$, existe um único elemento a^{-1} tal que $a \cdot a^{-1} = a^{-1} \cdot a = 1$. E a^{-1} é chamado o "inverso" de a.

Exemplo: Em $\mathbb{Z}_5$, os elementos 1, 2, 3, e 4 são invertíveis e:

$1^{-1} = 1$ $\quad 2^{-1} = 3$ $\quad 3^{-1} = 2$ $\quad 4^{-1} = 4$

Os principais modelos para a estrutura de corpo são os conjuntos numéricos $\mathbb{Q}$ dos números racionais, $\mathbb{R}$ dos números reais, juntamente com o conjunto $\mathbb{Z}_p$ de todas as classes de congruências módulo um primo p.

Corolário 1.1.7. Todo corpo é um domínio de integridade[1].

Demonstração. Suponha que $a \cdot b = 0$. Se $a = 0$ não há o que demonstrar. Suponha agora que $a \neq 0$. Logo a é invertível (isto é, existe a^{-1} no corpo). Multiplicando pelo inverso de a na igualdade $a \cdot b = 0$ temos:

$$a^{-1} \cdot a \cdot b = a^{-1} \cdot 0$$

[1]Em particular $\mathbb{Z}_p$ com p primo também o é

$$1 \cdot b = 0$$
$$b = 0.$$

Exemplo: Como 2 é primo, $\mathbb{Z}_2$ é um corpo, as operações de adição e multiplicação em $\mathbb{Z}_2 = \{0, 1\}$ são do seguinte modo:

$$0 + 0 = 0 \quad 0 + 1 = 1 \quad 1 + 0 = 1 \quad 1 + 1 = 0$$
$$0 \cdot 0 = 0 \quad 0 \cdot 1 = 0 \quad 1 \cdot 0 = 0 \quad 1 \cdot 1 = 1$$

De forma mais compacta, podemos escrever:

+	0	1
0	0	1
1	1	0

Tabela 1. Adição em $\mathbb{Z}_2$.

·	0	1
0	0	0
1	0	1

Tabela 2. Multiplicação em $\mathbb{Z}_2$.

1.2. **Espaço e subespaço vetorial**

Apresentamos nesta seção um resumo dos conceitos de álgebra linear que serão necessários ao longo do trabalho. Mais detalhes podem ser obtidos em Poole (2014).

Definição 1.2.1. (Espaço vetorial)

Um conjunto V não vazio é dito um espaço vetorial sobre um corpo K (por exemplo, $\mathbb{Q}$, $\mathbb{R}$ ou $\mathbb{C}$), se:

a) Se $u, v \in V$, então $u + v \in V$.

b) Se $v \in V$ e $k \in K$, então $k \cdot v \in V$.

Com as seguintes propriedades.

Considere $u, v, w \in V$ e $\alpha, \beta \in K$.

P1) $u + v = v + u$.

P2) $(u + v) + w = u + (v + w)$.

P3) Existe $0 \in V$, tal que para todo $v \in V$, temos que: $v + 0 = v$.

P4) Existe $-v \in V$, tal que para todo $v \in V$, temos que: $v + (-v) = 0$.

P5) $\alpha(u + v) = \alpha u + \alpha v$.

P6) $(\alpha + \beta)v = \alpha v + \beta v$.

P7) $\alpha(\beta u) = (\alpha\beta)u$.

P8) $1 \cdot u = u$.

Fazendo $K = \mathbb{R}$, temos os seguintes exemplos de espaços vetoriais:

a) $\mathbb{R}^n$, com $n \in \mathbb{N}\backslash\{0\}$: conjunto de todas as n-uplas de componentes reais.

b) $M_{m\times n}(\mathbb{R})$: conjunto de todas as matrizes reais de m linhas e n colunas, com elementos reais.

c) $\{a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 | a_i \in \mathbb{R}, i = 1, 2, \ldots, n\}$: conjunto de todos os polinômios de grau menor ou igual a n, incluindo o polinômio identicamente nulo.

Fazendo $K = \mathbb{Z}_p$, com p primo, temos que $\mathbb{Z}_p^n$ é um espaço vetorial.

Observação: os elementos de um espaço vetorial são ditos vetores.

Um código pode ser criado tomando como símbolos vetores sobre $\mathbb{Z}_k$. Definimos $\mathbb{Z}_k^n$ o conjunto de todos os vetores com n coordenadas em $\mathbb{Z}_k$. Isto é, $\mathbb{Z}_k^n = \{(x_1, x_2, \ldots, x_n): x_1, x_2, \ldots . x_n \in \mathbb{Z}_k\}$.

Quando $k = 2$, dizemos que os elementos de $\mathbb{Z}_2^n$ formam um código binário de comprimento n. Por exemplo, a tabela ASCII é um exemplo de código binário (aceita apenas coordenadas iguais a zero ou um) de comprimento 8 (têm exatamente oito coordenadas) e (0, 1, 1, 0, 0, 0, 0, 1) é um de seus símbolos, mais especificamente, o que representa o caractere "*a*".

Definição 1.2.2 (Subespaço vetorial)

W é subespaço vetorial de V se:

a) W é um subconjunto não vazio de V; e

b) W é um espaço vetorial, em relação às mesmas operações definidas em V.

Na prática, para concluir que W é subespaço vetorial de V, basta verificar que:

a) $0 \in W$

b) Se $u, v \in W$, então $u + v \in W$.

c) Se $v \in W$ e $k \in K$, então $k \cdot v \in W$.

Observação: $\{0\}$ e V são ditos subespaços triviais de V.

Subespaços vetoriais de $\mathbb{R}^2$:

a) a origem;

b) toda reta que contém a origem;

c) o próprio $\mathbb{R}^2$.

Subespaços vetoriais de $\mathbb{R}^3$:

a) a origem;

b) toda reta que contém a origem;

c) todo plano que contém a origem;

d) o próprio $\mathbb{R}^3$.

Subespaços vetoriais de $\mathbb{Z}_2^2$:

a) $\left\{\begin{bmatrix}0\\0\end{bmatrix}\right\}$

b) $\left\{\begin{bmatrix}0\\0\end{bmatrix}, v\right\}$, onde v pode ser $\begin{bmatrix}1\\0\end{bmatrix}$, $\begin{bmatrix}0\\1\end{bmatrix}$ ou $\begin{bmatrix}1\\1\end{bmatrix}$

c) $\left\{\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}\right\}$ que é o próprio $\mathbb{Z}_2^2$.

Definição 1.2.3 (Combinação linear)

Considere $u_1, u_2, \dots, u_{n-1}, u_n \in V$. Dizemos que u_n é uma combinação linear de $u_1, u_2, \dots, u_{n-1}$ se existem $\alpha_1, \alpha_2, \dots, \alpha_{n-1} \in K$ tais que:

$$u_n = \alpha_1 u_1 + \alpha_2 u_2 + \cdots + \alpha_{n-1} u_{n-1}$$

Definição 1.2.4 (Dependência linear)

Um conjunto $u_1, u_2, \dots, u_{n-1}, u_n \in V$ é dito linearmente independente (L.I.) quando:

$$\alpha_1 u_1 + \alpha_2 u_2 + \cdots + \alpha_n u_n = 0 \Rightarrow \alpha_1 = \alpha_2 = \cdots = \alpha_n = 0$$

Caso contrário, o conjunto é dito linearmente dependente (L.D.) e existe um elemento do conjunto que pode ser escrito como combinação linear dos outros.

Definição 1.2.5 (Vetores geradores)

Se $B = \{v_1, v_2, \dots, v_n\} \subset V$ e todo elemento de V pode ser escrito como combinação linear dos elementos de B, então dizemos que $v_1, v_2, \dots, v_n$ são geradores de V e indicamos por:

$$ger(v_1, v_2, \dots, v_n) = V$$

O conjunto de geradores de um espaço vetorial não é único.

Definição 1.2.6 (Base)

Dizemos que $\{v_1, v_2, \dots, v_n\}$ é uma base de V, se:

a) $ger(v_1, v_2, \dots, v_n) = V$; e

b) $v_1, v_2, \dots, v_n$ são L.I.

Teorema 1.2.7

Se $B_1 = \{v_1, v_2, \dots, v_n\}$ e $B_2 = \{u_1, u_2, \dots, u_m\}$ são bases de V, então $m = n$.

Observação. Nas condições do Teorema 1.2.7, dizemos que n é a dimensão de V, e indicamos $dim\, V = n$.

1.3. **Produto interno, norma e distância**

Definição 1.3.1 (Produto interno)

Seja V um espaço vetorial. Um Produto interno em V é uma função que a cada par de vetores u e v em V associa o número real $\langle u, v\rangle$ que satisfaz as seguintes propriedades:

1. $\langle u, v\rangle = \langle v, u\rangle$
2. $\langle u, v + w\rangle = \langle u, v\rangle + \langle u, w\rangle$
3. $\langle cu, v\rangle = c\langle u, v\rangle$
4. $\langle u, u\rangle \geq 0$ e $\langle u, u\rangle = 0$ se, e somente se, $u = 0$

Onde u, v e w são quaisquer vetores em V e c é qualquer escalar.

Definição 1.3.2 (Produto interno usual)

Sejam $u = (u_1, u_2, \dots, u_n)$ e $v = (v_1, v_2, \dots, v_n)$ em $\mathbb{R}^n$. Definimos

$$\langle u, v\rangle = u_1 \cdot v_1 + u_2 \cdot v_2 + \cdots + u_n \cdot v_n.$$

É fácil ver que a operação acima define um produto interno, chamado de produto interno usual de $\mathbb{R}^n$. Sem qualquer problema pode-se considerar também o produto interno usual em $\mathbb{Z}_2^n$ em vez de $\mathbb{R}^n$. Neste caso, <u, v> ϵ $\mathbb{Z}_2$.

A partir de agora vamos usar a notação $u \cdot v$ para representar o produto interno usual.

Definição 1.3.3 (Vetores ortogonais)

Quando $u \cdot v = 0$, dizemos que o vetores u e v são ortogonais.

Definição 1.3.4 (Norma)

Uma norma em um espaço vetorial V é uma aplicação que associa a cada vetor v um número real $\|v\|$, chamado norma de v, de modo que as seguintes propriedades sejam satisfeitas para todos os vetores u e v e todos os escalares c:

1. $\|v\| \geq 0$ e $\|v\| = 0$ se, e somente se, $v = 0$
2. $\|cv\| = |c| \cdot \|v\|$
3. $\|u + v\| \leq \|u\| + \|v\|$

Existem vários tipos de norma, segue abaixo dois exemplos.

Definição 1.3.5 (Norma do produto interno).

Em qualquer espaço vetorial com produto interno, $\|v\| = \sqrt{< \boldsymbol{v}, \boldsymbol{v} >}$ define uma norma.

Como ilustração, considerando $\mathbb{R}^n$com o produto interno usual usual:

Se $u = (u_1, u_2, \ldots u_n)$, então

$$\|u\| = \sqrt{\boldsymbol{u} \cdot \boldsymbol{u}} = \sqrt{u_1^2 + u_2^2 + \cdots + u_n^2}$$

Exemplo: $\|(3,4,0)\| = \sqrt{3^2 + 4^2 + 0^2} = 5$

Definição 1.3.6 (Norma de Hamming)

Considere o vetor $u \in \mathbb{Z}_2^n$, então a norma de Hamming é dada por

$$\|u\|_H = w(u)$$

Onde $w(u)$ é chamado peso de u e corresponde à quantidade de 1's que existem no vetor u.

Exemplo: $\|(1,0,1,0,1)\|_H = w(1,0,1,0,1) = 3$.

Definição 1.3.7 (Distância)

Em qualquer espaço vetorial normado, a distância entre os vetores u e v é o número real $d(u,v) = \|u - v\|$.

As seguintes propriedades são válidas para todos os vetores u, v e w em V:

1. $d(u,v) \geq 0$ e $d(u,v) = 0$ se, e somente se, $u = v$.
2. $d(u,v) = d(v,u)$
3. $d(u,w) \leq d(u,v) + d(v,w)$.

Para uma demostração das afirmações contidas nas definições 1.3.6 e 1.3.7 veja por exemplo o exemplo 4 e o Teorema 1, respectivamente, da seção 7.3 do livro de David Pool citado na bibliografia.

Definição 1.3.8 (Distância de Hamming)

Como em $\mathbb{Z}_2$, u−v = u + v, a distância

$$d_H(u, v) = \|u - v\|_H = \|u + v\|_H = w(u + v)$$

é chamada distância de Hamming.

Uma função distãncia d com essas propriedades é chamada de **métrica**. Um espaço vetorial que possui uma métrica é chamado de **espaço métrico**. Daí, a distância de Hamming é também chamada de métrica de Hamming.

2. **CÓDIGOS**

A função primária de um código é permitir a comunicação de informações. A língua portuguesa, com suas palavras e regras para combiná-las é um bom exemplo de código altamente complexo cujo aprendizado é uma tarefa para anos. Essa complexidade é necessária porque a língua exige grande versatilidade, para que ela seja apta à codificação dos mais variados assuntos e estilos: desde textos técnicos científicos, onde se espera o máximo de precisão; aos textos poéticos ou humorísticos, onde propositadamente se explora a polissemia e a ambiguidade.

Por outro lado, um código criado exclusivamente para um assunto mais específico pode e deve ser mais simples, como, por exemplo, as placas de trânsito, que podem ambicionar criar figuras e símbolos de entendimento universal. Entretanto, algumas vezes o código tem como objetivo não ser compreendido por pessoas não autorizadas. Para esse fim, utiliza-se desde o início da civilização as técnicas de criptografia, como o sistema do imperador romano César de permutação de letras.

De acordo com Formiga (2005)

> O primeiro relato de um algoritmo de criptografia que se tem é conhecido como algoritmo de César, usado pelo imperador Júlio César na Roma Antiga. Era um algoritmo simples que fazia substituições alfabéticas no texto da mensagem. As substituições aconteciam trocando letras por outras, três posições à frente no alfabeto, ou seja, a letra A seria substituída por D, a letra B por E, e assim por diante como mostrado a seguir.
>
> Texto simples: Vamos atacar o norte durante a noite.
> Texto cifrado: Zdprv dxdfdu r qruxh gyudqxh d qrlxh.

Códigos eletrônicos

As máquinas de calcular compartilham um modo interessante de codificar os algarismos de um número. Elas se utilizam de um padrão de sete barras, no qual algumas são iluminadas e outras ficam apagadas. Por exemplo, para codificar o algarismo oito, todas as sete barras são acesas.

Figura 1. Visor calculadora

Como cada uma das sete barras admite duas possibilidades (acesa ou apagada), cada espaço no visor de uma máquina de calcular desse tipo pode codificar até $2^7 - 1 = 127$ símbolos. Repare que a situação em que as sete barras estão apagadas não codifica símbolo algum.

Uma estratégia semelhante é usada pelos computadores, nos quais cada *byte* (unidade de armazenamento) é codificado por oito *bits*. Um *bit* é um algarismo binário, ou seja, cujo valor é zero ou um, que podem ser associados com os estados de *desligado* e *ligado*, respectivamente. Assim, cada *byte* é capaz de codificar $2^8 - 1 = 255$ símbolos. (O *byte* 00000000 não é usado).

Desta forma, cada sequência de oito dígitos cada um deles igual a *zero* ou *um*, pode codificar um símbolo, que pode ser uma letra maiúscula, minúscula, algarismo, e outros. A seguir, apresentamos uma parte da tabela ASCII, que permite fazer a conversão entre o conteúdo de um *byte* e seu significado.

Símbolo	ASCII	Símbolo	ASCII	Símbolo	ASCII
espaço	00100000	0	00110000	a	01100001
!	00100001	1	00110001	b	01100010
“	00100010	2	00110010	c	01100011
#	00100011	3	00110011	d	01100100
$	00100100	4	00110100	e	01100101
%	00100101	5	00110101	f	01100110
&	00100110	6	00110110	g	01100111
‘	00100111	7	00110111	h	01101000
(	00101000	8	00111000	i	01101001
)	00101001	9	00111001	j	01101010

Tabela 3. Tabela ASCII .Fonte: Brooksher (2003)

Detecção e correção de erros

Considere o código binário abaixo que controla o braço de um guindaste.

COMANDO	CÓDIGO
DESCER	0
SUBIR	1

Tabela 4. Código guindaste 1

Levando em conta a falta de confiabilidade na transmissão de informações, a aplicação desse código na prática seria no mínimo imprudente. Imagine que haja um erro no envio que troque o valor de zero por um. Isso pode fazer, por exemplo, com que a máquina tente ir além do seu limite técnico, provocando um acidente. Tudo porque o erro na transmissão gera uma palavra-código válida, ou seja, porque o erro não pode ser detectado.

Neste caso, podemos rapidamente vislumbrar uma solução simples: acrescentar mais um dígito pode tornar o código detector de erros. De fato, o código abaixo é detector,

COMANDO	CÓDIGO
DESCER	00
SUBIR	11

Tabela 5. Código guindaste 2

pois um erro simples (alteração de apenas uma componente) de envio no código 00 gera 01 ou 10, que não são códigos válidos. O mesmo ocorre se houver um erro simples no envio do código 11, que pode gerar os códigos 10 ou 01, ambos inválidos.

Entretanto esse código não é corrigível, pois se o código recebido for 01 ou 10, o receptor não poderá identificar qual é a mensagem correta. Portanto, nestes casos, seria necessário programar um protocolo na máquina para solicitar um reenvio do comando.

Por uma questão de eficiência, ou mesmo de necessidade, pode não ser desejável ou possível esperar por esse reenvio. Seria preciso, nestes casos, criar uma maneira de que a própria máquina pudesse ser capaz de corrigir o erro sem auxílio externo, da mesma forma que um editor de texto troca a palavra HIPOPÓTANO por HIPOPÓTAMO.

Surge a necessidade de criar um código não somente detectável, mas também corrigível. Veja que novamente podemos encontrar a solução na redundância. Acrescentar um terceiro dígito ao código torna-o corretor de erros. De fato, o código abaixo é corretor,

COMANDO	CÓDIGO
DESCER	000
SUBIR	111

Tabela 6. Código guindaste 3

pois:

Um erro simples de envio em 000 gera: 001, 010, 100.

Um erro simples de envio em 111 gera: 011, 101, 110.

Neste trabalho estudaremos alguns exemplos de códigos detectores (Capítulo 3) e veremos como criar um código corretor (Capítulos 4 a 6) de Hamming, e porque eles funcionam.

De acordo com Hefez (2008)

> A Teoria dos Códigos Corretores de Erros foi fundada pelo matemático C.E. Shannon, do Laboratório Bell, num trabalho publicado em 1948. Inicialmente, os maiores interessados em Teoria dos Códigos foram os matemáticos que a desenvolveram seriamente nas décadas de 50 e 60. A partir da década de 70, com as pesquisas espaciais e a grande popularização dos computadores, essa teoria começou a interessar também aos engenheiros. Hoje em dia, os códigos corretores de erros são utilizados sempre que se deseja transmitir ou armazenar dados, garantindo a sua confiabilidade. São exemplos disso todas as comunicações via satélite, as comunicações internas de um computador, [...] ou o armazenamento ótico de dados.

De acordo com Poole (2014)

> Richard W. Hamming (1915 – 1988) obteve seu Ph.D em Matemática na Universidade de Illinois, em 1942. [...] De 1946 a 1976, trabalhou no Bell Labs, depois integrou-se ao corpo docente na US Naval Postgraduate School, em Monterey, Califórnia. Em 1950, publicou seu trabalho fundador em códigos corretores de erros, dando uma construção explícita para os códigos de otimização que Claude Shannon tinha provado serem teoricamente possíveis, em 1948[2].

[2]Os artigos originais de Hamming e Shannon podem ser acessados em: http://www.sns.ias.edu/~tlusty/courses/InfoInBio/Papers/Hamming1950.pdf e http://worrydream.com/refs/Shannon%20-%20A%20Mathematical%20Theory%20of%20Communication.pdf, respectivamente.

3. **DETECÇÃO DE ERROS**

Infelizmente, a transmissão de um código não é absolutamente confiável. Do mesmo modo que em uma conversa algumas palavras podem não ser ouvidas ou mal compreendidas pelo interlocutor, também nas comunicações eletrônicas alguns dados se perdem ou são alterados durante a transmissão. Por exemplo, um *zero* pode ser trocado por *um* durante o envio de um byte. Por esse motivo, é útil dispor de um mecanismo capaz de identificar a ocorrência desses erros. Neste capítulo veremos alguns exemplos de códigos criados com essa motivação.

3.1. **Dígito de checagem**

Considere um sistema de direção remota que receba apenas os quatro comandos abaixo. Chamaremos o código abaixo de SDR (Sistema de Direção Remota).

COMANDO	CÓDIGO
CIMA	001
BAIXO	010
ESQUERDA	011
DIREITA	100

Tabela 7. Código SDR 1

Em determinado momento, é possível que o código enviado seja 011 (ESQUERDA), mas que um erro de transmissão troque o primeiro dígito, transmitindo, desta forma, o código incorreto 111. Como esse código não representa nenhum comando é fácil perceber que houve um erro de envio. Entretanto, se o erro tivesse ocorrido no último digito, o código enviado teria sido 010 (BAIXO), que é um código válido, e, portanto, o erro não seria detectável.

Uma maneira de contornar esse tipo de situação é inserir um quarto dígito – que chamaremos de *dígito verificador*, ou *dígito de checagem* – de modo que a soma dos dígitos do código, em $\mathbb{Z}_2$, seja igual a zero. Por exemplo:

COMANDO	CÓDIGO
CIMA	0011
BAIXO	0101
ESQUERDA	0110
DIREITA	1001

Tabela 8. Código SDR 2

Aplicando a aritmética da soma no conjunto $\mathbb{Z}_2$, e a propriedade associativa da adição:

$$\big(((0 + 0) + 1) + 1\big) = \big((0 + 1) + 1\big) = (1 + 1) = 0$$

Analogamente:

$$0 + 1 + 0 + 1 = 0$$
$$0 + 1 + 1 + 0 = 0$$
$$1 + 0 + 0 + 1 = 0$$

Outra maneira de fazer as somas acima é contar a quantidade de dígitos iguais a 1 (um). Se essa quantidade for *par*, a soma será *zero*, se a quantidade for *ímpar*, a soma será igual a *um*. Portanto, em todo código válido o número de dígitos iguais a um é par. E, no caso de ocorrer um erro em apenas um dígito (ou seja, um *erro simples*), será gerado um código inválido, já que a quantidade de dígitos iguais a *um* será ímpar. Ou seja, se for recebido um código como 1011 (erro simples no envio da primeira coordenada de CIMA), então será possível constatar a ocorrência do erro de envio, pois:

$$1 + 0 + 1 + 1 = 1$$

Assim, o dispositivo receptor poderá solicitar ao emissor o reenvio da informação. Observe que o receptor não tem condições de corrigir o erro, pois de seu ponto de vista, mesmo que ele saiba que houve um erro simples, não há como saber em qual das coordenadas ele ocorreu, afinal 1011 poderia ter se originado de um erro simples na terceira coordenada de DIREITA (1001).

Outro problema com esse tipo de solução é que um erro em duas coordenadas (mais geralmente, em um número par de coordenadas) não seria detectável, pois a soma de suas coordenadas seria igual à zero.

Formalizando, o que estamos fazendo é considerar os vetores u e c sobre $\mathbb{Z}_2^4$. O vetor u é o código transmitido, e $c = [1, 1, 1, 1]$ é o vetor de checagem. Em seguida, calculamos o *produto interno* $u \cdot c$, em $\mathbb{Z}_2$. Se o resultado for diferente de zero, isso significa que o código é inválido.

Definição 3.1.1.

Sejam $u = (u_1, u_2, \dots, u_n)$ e $v = (v_1, v_2, \dots, v_n)$ dois vetores em $\mathbb{Z}_m^n$, definimos produto interno de u por v, e indicamos por $u \cdot v$ a soma: $u_1 \cdot v_1 + u_2 \cdot v_2 + \cdots + u_n \cdot v_n$.

Teorema 3.1.2.

Sejam os vetores u e $c = [1, 1, 1, \dots, 1]$ em $\mathbb{Z}_m^n$, e v um vetor obtido trocando exatamente um dígito de u. Então $u \cdot c \neq v \cdot c$.

Demonstração

Fazendo $u = (u_1, u_2, \dots, u_n)$, temos que:

$$u \cdot c = u_1 + u_2 + \cdots + u_n$$

Fazendo $v = (v_1, v_2, \dots, v_n)$, temos que:

$$v \cdot c = v_1 + v_2 + \cdots + v_n$$

Se u e v são idênticos, exceto pela k-ésima coordenada, em símbolos: $u_i = v_i$ para $i = 1, 2, \dots, k-1, k+1, \dots n$ e $u_k \neq v_k$, então:

$$u \cdot c - v \cdot c = (u_1 - v_1) + (u_2 - v_2) + \cdots + (u_k - v_k) + \cdots + (u_n - v_n)$$
$$= 0 + 0 + \cdots + u_k - v_k + \cdots + 0 = u_k - v_k \neq 0$$

Portanto,

$$u \cdot c - v \cdot c \neq 0$$

Implicando que:

$$u \cdot c \neq v \cdot c$$

O Teorema 3.1.2 mostra que o vetor de checagem $c = [1,1,1,\dots,1]$ sempre acusa a ocorrência de erros simples, isto é, em apenas uma coordenada. Nas próximas seções veremos aplicações dessas ideias em códigos usados na prática.

3.2. **UPC: código de barras**

De acordo com Milies (2006), o código UPC (Universal Product Code) foi criado por George J. Laurer da IBM e passou a ser usado nos EUA e Canadá em 1973. Esse código é uma sequência de 12 dígitos decimais (algarismos de 0 a 9), representados, de forma redundante, por números e também por barras. Ou seja, o UPC trabalha com vetores em $\mathbb{Z}_{10}^{12}$.

A regra é parecida com a do nosso SDR (seção 3.1.), mas com vetor de checagem $c = [3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1]$.

Como estamos trabalhando em $\mathbb{Z}_{10}$, usaremos as seguintes tabelas:

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	0
2	2	3	4	5	6	7	8	9	0	1
3	3	4	5	6	7	8	9	0	1	2
4	4	5	6	7	8	9	0	1	2	3
5	5	6	7	8	9	0	1	2	3	4
6	6	7	8	9	0	1	2	3	4	5
7	7	8	9	0	1	2	3	4	5	6
8	8	9	0	1	2	3	4	5	6	7
9	9	0	1	2	3	4	5	6	7	8

Tabela 9. Adição em $\mathbb{Z}_{10}$.

·	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	0	2	4	6	8
3	0	3	6	9	2	5	8	1	4	7
4	0	4	8	2	6	0	4	8	2	6
5	0	5	0	5	0	5	0	5	0	5
6	0	6	2	8	4	0	6	2	8	4
7	0	7	4	1	8	5	2	9	6	3
8	0	8	6	4	2	0	8	6	4	2
9	0	9	8	7	6	5	4	3	2	1

Tabela 10. Multiplicação em $\mathbb{Z}_{10}$.

Exemplo: Mostrar que o código de barras abaixo é válido.

Figura 2. Código de Barras

Solução: $u = (1,8,8,1,1,4,7,7,1,2,1,1)$ é um código de barras válido, pois:

$$u \cdot c = 1 \cdot 3 + 8 \cdot 1 + 8 \cdot 3 + 1 \cdot 1 + 1 \cdot 3 + 4 \cdot 1 + 7 \cdot 3 + 7 \cdot 1 + 1 \cdot 3 + 2 \cdot 1 + 1 \cdot 3 + 1 \cdot 1$$

$$= 3(1 + 8 + 1 + 7 + 1 + 1) + (8 + 1 + 4 + 7 + 2 + 1)$$

$$= 3(9) + 3 = 7 + 3 = 0$$

Exemplo: Calcular o dígito verificador d que torna o código de barras abaixo válido.

Figura 3. Código de Barras

Solução:

$$u \cdot c = 0 \cdot 3 + 2 \cdot 1 + 2 \cdot 3 + 3 \cdot 1 + 3 \cdot 3 + 4 \cdot 1 + 5 \cdot 3 + 4 \cdot 1 + 5 \cdot 3 + 4 \cdot 1 + 5 \cdot 3 + 1 \cdot d$$

$$= 3(0 + 2 + 3 + 5 + 5 + 5) + (2 + 3 + 4 + 4 + 4 + d)$$

$$= 3(0) + 7 + d = 7 + d = 0$$

Portanto, $d = 3$ torna esse código válido.

O teorema a seguir nos garante que o UPC detecta os erros simples.

Teorema 3.2.1.

Sejam os vetores u e $c = [3,1,3,1,3,1,3,1,3,1,3,1]$ em $\mathbb{Z}_{10}^{12}$, e v um vetor obtido trocando exatamente um dígito de u. Então $u \cdot c \neq v \cdot c$.

Demonstração:

Fazendo $u = (u_1, u_2, \dots, u_{12})$, temos que:

$$u \cdot c = 3(u_1 + u_3 + \cdots + u_{11}) + (u_2 + u_4 + \cdots + u_{12})$$

Fazendo $v = (v_1, v_2, \dots, v_{12})$, temos que:

$$v \cdot c = 3(v_1 + v_3 + \cdots + v_{11}) + (v_2 + v_4 + \cdots + v_{12})$$

Se u e v são idênticos, exceto pela k-ésima coordenada, em símbolos: $u_i = v_i$ para $i = 1, 2, \dots, k-1, k+1, \dots 12$ e $u_k \neq v_k$, então:

i) Caso k é impar:

$$u \cdot c - v \cdot c = 3[(u_1 - v_1) + \cdots + (u_k - v_k) + \cdots (u_{11} - v_{11})] + (u_2 - v_2) + \cdots + (u_{12} - v_{12}) = 3(u_k - v_k)$$

Como $u_k - v_k \neq 0$ e 3 é primo com 10, temos que $3(u_k - v_k) \neq 0$. Portanto, $u \cdot c \neq v \cdot c$.

ii) Caso k é par:

$$u \cdot c - v \cdot c = 3[(u_1 - v_1) + \cdots + (u_{11} - v_{11})] + (u_2 - v_2) + \cdots + (u_k - v_k) + \cdots + (u_{12} - v_{12}) = u_k - v_k \neq 0$$

Portanto, $u \cdot c \neq v \cdot c$.

O teorema abaixo nos mostra o comportamento do UPC quando há permuta de dígitos adjacentes (o que pode ocorrer no caso de o código ser digitado por uma pessoa).

Teorema 3.2.2.

Sejam os vetores u e $c = [3,1,3,1,3,1,3,1,3,1,3,1]$ em $\mathbb{Z}_{10}^{12}$, e v um vetor obtido permutando entre si dois dígitos adjacentes de u. Se $u \cdot c = v \cdot c$, então uma das seguintes situações acontece:

a) os dígitos permutados são iguais entre si;

b) a diferença absoluta entre os dígitos permutados é igual a cinco.

Demonstração:

Fazendo $u = (u_1, u_2, \dots, u_{12})$, temos que:

$$u \cdot c = 3(u_1 + u_3 + \cdots + u_{11}) + (u_2 + u_4 + \cdots + u_{12})$$

Fazendo $v = (v_1, v_2, \dots, v_{12})$, temos que:

$$v \cdot c = 3(v_1 + v_3 + \cdots + v_{11}) + (v_2 + v_4 + \cdots + v_{12})$$

Se u e v são idênticos, exceto, possivelmente, pelas k-ésima e $(k+1)$-ésima coordenadas, com k ímpar (sem perda de generalidade) em símbolos: $u_i = v_i$ para $i = 1, 2, \dots, k-1, k+2, \dots 12$ e $v_k = u_{k+1}$, e $v_{k+1} = u_k$ então:

$$u \cdot c - v \cdot c = 3[(u_1 - v_1) + \cdots + (u_k - v_k) + \cdots (u_{11} - v_{11})] + (u_2 - v_2) + \cdots + (u_{k+1} - v_{k+1}) + \cdots + (u_{12} - v_{12}) = 3(u_k - v_k) + (u_{k+1} - v_{k+1}) = 3(u_k - u_{k+1}) + (u_{k+1} - u_k) = 2u_k - 2u_{k+1} = 2(u_k - u_{k+1})$$

Repare que $2(u_k - u_{k+1}) = 0$ somente nos casos:

i) $u_k = u_{k+1}$, ou seja, os dígitos permutados são idênticos (neste caso $u = v$) ou;

ii) $|u_k - u_{k+1}| = 5$, ou seja, a diferença absoluta entre os dígitos permutados é cinco, por exemplo 0 e 5, 1 e 6, 2 e 7, 3 e 8, 4 e 9.

Portanto, o Teorema 3.2.2 nos mostra que não é possível detectar todos os erros de permuta de dígitos adjacentes no código UPC, o que compromete a sua confiabilidade. Para termos uma ideia mais quantitativa do significado disso, vamos resolver um problema.

Problema 3.2.3

Supondo que a escolha dos dígitos em uma palavra no código UPC seja aleatória, qual é a probabilidade de que a diferença entre dois dígitos adjacentes seja igual a cinco?

Solução: Dois dígitos adjacentes (escolhidos entre os algarismos de 0 a 9, podendo haver repetição) podem ser escolhidos de $10 \times 10 = 100$ modos. Dos quais, em dez desses casos a diferença entre os algarismos é cinco (São elas: (0, 5) (1, 6), (2, 7), (3, 8), (4, 9), (5, 0), (6, 1), (7, 2), (8, 3), (9, 4)). Logo, dados dois dígitos adjacentes, a probabilidade de a diferença entre eles seja igual a cinco é $\frac{10}{100} = 10\%$. E dado um código UPC com 12 dígitos, a probabilidade de que haja pelo menos dois adjacentes cuja diferença é igual a cinco é $1 - \left(\frac{9}{10}\right)^{11} \cong 0{,}686$.

Segue abaixo o resultado de um experimento computacional que confirma esses cálculos.

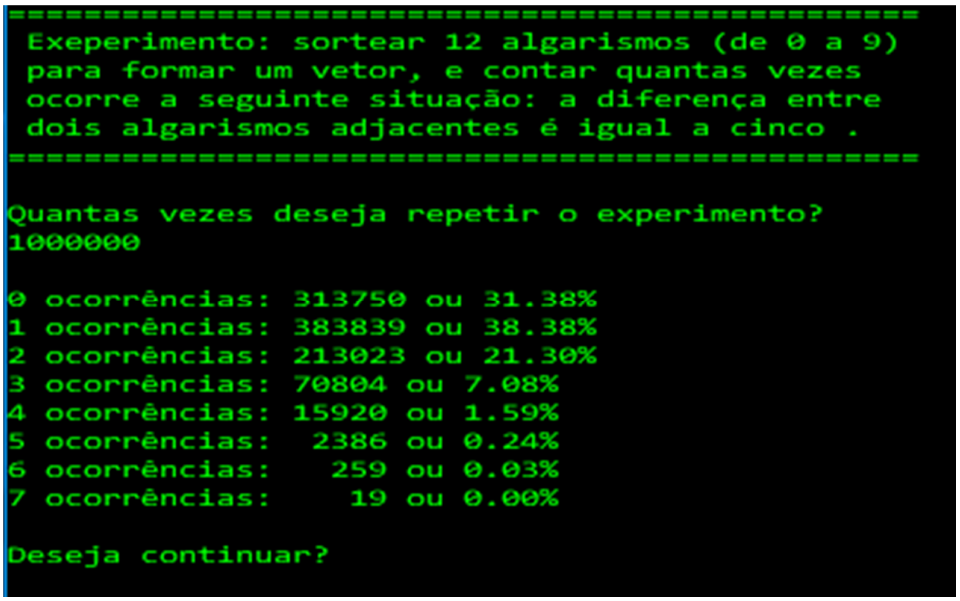

Figura 4. Relatório de processamento

No experimento, o sorteio foi repetido um milhão de vezes, e a situação não ocorreu com frequência 31,38%, ou seja, ocorreu em 68,62% das vezes. Isso não quer dizer, entretanto que o UPC terá 68,62% de probabilidade de falhar, pois o simples fato de haver dígitos cuja diferença seja cinco na palavra-código não significa que eles serão de fato permutadas.

3.3. **ISBN: código de identificação de livros**

O endereço eletrônico do representante oficial do ISBN no Brasil explica que:

> O ISBN – International Standard Book Number – é um sistema internacional padronizado que identifica numericamente os livros segundo o título, o autor, o país, a editora, individualizando-os inclusive por edição.
> Criado em 1967 por editores ingleses, o sistema passou a ser amplamente empregado, tanto pelos comerciantes de livros quanto pelas bibliotecas, até ser oficializado, em 1972, como norma internacional pela International Organization for Standartization - ISO 2108 - 1972.
> O sistema ISBN é controlado pela Agência Internacional do ISBN, que orienta, coordena e delega poderes às agências nacionais designadas em cada país. Desde 1978, a Fundação Biblioteca Nacional representa a Agência Brasileira, com a função de atribuir o número de identificação aos livros editados no país.
> Uma vez fixada a identificação, ela só se aplica àquela obra e edição, não se repetindo jamais em outra.
> <http://www.isbn.bn.br/website/o-que-e-isbn>. Acesso em 04/08/2017

O código ISBN é uma sequência de 10 dígitos, podendo assumir os valores de 0 a 10, neste último caso, substituímos 10 por sua representação em algarismos romanos X para que ocupe um único dígito. Ou seja, o ISBN trabalha com vetores em $\mathbb{Z}_{11}^{10}$, com vetor de checagem $c = [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]$.

Vale a pena aqui fazer um comentário sobre a estratégia de escolha das componentes do vetor de checagem:

- Quando estudamos o UPC, cujos cálculos são feitos em $\mathbb{Z}_{10}$, vimos que as componentes eram 1 ou 3, esses números foram escolhidos por serem primos com 10. Poderíamos criar um código igualmente eficiente do pontos de vista da detecção de erros, usando os números 7 e 9, embora talvez fosse menos eficiente do ponto de vista computacional, porque números números maiores geram contas mais trabalhosas.
- Já no ISBN, cujos cálculos são feitos em $\mathbb{Z}_{11}$, as componentes podem ser escolhidas de modo mais livre, já que todos os valores de 1 a 10 são primos com 11.

Como estamos trabalhando em $\mathbb{Z}_{11}$, usaremos as seguintes tabelas:

+	0	1	2	3	4	5	6	7	8	9	10
0	0	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10	0
2	2	3	4	5	6	7	8	9	10	0	1
3	3	4	5	6	7	8	9	10	0	1	2
4	4	5	6	7	8	9	10	0	1	2	3
5	5	6	7	8	9	10	0	1	2	3	4
6	6	7	8	9	10	0	1	2	3	4	5
7	7	8	9	10	0	1	2	3	4	5	6
8	8	9	10	0	1	2	3	4	5	6	7
9	9	10	0	1	2	3	4	5	6	7	8
10	10	0	1	2	3	4	5	6	7	8	9

Tabela 11. Adição em $\mathbb{Z}_{11}$.

·	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10
2	0	2	4	6	8	10	1	3	5	7	9
3	0	3	6	9	1	4	7	10	2	5	8
4	0	4	8	1	5	9	2	6	10	3	7
5	0	5	10	4	9	3	8	2	7	1	6
6	0	6	1	7	2	8	3	9	4	10	5
7	0	7	3	10	6	2	9	5	1	8	4
8	0	8	5	2	10	7	4	1	9	6	3
9	0	9	7	5	3	1	10	8	6	4	2
10	0	10	9	8	7	6	5	4	3	2	1

Tabela 12. Multiplicação em $\mathbb{Z}_{11}$.

Exemplo: Mostrar que o código ISBN abaixo é válido.

Figura 5. Código de barras

Solução:

$u = (8,1,7,5,2,5,7,6,6,0)$ é um código de barras válido, pois:

$$u \cdot c = 8 \cdot 10 + 1 \cdot 9 + 7 \cdot 8 + 5 \cdot 7 + 2 \cdot 6 + 5 \cdot 5 + 7 \cdot 4 + 6 \cdot 3 + 6 \cdot 2 + 0 \cdot 1$$

$$= 3 + 9 + 1 + 2 + 1 + 3 + 6 + 7 + 1 = 0$$

Recorde que os cálculos são em $\mathbb{Z}_{11}$.

O resultado abaixo nos diz que o ISBN é capaz de detectar erros simples.

Teorema 3.3.1.

Sejam os vetores u e $c = [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]$ em $\mathbb{Z}_{11}^{10}$, e v um vetor obtido trocando exatamente um dígito de u. Então $u \cdot c \neq v \cdot c$.

Demonstração:

Fazendo $u = (u_1, u_2, \dots, u_{10})$, temos que:

$$u \cdot c = 10 \cdot u_1 + 9 \cdot u_2 + 8 \cdot u_3 + 7 \cdot u_4 + 6 \cdot u_5 + 5 \cdot u_6 + 4 \cdot u_7 + 3 \cdot u_8 + 2 \cdot u_9 + 1 \cdot u_{10}$$

Fazendo $v = (v_1, v_2, \dots, v_{10})$, temos que:

$$v \cdot c = 10 \cdot v_1 + 9 \cdot v_2 + 8 \cdot v_3 + 7 \cdot v_4 + 6 \cdot v_5 + 5 \cdot v_6 + 4 \cdot v_7 + 3 \cdot v_8 + 2 \cdot v_9 + 1 \cdot v_{10}$$

Se u e v são idênticos, exceto pela k-ésima coordenada, em símbolos: $u_i = v_i$ para $i = 1, 2, \dots, k-1, k+1, \dots 10$ e $u_k \neq v_k$, então:

$$\begin{aligned} u \cdot c - v \cdot c &= 10(u_1 - v_1) + \cdots + (11-k)(u_k - v_k) + \cdots + 1(u_{10} - v_{10}) \\ &= (11-k)(u_k - v_k) \end{aligned}$$

Mas, $11 - k > 0$, $u_k - v_k \neq 0$ e 11 primo, segue pelo Corolário 1.1.7 que $(11-k)(u_k - v_k) \neq 0$. Logo, $u \cdot c \neq v \cdot c$.

O ISBN também é capaz de detectar erros de permuta de dígitos adjacentes (erros de transposição).

Teorema 3.3.2.

Sejam os vetores u e $c = [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]$ em $\mathbb{Z}_{11}^{10}$, e v um vetor obtido trocando entre si dois dígitos adjacentes distintos de u. Então $u \cdot c \neq v \cdot c$.

Demonstração:

Fazendo $u = (u_1, u_2, \dots, u_{10})$, temos que:

$$u \cdot c = 10 \cdot u_1 + 9 \cdot u_2 + 8 \cdot u_3 + 7 \cdot u_4 + 6 \cdot u_5 + 5 \cdot u_6 + 4 \cdot u_7 + 3 \cdot u_8 + 2 \cdot u_9 + 1 \cdot u_{10}$$

Fazendo $v = (v_1, v_2, \dots, v_{10})$, temos que:

$$v \cdot c = 10 \cdot v_1 + 9 \cdot v_2 + 8 \cdot v_3 + 7 \cdot v_4 + 6 \cdot v_5 + 5 \cdot v_6 + 4 \cdot v_7 + 3 \cdot v_8 + 2 \cdot v_9 + 1 \cdot v_{10}$$

Se u e v são idênticos, exceto pelas k-ésima e $k+1$-ésima coordenadas, em símbolos: $u_i = v_i$ para $i = 1, 2, \dots, k-1, k+2, \dots 10$ e $v_k = u_{k+1}$, e $v_{k+1} = u_k$ então:

$$u \cdot c - v \cdot c = 10(u_1 - v_1) + \cdots + (11-k)(u_k - v_k) + (10-k)(u_{k+1} - v_{k+1}) + \cdots$$
$$+ 1(u_{10} - v_{10}) = (11-k)(u_k - v_k) + (10-k)(v_k - u_k) = u_k - v_k \neq 0$$

Logo, $u \cdot c \neq v \cdot c$.

3.4. **CPF: Cadastro de Pessoas Físicas**

O CPF é uma sequência de 11 dígitos, podendo assumir os valores de 0 a 9, ou seja, o CPF trabalha com vetores em $\mathbb{Z}_{10}^{11}$, com dois vetores de checagem $c_1 = [10, 9, 8, 7, 6, 5, 4, 3, 2]$ e $c_2 = [11, 10, 9, 8, 7, 6, 5, 4, 3, 2]$.

Sejam d_1 e d_2 os dígitos verificadores, ou seja, os dois últimos dígitos do CPF, e u_9 o vetor composto pelos primeiros nove dígitos do CPF. Então, $d_1 = u_9 \cdot c_1$, em $\mathbb{Z}_{10}$.

Seja u_{10} o vetor de dez componentes cujas nove primeiras correspondem a u_9 e última é igual a d_1. Então, $d_2 = u_{10} \cdot c_2$, em $\mathbb{Z}_{10}$.

Da mesma forma que o ISBN, o CPF também corrige erros simples e de transposição.

CÓDIGO	**Detecta erros simples**	**Detecta erros de transposição**	**Corrige erros**
SDR	**SIM**	**NÃO**	**NÃO**
UPC	**SIM**	**SIM***	**NÃO**
ISBN	**SIM**	**SIM**	**NÃO**
CPF	**SIM**	**SIM**	**NÃO**

Tabela 13. Comparação de códigos

*Exceto quando a diferença absoluta entre os algarismos transpostos é igual a cinco.

Vimos que o ISBN tem vantagem de segurança sobre o UPC, pois detecta todos os erros de transposição. Por outro lado, o UPC tem 11 dígitos disponíveis para codificação do produto (mais o dígito de checagem), enquanto o ISBN tem apenas 9 dígitos para isso (mais o dígito de checagem), portanto, o UPC tem vantagem sobre o ISBN na quantidade de produtos possíveis de serem catalogados.

O CPF usa 9 dígitos para codificação (mais dois dígitos verificadores), a principio ele poderia ser trocado por um sistema similar ao ISBN, com a mesma segurança, mas com a vantagem de usar apenas um dígito de checagem. Entretanto, o ISBN tem a desvantagem de usar o X no dígito de checagem, e, tendo em vista que alguns dispositivos não permitem a transmissão do X, por apresentarem apenas teclado numérico (como telefones), nestes casos, é necessário trocar o X por zero, o que compromete a detecção de erros nesses casos. Daí, o fato de o CPF usar apenas algarismos de 0 a 9 é uma vantagem. Por outro lado, o CPF exige dois cálculos para a checagem, tendo, portanto, uma desvantagem computacional.

4. **CORREÇÃO DE ERROS**

No capítulo anterior, apresentamos alguns exemplos de códigos que detectam erros simples, mas que não possuem a capacidade de corrigi-los. Neste capítulo apresentamos uma forma de criar alguns tipos de códigos que possam tanto detectar quanto corrigir alguns erros. Dentre eles os códigos de Hamming.

4.1. **Matriz geradora padrão e matriz de verificação de paridade padrão**

Veremos um algoritmo, devido a Hamming, que permite:
a) transformar as palavras de um código binário em palavras corrigíveis (caso em que ocorre apenas um erro), multiplicando o vetor original por uma *matriz geradora*;
b) identificar o dígito incorreto, multiplicando o vetor recebido por uma *matriz de verificação de paridade*, para em seguida corrigi-lo.

O seguinte exemplo servirá para ilustrar a abordagem desta seção.

Exemplo: Considere a tabela abaixo mostrando o código da fonte ou código de mensagem correspondente a cada comando:

COMANDO	CÓDIGO
CIMA	00
BAIXO	01
ESQUERDA	10
DIREITA	11

Tabela 14. Hamming 1

Vamos criar um código que detecte e corrija um erro ocorrido em apenas uma coordenada. A partir de um código de comprimento k, criaremos um novo código de comprimento n $(n > k)$, acrescentando $n - k$ dígitos ao final de cada código.
Para isso usaremos duas matrizes:

- Uma matriz G da forma $G = \begin{bmatrix} I_k \\ A \end{bmatrix}$ de ordem $n \times k$ que transforme um vetor de mensagem x em $\mathbb{Z}_2^k$ num vetor de código Gx em $\mathbb{Z}_2^n$ chamada de *matriz geradora do código*;
- E uma matriz de checagem $P = [B \quad I_{n-k}]$,, de ordem $(n-k) \times n$, chamada *matriz de verificação de paridade* de modo que para cada $x \in \mathbb{Z}_2^k$, se $c = Gx$, então $Pc = 0$.

No nosso exemplo particular, consideramos: $n = 5$, $k = 2$ e $A = B = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}$. Desta forma,

$$G = \begin{bmatrix} I_2 \\ A \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \text{ e } P = [A \quad I_3] = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}. \text{ Daí,}$$

$$c_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad c_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

$$c_3 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \qquad c_4 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

Onde c_1, c_2, c_3 e c_4 são chamados **vetores de código.**

A tabela abaixo mostra os vetores de código associados a cada comando

COMANDO	CÓDIGO
CIMA	00000
BAIXO	01011
ESQUERDA	10110
DIREITA	11101

Tabela 15. Hamming 2

É fácil verificar que $Pc_i = 0$ para cada $i = 1, 2, 3, 4$. O que confirma que os códigos são válidos. Por exemplo,

$$Pc_1 = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \text{ confirma que } c_1 \text{ é um vetor de código válido.}$$

Por outro lado, se houvesse um erro em alguma componente do vetor de código, digamos $c' = [0, 1, 1, 1, 1]$ com erro na terceira componente de c_2, então

$$\begin{bmatrix}1 & 0 & 1 & 0 & 0\\ 1 & 1 & 0 & 1 & 0\\ 0 & 1 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}0\\1\\1\\1\\1\end{bmatrix} = \begin{bmatrix}1\\0\\0\end{bmatrix}$$ confirma o erro na terceira componente pois o vetor resultante é igual ao vetor da terceira coluna da matriz P e portanto o erro pode ser corrigido.

Agora vamos formalizar estas ideias.

Definição 4.1.1.

Sejam n e k dois inteiros positivos com $k < n$. Toda matriz $n \times k$ da forma $G = \begin{bmatrix}I_k\\A\end{bmatrix}$, em que A é uma matriz de ordem $(n-k) \times k$ sobre $\mathbb{Z}_2$ é chamada de **matriz geradora padrão** para um **código binário** $(\boldsymbol{n}, \boldsymbol{k})$ que transforma vetores de código de $\mathbb{Z}_2^k$ em vetores de código em $\mathbb{Z}_2^n$. Qualquer matriz $(n-k) \times n$ da forma $P = [B \quad I_{n-k}]$, em que B é uma matriz $(n-k) \times k$ sobre $\mathbb{Z}_2$ é chamada de **matriz de verificação de paridade padrão.** Dizemos então que o código tem **comprimento** n e **dimensão** k**.**

As matrizes G e P do exemplo acima obedecem a definição 4.1.1. Observa-se ainda nesse exemplo, que a matriz A que completa a matriz G é igual a matriz B que completa a matriz P. O seguinte resultado diz que esse fato é necessário e suficiente para que P seja uma matriz de verificação de paridade.

Teorema 4.1.2.

Se $G = \begin{bmatrix}I_k\\A\end{bmatrix}$ for uma matriz geradora padrão e $P = [B \quad I_{n-k}]$ for uma matriz de verificação padrão, então

(a) P será a matriz de verificação de paridade associada a G se, e somente se, $A = B$.

(b) O código binário correspondente (n, k) será um corretor de erros (em uma componente) se, e somente se, as colunas de P forem não nulas e distintas.

Demonstração

(a) Seja a_i a i-ésima coluna de A.

Suponha que P e G estão associados ao mesmo código. Isso quer dizer que $PGx = 0$, para todo $x \in \mathbb{Z}_2^k$. Então:

$$Bx + Ax = (B + A)x = (BI + IA)x = [B \quad I]\begin{bmatrix} I \\ A \end{bmatrix} x = PGx = 0$$

Portanto, $Bx + Ax = 0$, ou seja, $Bx = Ax$[3] para todo $x \in \mathbb{Z}_2^k$.

Seja $x = e_i$ o i-ésimo vetor de base canônica de $\mathbb{Z}_2^k$.

Então: $b_i = Be_i = Ae_i = a_i$, para todo i.

Logo, $B = A$.

Reciprocamente, suponha que $B = A$. Então:

$$PGx = [B \quad I]\begin{bmatrix} I \\ B \end{bmatrix} x = \left(B + B^4\right)x = 0x = 0$$

Logo, P e G estão associados ao mesmo código.

(b) Suponha que as colunas de P sejam não nulas e distintas. Se x for um vetor de mensagem em $\mathbb{Z}_2^k$ e $c = Gx$ for o vetor de código correspondente, então: $Pc = 0$.

Se $c' = c + e_i$ é o vetor c com um erro na i-ésima componente, então:

$$Pc' = P(c + e_i) = Pc + Pe_i = 0 + p_i = p_i$$

- Se $p_i = 0$, então um erro na i-ésima componente não será detectado. De fato, se $c' = c + e_i$ é um vetor com erro na i-ésima componente, então $Pc' = 0$.
- Se $p_i = p_j = p \neq 0$, com $i \neq j$, o erro será detectável, mas não corrigível, pois não será possível saber se ele ocorreu na i-ésima ou na j-ésima componente. De fato, se $c' = c + e_i$, ou se $c' = c + e_j$, então $Pc' = p$.

Na verdade, a matriz geradora e a de verificação de paridade são definidas de forma menos restrita do que foi feito acima. Na seção seguinte generalizamos esses conceitos.

4.2. **Matriz geradora e matriz de verificação de paridade**

Para motivar a definição destas matrizes lembre-se primeiramente que a matriz geradora padrão $n \times k$ e a de verificação de paridade padrão de ordem $(n - k) \times n$ para um código são da forma

$$\mathrm{G} = \begin{bmatrix} I_k \\ A \end{bmatrix} \text{ e } P = [B \quad I_{n-k}]$$

[3] Como estamos trabalhando em $\mathbb{Z}_2$, $\boldsymbol{Ax = -Ax}$

[4] Como estamos trabalhando em $\mathbb{Z}_2$, $\boldsymbol{B + B = 0}$

e, portanto pode-se garantir que as colunas de G e as linhas de P são linearmente independentes. Alem disso, pelo Teorema 4.1.2, G e P são associadas ao mesmo código se, e somente se, $A = B$, o que é equivalente a $PG = 0$.

Definição 4.2.1

Para $n > k$, uma matriz G, $n \times k$, e uma matriz P, $(n-k) \times n$ (com elementos em $\mathbb{Z}_2$) são **uma matriz geradora** e **uma matriz de verificação de paridade**, respectivamente, para um código binário $C(n,k)$, se as seguintes condições forem satisfeitas:

i) As colunas de G são linearmente independentes.

ii) As linhas de P são linearmente independentes.

iii) $PG = O$.

Observações

1. Como $\mathbb{Z}_2$ é um corpo, $\mathbb{Z}_2^m$ é um espaço vetorial sobre $\mathbb{Z}_2$ para qualquer inteiro positivo m.
2. Se $v_1, v_2, \ldots, v_k$ são as k colunas de G, então, como
$$0 = PG = P[v_1\ v_2 \ldots . v_k] = [Pv_1\ Pv_2 \ldots . . Pv_k]$$
Tem-se que $Pv_i = 0$ para cada $i = 1, 2, \ldots, k$ e, portanto, cada coluna de G é um vetor de código em C. Além disso, se x é um código de mensagem em $\mathbb{Z}_2^k$, $Gx = v_1x + v_2x + \cdots v_kx$. Daí, C é gerado pelos vetores colunas de G. Isto é, C é um espaço coluna de G.
3. Sabe-se que o espaço linha de uma matriz não muda com as operações elementares linha na matriz e analogamente, o espaço coluna de uma matriz não muda com as operações elementares coluna na matriz.
4. Para uma matriz com entradas em $\mathbb{Z}_2$ existem apenas duas operações elementares coluna: permutar duas colunas (C1) e somar uma coluna a outra (C2). E existem duas operações elementares linha: permutar duas linhas (L1) e somar uma linha a outra (L2).
5. Se E é uma matriz elementar, P uma matriz de verificação de paridade e c é um vetor de código, então $(EP)c = E(Pc) = E0 = 0$.
Portanto, EP também é uma matriz de verificação de paridade para C.

Essas observações mostram que existem muitas matrizes geradoras e muitas matrizes de verificação de paridade diferentes para um mesmo código.

Nem sempre é possível trazer a matriz geradora ou de verificação de paridade à forma padrão, usando apenas operações elementares coluna em G e operações elementares linha em P. Veremos que para garantir essa possibilidade precisaremos acrescentar uma operação elementar linha na matriz G e uma operação elementar coluna em P.

Mas antes de enunciar esse resultado precisamos de outra definição.

Definição 4.2.2.

Dizemos que dois códigos C_1 e C_2 são **equivalentes** quando existe uma matriz de permutação M tal que

$$\{Mx: x \text{ em } C_1\} = C_2$$

Exemplo. Considere o código $C_1 = \{u_1, u_2, u_3, u_4\}$, onde $u_1 = \begin{bmatrix}0\\0\\0\end{bmatrix}, u_2 = \begin{bmatrix}1\\0\\0\end{bmatrix}, u_3 = \begin{bmatrix}1\\0\\1\end{bmatrix}$ e $u_4 = \begin{bmatrix}1\\1\\1\end{bmatrix}$. E a matriz $M = \begin{bmatrix}0 & 1 & 0\\1 & 0 & 0\\0 & 0 & 1\end{bmatrix}$. O efeito da matriz M sobre um vetor $v = \begin{bmatrix}a\\b\\c\end{bmatrix}$ é permutar entre si a primeira e a segunda componentes. De fato:

$$Mv = \begin{bmatrix}0 & 1 & 0\\1 & 0 & 0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}a\\b\\c\end{bmatrix} = \begin{bmatrix}b\\a\\c\end{bmatrix}$$

Repare que a matriz de permutação deve ser quadrada e ter exatamente um elemento igual a 1 em cada linha e em cada coluna. Os outros elementos devem ser nulos.

Portanto, $C_2 = \{v_1, v_2, v_3, v_4,\} = \{Mu_1, Mu_2, Mu_3, Mu_4\} = \left\{\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\1\\1\end{bmatrix}\right\}$, o que nos mostra que os códigos C_1 e C_2 são equivalentes.

Exemplo: $C_1 = \left\{\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\1\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}\right\}$ e $C_2 = \left\{\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}\right\}$

São equivalentes pela matriz de permutação $M = \begin{bmatrix}0 & 0 & 1\\1 & 0 & 0\\0 & 1 & 0\end{bmatrix}$.

Das observações anteriores mais o fato de que permutar os elementos dos vetores de código corresponde a permutar linhas da matriz geradora e a permutar colunas da matriz de verificação, segue o seguinte resultado.

Teorema 4.2.3

(a) Pode-se trazer à forma padrão qualquer matriz geradora para um código por intermédio das operações C1, C2 e L1. Se L1 não for usada, tem-se o mesmo código; se L1 for usada, se obtém um código equivalente.

(b) Pode-se trazer à forma padrão qualquer matriz de verificação de paridade para um código por meio das operações L1, L2 e C1. Se C1 não for usada, obtém-se o mesmo código; se C1 for usada, tem-se um código equivalente.

Vejamos nos próximos exemplos como isso pode ser feito.

Exemplo: Trazer a matriz geradora $G = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$ para a forma padrão e encontrar a matriz de verificação de paridade correspondente.

Solução: Basta usar a operação L1 em G:

$$G = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \xrightarrow[\ell_1 \leftrightarrow \ell_2]{} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} I \\ A \end{bmatrix} = G'$$

Temos que $A = [1 \quad 0]$, e, portanto,

$$P = [A \quad I] = [1 \quad 0 \quad 1]$$

Seja C_1 o código gerado por G, então:

$$\begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} a \\ a \\ b \end{bmatrix}$$

$$C_1 = \left\{ \begin{bmatrix} a \\ a \\ b \end{bmatrix} : a, b \in \mathbb{Z}_2 \right\} \left\{ a \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} + b \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} : a, b \in \mathbb{Z}_2 \right\} = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\}$$

Seja C_2 o código gerado por G', então:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} a \\ b \\ a \end{bmatrix}$$

$$C_1 = \left\{ \begin{bmatrix} a \\ b \\ a \end{bmatrix} : a, b \in \mathbb{Z}_2 \right\} \left\{ a \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} + b \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} : a, b \in \mathbb{Z}_2 \right\} = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\}$$

Repare que C_1 e C_2 são códigos equivalentes. Eles admitem a seguinte matriz de permutação:

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Essa matriz tem o efeito de permutar a segunda e terceira componentes.

Exemplo: Trazer a matriz de verificação de paridade $P = [1 \quad 1 \quad 0]$ para a forma padrão e encontrar a matriz geradora correspondente.

Solução: Basta usar a operação C1 em P:

$$P = [1 \quad 1 \quad 0] \xrightarrow[c_2 \leftrightarrow c_3]{} [1 \quad 0 \quad 1] = [A' \quad I] = P'$$

E neste caso, a matriz geradora é $G' = \begin{bmatrix} I \\ A' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$

Ou então, poderíamos fazer:

$$P = [1 \quad 1 \quad 0] \xrightarrow[c_1 \leftrightarrow c_3]{} [0 \quad 1 \quad 1] = [A'' \quad I] = P''$$

E neste caso, a matriz geradora é $G'' = \begin{bmatrix} I \\ A'' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$

Comparando as duas possíveis matrizes geradoras, verificamos que elas geram códigos equivalentes.

$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} a \\ b \\ a \end{bmatrix}$, que gera o código $C_1 = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\}$.

$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} a \\ b \\ b \end{bmatrix}$, que gera o código $C_2 = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\}$.

De fato, a matriz de permutação entre eles é $M = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Exemplo: Trazer a matriz de verificação de paridade $P = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$ para a forma padrão e encontrar a matriz geradora correspondente.

Solução: Basta usar a operação C1 em P:

$$P = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} \xrightarrow[c_2 \leftrightarrow c_3]{c_1 \leftrightarrow c_4} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} = [A' \quad I] = P'$$

E neste caso, a matriz geradora é $G' = \begin{bmatrix} I \\ A' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{bmatrix}$.

Observe que neste último exemplo P não corrige erros, pois possui uma coluna nula (Teorema 4.1.2).

4.3. **Códigos lineares**

Definição 4.3.1

Seja p um número primo. Um código linear p-ário é um subespaço C[5] de $\mathbb{Z}_p^n$. Em particular, para $p = 2$, dizemos que o código linear é binário.

Exemplo: $C_1 = \left\{\begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}\right\}$ não é subespaço de $\mathbb{Z}_2^2$, pois não contém o vetor nulo $\begin{bmatrix}0\\0\end{bmatrix}$, portanto, C_1 não é um código linear.

Também o código $C_2 = \left\{\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\1\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}\right\}$ não é linear, porque não é um subespaço de $\mathbb{Z}_2^3$.

De fato, apesar de conter o vetor nulo, C_2 não é fechado para a adição, pois:

$$\begin{bmatrix}1\\0\\1\end{bmatrix} + \begin{bmatrix}1\\1\\0\end{bmatrix} = \begin{bmatrix}0\\1\\1\end{bmatrix} \notin C_2$$

Finalmente, $C = \left\{\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}\right\}$ é um subespaço de $\mathbb{Z}_2^3$, pois contém o vetor nulo, e, além disso, para todo u e v em C, temos que $u + v$ está em C. De fato,

i) Para $u = 0, u + v = v \in C$.

ii) Para $u = v, u + v = 0 \in C$.

iii) Quanto aos outros casos, os cálculos mostram que:

$\begin{bmatrix}0\\1\\0\end{bmatrix} + \begin{bmatrix}0\\1\\1\end{bmatrix} = \begin{bmatrix}0\\0\\1\end{bmatrix} \in C$, $\begin{bmatrix}0\\1\\0\end{bmatrix} + \begin{bmatrix}0\\0\\1\end{bmatrix} = \begin{bmatrix}0\\1\\1\end{bmatrix} \in C$ e $\begin{bmatrix}0\\1\\1\end{bmatrix} + \begin{bmatrix}0\\0\\1\end{bmatrix} = \begin{bmatrix}0\\1\\0\end{bmatrix} \in C$

Portanto, C é um código linear.

Se um código linear binário C for um subespaço de $\mathbb{Z}_2^n$, de dimensão k, então diremos que C é um **código (n, k).**

[5] $\mathbb{Z}_p$ é o oconjunto de escalares

4.4. **Códigos duais e autoduais**

Definição 4.4.1

Seja C um conjunto de vetores de código em $\mathbb{Z}_2^n$. O complemento ortogonal de C, chamado código dual de C, é denotado por $C^\perp$ e definido como segue:

$$C^\perp = \{x \in \mathbb{Z}_2^n : c \cdot x = 0, \forall c \in C\}$$

Exemplo: Considere $C = \left\{\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}\right\}$. O conjunto de todos os vetores em $\mathbb{Z}_2^2$ que são ortogonais a C é $C^\perp = \left\{\begin{bmatrix}0\\0\end{bmatrix}\right\}$.

Teorema 4.4.2.

Se C é um código binário (n, k) com matriz geradora G e matriz de verificação de paridade P, então $C^\perp$ é um código binário $(n, n-k)$ tal que:

i) G^T é uma matriz de verificação de paridade de $C^\perp$.

ii) P^T é uma matriz geradora para $C^\perp$.

Demonstração: Como G é uma matriz $n \times k$ com colunas linearmente independentes e P é uma matriz $(n-k) \times n$ com linhas linearmente independentes e $PG = 0$. Logo, as linhas de G^T e as colunas de P^T são linearmente independentes, e

$$G^T P^T = (PG)^T = O^T = O$$

Isso mostra que G^T é a matriz de verificação de paridade para $C^\perp$ e P^T é uma matriz geradora para $C^\perp$. Como $P^\perp$ é $n \times (n-k)$, $C^\perp$ é um código $(n, n-k)$.

Teorema 4.4.3.

Seja C um código linear (n, k). Pode-se afirmar que:

i) O código dual $C^\perp$ é um código linear $(n, n-k)$.

ii) C contém 2^k vetores, e $C^\perp$ contém 2^{n-k} vetores.

Demonstração:

i) Como C é um código linear (n, k), ele é um subespaço de dimensão k de $\mathbb{Z}_2^n$. Seu dual é o complemento ortogonal de C e é também um subespaço de $\mathbb{Z}_2^n$. Portanto, $C^\perp$ é um código linear. Para finalizar, basta observar que:

$$\dim C^\perp = n - \dim C = n - k$$

ii) Seja $\{v_1, v_2, \dots, v_k\}$ uma base para C. Então, os elementos de C são da forma

$$v = c_1 v_1 + c_2 v_2 + \cdots + c_k v_k$$

Onde cada c_i é zero ou um. Portanto, existem duas possibilidades para c_1, e para cada uma delas, duas possibilidades para c_2, e assim por diante. Portanto, são 2^k possíveis vetores.
O mesmo argumento mostra que o seu dual $(n, n-k)$ tem 2^{n-k} vetores.

Definição 4.4.4

Um código com a propriedade $C^{\perp} = C$ é chamado **autodual**.

Exemplo: Considere $C = \left\{\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}\right\}$. O conjunto de todos os vetores em $\mathbb{Z}_2^2$, que são ortogonais a C é $C^{\perp} = \left\{\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}\right\}$. Portanto, C é autodual.

Teorema 4.4.5

Seja C um código linear. O dual do dual de C é o próprio C, em símbolos: $(C^{\perp})^{\perp} = C$.

Demonstração: Suponha que $x \in C$. Para todo $y \in C^{\perp}$, temos que $x \cdot y = 0$. Como $(C^{\perp})^{\perp}$ contém todos os vetores ortogonais a y, então também contém o vetor x, ou seja, $x \in (C^{\perp})^{\perp}$. Portanto, $C \subset (C^{\perp})^{\perp}$.
Suponha que $x \in (C^{\perp})^{\perp}$. Para todo $y \in C^{\perp}$, temos que $x \cdot y = 0$. Como C contém todos os vetores ortogonais a y, então também contém o vetor x, ou seja, $x \in C$. Portanto, $(C^{\perp})^{\perp} \subset C$.
Logo, vale a igualdade: $C = (C^{\perp})^{\perp}$.

Corolário 4.4.6

Se C_1 e C_2 são códigos lineares (n, k) tais que $(C_1)^{\perp} = C_2$, então $(C_2)^{\perp} = C_1$.

Demonstração Se $(C_1)^{\perp} = C_2$, então $((C_1)^{\perp})^{\perp} = (C_2)^{\perp}$. Do teorema anterior, $((C_1)^{\perp})^{\perp} = C_1$. Logo, $C_1 = (C_2)^{\perp}$.

Definição 4.4.7

Seja x um vetor em $\mathbb{Z}_2^n$. O peso de x, denotado por $w(x)$ é o números de 1s em x.

Exemplo: $w(1,1,0,1) = 3$ e $w(1,0,0,1,0) = 2$.

Exemplo: Seja E_n o código binário formado por todos os vetores em $\mathbb{Z}_2^n$ que tem peso par.
Seja Rep_n o código binário com n componentes que possui apenas dois vetores: 0 e 1 (onde 1 é o vetor com todas as componentes iguais a 1).

Mostre que E_n e Rep_n são duais um do outro.

Solução: Vamos pensar primeiramente no caso $n = 3$, para em seguida generalizar.

$$E_3 = \left\{\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\0\\1\end{bmatrix}\right\} \text{ e } Rep_3 = \left\{\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\1\\1\end{bmatrix}\right\}$$

Para mostrar que esses códigos são duais, podemos verificar que:

a) cada elemento de Rep_3 é ortogonal a cada um dos elementos de E_3.

De fato,

$$\begin{bmatrix}0\\0\\0\end{bmatrix} \cdot v = 0, \text{ para todo vetor } v \text{ em } \mathbb{Z}_2^3.$$

E ainda,

$$\begin{bmatrix}1\\1\\1\end{bmatrix} \cdot \begin{bmatrix}1\\1\\0\end{bmatrix} = \begin{bmatrix}1\\1\\1\end{bmatrix} \cdot \begin{bmatrix}0\\1\\1\end{bmatrix} = \begin{bmatrix}1\\1\\1\end{bmatrix} \cdot \begin{bmatrix}1\\0\\1\end{bmatrix} = 0.$$

b) não existe outro elemento em $\mathbb{Z}_2^3$ ortogonal a cada elemento de E_3.

Podemos proceder por tentativas, verificando para cada um dos vetores do conjunto $\left\{\begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\0\\1\end{bmatrix}\right\}$ que nenhum deles cumpre a condição.

Outra maneira é encontrar o vetor $x = \begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}$ que é o ortogonal a todos os vetores de E_3, resolvendo o sistema abaixo:

$$x \cdot \begin{bmatrix}1\\1\\0\end{bmatrix} = x \cdot \begin{bmatrix}0\\1\\1\end{bmatrix} = x \cdot \begin{bmatrix}1\\0\\1\end{bmatrix} = 0$$

$$\begin{cases} x_1 + x_2 = 0 \\ + x_2 + x_3 = 0 \\ x_1 + + x_3 = 0 \end{cases}$$

$$\left[\begin{array}{ccc|c}1&1&0&0\\0&1&1&0\\1&0&1&0\end{array}\right] \xrightarrow[\ell_3 \leftarrow \ell_3 + \ell_1]{} \left[\begin{array}{ccc|c}1&1&0&0\\0&1&1&0\\0&1&1&0\end{array}\right] \xrightarrow[\ell_3 \leftarrow \ell_3 + \ell_2]{} \left[\begin{array}{ccc|c}1&1&0&0\\0&1&1&0\\0&0&0&0\end{array}\right]$$

$$\begin{cases} x_1 + x_2 = 0 \\ x_2 + x_3 = 0 \end{cases} \Rightarrow \begin{cases} x_1 = x_2 \\ x_2 = x_3 \end{cases} \Rightarrow x_1 = x_2 = x_3$$

Portanto,

$$\left\{ x_1 \begin{bmatrix}1\\1\\1\end{bmatrix} \middle| x_1 \in \mathbb{Z}_2 \right\} = \left\{\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\1\\1\end{bmatrix}\right\} = Rep_3$$

Portanto, a afirmação é válida para $n = 3$.

No caso geral, observe primeiramente que

$$Rep_n = \left\{ \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \right\}$$

é um subespaço de $\mathbb{Z}_2^n$.

Continua verdadeiro que $\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ é ortogonal a qualquer vetor de $\mathbb{Z}_2^n$ e em particular, qualquer vetor de E_n. Basta verificar que $\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$ também o é. Para isso, vamos considerar um vetor genérico $x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ em $\mathbb{Z}_2^n$. Então:

$$\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = x_1 + x_2 + \cdots + x_n$$

A expressão acima é nula se, e somente se existe uma quantidade par de 1s entre as componentes $x_1, x_2, \ldots, x_n$, Ou seja, se e somente se $x \in E_n$.

Teorema 4.4.8

Se $C(n, k)$ é um código auto dual, com vetores em $\mathbb{Z}_2^n$, então

i) Todo vetor em C tem peso par

ii) 1 está em C.

iii) n é par

Demonstração

i) Suponha que o vetor $x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in C$ tenha peso ímpar, então $x_1 + x_2 + \cdots + x_n = 1$, e, portanto, $x \cdot x = 1$. Mas isso não pode ocorrer, pois como C é auto dual, x deve ser ortogonal a si mesmo, ou seja, $x \cdot x = 0$. Portanto, o peso de x não é ímpar, logo, é par.

ii) O vetor $1 = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$ é tal que $1 \cdot x = 0$, para qualquer $x \in C$ (do item anterior, já sabemos que x tem peso par). E como todo elemento de $\mathbb{Z}_2^n$ que é ortogonal a todos os elementos de C, também pertence a C, concluímos que $1 \in C$.

iii) Dos fatos: todo elemento de C tem peso par (item i), e $1 \in C$ (item ii), concluímos que $1 = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$ tem peso par, ou seja, a quantidade de coordenadas, n, é necessariamente par.

Teorema 4.4.9

Seja C um código binário (n, k) com matriz geradora G e matriz de verificação de paridade P. Se C é autodual, então $n = 2k$, G é de ordem $(2k) \times k$ e $P = G^T$ é uma matriz de verificação de paridade para C.

Demonstração Se C é um código binário (n, k) então, pelo Teorema 4.4.2., $C^{\perp}$ é um código binário $(n, n-k)$. Mas se $C = C^{\perp}$ então $k = n - k$ e portanto $n = 2k$.

Além disso, ainda pelo Teorema 4.4.2. P^T é a matriz geradora de $C^{\perp}$, mas como $C^{\perp} = C$, então $P = G^T$, onde G é uma matriz de ordem $(2k) \times k$ e obviamente P é de ordem $k \times (2k)$.

5. **MÉTRICA DE HAMMING**

Nosso objetivo principal neste capítulo é estabelecer um resultado que permita determinar quantos erros o código pode detectar e quantos erros podem ser corrigidos. Para mais detalhes veja a seção 1.3.

5.1. **Distância de Hamming**

Definição 5.1.1

Dados dois elementos $u = (u_1, u_2, \dots, u_n)$ e $v = (v_1, v_2, \dots, v_n)$ em $\mathbb{Z}_2^n$, a distância de Hamming entre u e v é definida por

$$d_H(u, v) = \|u + v\|_H = w(u + v)$$

onde $w(u + v)$ é o peso do vetor $u + v$

Exemplo: Encontre a distância de Hamming entre os vetores $u = (1,0,1,0,1)$ e $v = (1,0,0,1,1)$ em $\mathbb{Z}_2^5$.

Solução: Como $u + v = (0,0,1,1,0)$, então

$$d_H(u, v) = \|u + v\|_H = \|(0,0,1,1,0)\|_H = w(0,0,1,1,0) = 2$$

Toda função distância definida a partir de uma norma num espaço vetorial V satisfaz às seguintes propriedades:

Teorema 5.1.2

Para todos os vetores u, v, w em V valem:

a) $d(u, v) \geq 0$ e $d(u, v) = 0$ se, e somente se, $u = v$ (positividade)

b) $d(u, v) = d(v, u)$ (simetria)

c) $d(u, v) \leq d(u, w) + d(w, v)$ (desigualdade triangular)

Em particular, para o espaço vetorial $\mathbb{Z}_2^n$, se d é a distância d_H de Hamming e $\|u\|$ é a norma $\|u\|_H$ de Hamming, o teorema anterior também vale e, portanto, as propriedades acima permitem dizer que d_H é uma métrica em $\mathbb{Z}_2^n$, chamada **métrica de Hamming**.

Introduziremos a seguir a noção de distância nos códigos binários.

Definição 5.1.3

Seja C um código binário, a distância mínima de C é a menor distância de Hamming entre quaisquer dois vetores distintos de C. Isto é,

$$d(C) = \min\{d_H(x,y): x \neq y \text{ em } C\}$$

Exemplo: Encontre a distância mínima do código $C = \{c_0, c_1, c_2, c_3\}$ onde $c_0 = \begin{bmatrix}0\\0\\0\\0\\0\end{bmatrix}$, $c_1 = \begin{bmatrix}0\\1\\0\\1\\1\end{bmatrix}$, $c_2 = \begin{bmatrix}1\\0\\1\\1\\0\end{bmatrix}$ e $c_3 = \begin{bmatrix}1\\1\\1\\0\\1\end{bmatrix}$

Solução: Precisamos calcular a distância de Hamming entre os $\binom{4}{2} = 6$ pares de vetores:

$d_H(c_0, c_1) = 3$ $\quad$ $d_H(c_0, c_2) = 3$ $\quad$ $d_H(c_0, c_3) = 4$

$d_H(c_1, c_2) = 4$ $\quad$ $d_H(c_1, c_3) = 3$ $\quad$ $d_H(c_2, c_3) = 3$

Logo,

$$d(C) = \min\{3,3,4,4,3,3\} = \min\{3,4\} = 3$$

Em geral, para calcular $d(C)$ num código C de M vetores – por este método – é necessário calcular $\binom{M}{2}$ distâncias, o que é inviável para M muito grande. Veremos adiante um método mais eficiente.

O seguinte resultado mostra que em códigos lineares com M elementos, pode-se calcular $d(C)$ apenas com $M - 1$ cálculos de distâncias em vez dos $\binom{M}{2}$ cálculos anteriormente requeridos.

Proposição 5.1.4

Se $C \subset \mathbb{Z}_2^n$ é um código linear com distância mínima d, então

$$d = w(C) \coloneqq \min\{w(x): x \in C \setminus \{0\}\}$$

onde $w(C)$ é o peso mínimo de C ou simplesmente o peso de C.

Demonstração Basta provar que os conjuntos finitos $A = \{d_H(x,y): x \neq y \in C\}$ e $B = \{w(z): z \in C \setminus \{0\}\}$ são iguais, pois se $A = B$, $d(C) = \min A = \min B = w(C)$ e tem-se o resultado.

Para fazer isto, considere $w(z) \in B$ com z arbitrário em $C\backslash\{0\}$. Como $w(z) = d_H(z,0)$, com $z \neq 0$ segue que $w(z) \in A$ e, portanto $B \subset A$.

Reciprocamente, seja $d_H(x,y) \in A$. Como $x + y \in C$, com $x \neq y$, e C é um espaço vetorial tem-se que $z = x + y \in C\backslash\{0\}$ e, portanto $d_H(x,y) = w(x+y) = w(z)$ com $z \neq 0$. Logo, $d_H(x,y) \in B$ e, portanto $A \subset B$.

Agora estamos em condições de refazer o exemplo anterior com menos esforço. Primeiramente, observe que $C = \{c_0, c_1, c_2, c_3\}$ como definido anteriormente é um subespaço de $\mathbb{Z}_2^5$ e, portanto, trata-se de um código linear. Vimos que $d(C) = \min\{3,3,4,4,3,3\} = \min\{3,4\} = 3$. Agora vamos calcular $w(C) = \min\{w(c_1), w(c_2), w(c_3)\}$:

$$w(c_1) = w(0,1,0,1,1) = 3$$
$$w(c_2) = w(1,0,1,1,0) = 3$$
$$w(c_3) = w(1,1,1,0,1) = 4$$

Daí, $d(C) = w(C) = \min\{3,3,4\} = \min\{3,4\} = 3$.

5.2. **Códigos de correção de erros**

Quando digitamos a palavra ELEFANTR em um editor de textos, é possível que ele faça a correção, se for programado para fazer duas suposições:

- Nossa intenção foi escrever uma palavra em português, e não em outra língua, nem uma sigla;
- A quantidade de erros que cometemos na digitação foi a menor possível.

Deste modo, o editor pode fazer a correção para ELEFANTE.

Por outro lado, se a palavra fosse ELETANTE o programa enfrentaria maiores dificuldades. A palavra correta seria ELEFANTE ou ELEGANTE? O fato de existir duas palavras como elefante e elegante cuja distância (número de componentes diferentes) é igual a 1, não permite que a correção seja feita sempre.

Se a palavra fosse MOLO, novamente seria difícil decidir. A palavra correta seria BOLO ou MOLA, por exemplo. O fato de a distância entre bolo e mola ser igual a 2 gera essa dificuldade.

De maneira geral, para que seja sempre possível corrigir um erro, é necessário que a distância mínima entre duas palavras seja maior ou igual a 3 Isso garante que haja uma única palavra mais próxima da palavra com um erro, que é justamente a palavra correta.

É exatamente desta forma que o código de Hamming funciona, como veremos nesta seção e no capítulo seguinte.

Definição 5.2.1

Dado um elemento $x \in \mathbb{Z}_2^n$ e um número real $t > 0$, o disco e a esfera de centro em x e raio t são respectivamente os conjuntos

$$D(x,t) = \{y \in \mathbb{Z}_2^n : d(x,y) \leq t\}$$
$$S(x,t) = \{y \in \mathbb{Z}_2^n : d(y,x) = t\}$$

Definição 5.2.2

Para cada vetor código $x \in C$ e um número real $t > 0$, o disco (de Hamming) com centro x e raio t será o conjunto

$$D_H(x,t) = \{y \in C : d_H(y,x) \leq t\}$$

Observe inicialmente que se $c \in C$, $d_H(b,c) \geq d$ para $c \neq b \in C$, e, portanto:

- Se $t < d$, c é o único vetor código no disco $D_H(c,t)$
- Se $d < 2t$, $D_H(c,t) \cap D_H(b,t) = \emptyset$.

Para que C seja um código (binário) corretor de κ erros é usado o processo conhecido como "decodificador pelo vizinho mais próximo". Para isso deve-se encontrar um raio κ apropriado tal que todos os discos centrados em vetores de código não se interceptem, e, portanto, ao receber c' obtido por até κ trocas nas entradas de $c \in C$, ele se encontrará apenas no disco $D_H(c,\kappa)$ e, portanto c' será decodificado por c, pois c será seu vizinho mais próximo (dentre todos os vetores código). Agora vamos formalizar esses conceitos.

Definição 5.2.3

Seja C um código (binário). Para um vetor $c \in C$, seja c' um vetor obtido por até κ trocas nas entradas de c. Dizemos que:

a) C detecta κ erros se c' não é um vetor código.

b) C corrige κ erros se o vizinho decodificado mais próximo de c' produz c.

Dado um código C com distância mínima d, define-se $\kappa = \left[\frac{d-1}{2}\right]$, onde $[t]$ representa a parte inteira de um número real t, isto é, $[t]$ é o maior inteiro menor do que ou igual a t.

Lema 5.2.4

Seja C um código com distância mínima d. Se b e c são vetores código distintos de C, então

$$D(b,\kappa) \cap D(c,\kappa) = \emptyset$$

Demonstração: Suponha por absurdo que $D(b,\kappa) \cap D(c,\kappa) \neq \emptyset$ e considere $x \in D(b,k) \cap D(c,\kappa)$. Logo $d_H(x,b) \leq \kappa$ e $d_H(x,c) \leq \kappa$ e, portanto, pela simetria e pela desigualdade triangular

$$d(b,c) \leq d(b,x) + d(x,c) \leq 2\kappa$$

Da definição de κ tem-se que $\kappa \leq \frac{d-1}{2}$ e, portanto, $2\kappa \leq d-1$.

Logo, $d(b,c) \leq d-1$ o que é uma contradição, pois $d_H(b,c) \geq d$.

O resultado mais importante deste capítulo é o seguinte.

Teorema 5.2.5

Seja C um código com distância mínima d. Então, C pode corrigir até $\kappa = \left[\frac{d-1}{2}\right]$ erros e detectar até $d-1$ erros.

Demonstração Seja c um vetor código em C. Se ao transmitirmos c é recebido o vetor c' com t erros nas componentes de c, com $t \leq \kappa$, então $d_H(c',c) = t \leq \kappa$ e, portanto $c' \in D_H(c,\kappa)$. Pelo Lema 5.2.4 segue que para cada $b \in C$ com $b \neq c$, $c' \notin D_H(b,\kappa)$ e, portanto c' é decodificado pelo vizinho mais próximo, que é c.

Por outro lado, dado um vetor código $c \in C$, se são introduzidos $d-1$ erros nas suas coordenadas, recebendo o vetor c', então $d_H(c,c') = d-1$ e, portanto $c' \notin C$ pois, caso contrário, teríamos $d_H(c,c') \geq d$.

Por exemplo, vimos que o código C apresentado após a definição 5.1.3 é tal que $d = 3$. Logo, nele é possível corrigir até $\kappa = \left[\frac{3-1}{2}\right] = 1$ erro e detectar até $3-1=2$ erros.

O teorema anterior ressalta a importância de se conhecer a distância mínima de um código. No caso de códigos lineares a proposição 5.1.4 facilita essa tarefa. Também é possível determinar $d(C)$ examinando a matriz de verificação de paridade de C.

Teorema 5.2.6

Seja C um código linear (n, k) com matriz de verificação de paridade P. Então a distância mínima de C é o menor inteiro d para o qual existem d colunas linearmente dependentes.

Demonstração Suponha que $d(C) = d$. Se C é um código linear (n, k) com matriz de verificação de paridade P, então P é de ordem $(n-k) \times n$ e $Px = 0$, $\forall x \in C \subset \mathbb{Z}_2^n$. Pela Proposição 5.1.4, C contém um vetor c de peso d. Então Pc é uma combinação linear de exatamente d colunas de P. Mas como $Pc = 0$ tem-se que existe um subconjunto com d colunas de P que é linearmente dependente. A seguir provaremos que d é o número mínimo de colunas linearmente dependentes na matriz P.

De fato, suponha por absurdo que existam t, com $t < d$, colunas linearmente dependentes, digamos que

$$P_{i_1}, P_{i_2}, \dots, P_{i_t}$$

sejam t colunas linearmente dependentes dentre as colunas de P. Logo, existem escalares não todos nulos $\alpha_{i_j} \in \mathbb{Z}_2$ tais que

$$\alpha_{i_1} P_{i_1} + \alpha_{i_2} P_{i_2} + \cdots + \alpha_{i_t} P_{i_t} = 0$$

Pela minimalidade de t segue que esses escalares são todos iguais a 1 e portanto

$$P_{i_1} + P_{i_2} + \cdots + P_{i_t} = 0$$

Seja $x \in \mathbb{Z}_2^n$ com entradas 1 nas posições $i_1, i_2, \dots, i_t$ e zero nas demais. Então $w(x) = t$ e $Px = 0$. Logo x é um vetor código com peso $t < d = d(C)$. Isto contradiz a Proposição 5.1.4 e, portanto temos um absurdo.

Reciprocamente, suponha que qualquer $d-1$ colunas de P sejam L.I. e que alguns conjuntos de d colunas de P sejam L.D. Como Px é uma combinação linear das colunas de P correspondentes às posições dos 1s em x, $Px \neq 0$ para qualquer vetor x com peso $d-1$ ou menos. Portanto, não existem vetores código não nulos com peso menor que d. Mas algum conjunto de d colunas de P é L.D., portanto, existe um vetor x com peso d tal que $Px = 0$. Logo, esse x é um vetor código com peso d. Novamente, pela Proposição 5.1.4, deduzimos que $d(C) = d$.

Exemplo: Suponha que uma mensagem contenha os vetores $x_0 = (0,0)$, $x_1 = (0,1)$, $x_2 = (1,0)$ e $x_3 = (1,1)$ em $\mathbb{Z}_2^2$ e que a matriz geradora padrão seja tal que

$$Gx_0 = c_0, Gx_1 = c_1, Gx_2 = c_2, Gx_3 = c_3, \text{ onde } C = \{c_0, c_1, c_2, c_3\}$$

é o código do exemplo dado após definição 5.1.3. É fácil determinar G como sendo a matriz $\left(\frac{I_2}{A}\right) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}$ e portanto $P = (A|I) = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix}$ é a matriz de verificação de paridade.

Temos que C é um código linear $(5, 2)$. Observe que os vetores e_1, e_2 e $e_1 + e_2$ (que correspondem às 3ª, 4ª e 1ª colunas de P) são L.D. e que quaisquer duas colunas de P são L.I. Pelo Teorema 5.2.6, isto significa que $d(C) = 3$ (coincidindo com nossas contas anteriores).

Em vista da importância da distância mínima nos códigos lineares, existe a seguinte:

Notação: Se um código linear (n, k) tem distância mínima d, ele é chamado de código (n, k, d).

6. **CÓDIGO DE HAMMING**

Ao buscarmos um código que possa transmitir a maior quantidade de informações possível é desejado que k (e, portanto n) seja o maior possível. Pelo Teorema 4.1.2 as n colunas de P precisam ser não nulas e distintas, então o máximo ocorre quando elas consistem de todos os vetores não nulos de $\mathbb{Z}_2^{n-k}$. Como $\mathbb{Z}_2^{n-k}$ possui $2^{n-k}-1$ vetores não nulos e distintos, esse máximo é atingido quando $n = 2^{n-k}-1$.

Definição 6.1.1. (Códigos de Hamming)

Um **código de Hamming** $(\boldsymbol{n}, \boldsymbol{k})$ sobre $\mathbb{Z}_2$ é um código com matriz de verificação de paridade P de ordem $(n-k)\times n$ cujas colunas são todos os vetores de $\mathbb{Z}_2^{n-k} \setminus \{0\}$ numa ordem qualquer.

Observe que para cada inteiro $m = n-k > 1$, existem únicos n e k tais que $n = 2^{n-k}-1$.

Exemplo: Se $m = 2$, $n = 3$ então $k = 1$ e, portanto, existem códigos de Hamming (3,1). Como ilustração, se $G = \begin{bmatrix}1\\1\\1\end{bmatrix}$ é a matriz geradora padrão do código, então $P = \begin{bmatrix}1 & 1 & 0\\1 & 0 & 1\end{bmatrix}$ é a matriz de verificação de paridade padrão associada a G para esse código. Neste caso, os vetores de mensagem são os dígitos 0 e 1 e os vetores de código são [0, 0, 0] e [1, 1, 1].

Exemplo: Se $m = 3$, $n = 7$ e $k = 4$. Uma matriz candidata a matriz de verificação de paridade para um código de Hamming (7, 4) é

$P = \begin{bmatrix}1 & 0 & 1 & 1 & 1 & 0 & 0\\1 & 1 & 0 & 1 & 0 & 1 & 0\\0 & 1 & 1 & 1 & 0 & 0 & 1\end{bmatrix} = [A \quad I_3]$ e, pelo Teorema 4.1.2, a matriz geradora padrão desse código é $G = \begin{bmatrix}I_4\\A\end{bmatrix}$

Corolário 6.1.2.

Se $C(n,k)$ é um código binário autodual, então C **não** é um código de Hamming (n,k).

Demonstração Se C fosse de Hamming então n seria ímpar (pois $n = 2^{n-k}-1$), mas isto é um absurdo, pois pelo Teorema 4.4.8, n deve ser par.

Teorema 6.1.3

Todo código de Hamming tem distância mínima 3. Isto é, códigos de Hamming são códigos (n, k, d) onde $n = 2^{n-k} - 1$ e $d = 3$.

Demonstração: Considere um código de Hamming C com matriz de verificação de paridade P. Duas colunas em P são sempre LI, e sempre é possível achar três colunas LD em P, por exemplo, e_1, e_2 e $e_1 + e_2$. Portanto, pelo Teorema 5.2.6, temos que $d = 3$.

Corolário 6.1.4

Os códigos de Hamming detectam até dois erros e corrigem um erro.

Demonstração: Pelo Teorema 6.1.3 sabe-se que Se $d = 3$, então $\kappa = \left[\frac{d-1}{2}\right] = 1$ e, portanto, tem-se o resultado.

CONCLUSÃO

Neste trabalho, tivemos a oportunidade de estudar os códigos de Hamming para detecção e correção de erros, mostrando uma aplicação prática da matemática na área da Tecnologia da Informação. Ter conhecimento de aplicações como essas é útil ao professor de matemática, que está sempre sendo questionado pelos alunos sobre a importância da matemática para além da sala de aula. Por exemplo, os códigos de detecção e correção de erros que vimos ao longo do texto podem ser apresentados aos alunos, com as devidas simplificações, como motivação adcional ao ensino aritmética e de matrizes.

O leitor interessado em se aprofundar, pode pesquisar sobre outras famílias de códigos lineares como os códigos de Reed-Muller e o código de Golay, ambos utilizados pela NASA nas décadas de 1970 e 1980, de acordo com Poole (2014) para transmissão confiável de fotos vindas de sondas espaciais como a Mariner 9 e o Voyager 1 e 2. Além destes, há o CRC (estudado em cursos de Redes de Computadores) como um exemplo de código utilizado atualmente. Entretanto, o CRC usa outras ferramentas matemáticas para corregir os erros de transmissão.

REFERÊNCIAS

BROOKSHEAR, J.G. **Ciência da computação: uma visão abrangente.** 7. Ed. São Paulo: Artmed, 2003.

FERREIRA, André Arantes. **Detecção e Correção de Erros em Códigos Binários.** Dissertação de mestrado. Universidade do Porto, 2007. Disponível em http://www.fc.up.pt/fcup/contactos/teses/t_030370081.pdf

FORMIGA, Maurício Marinho. **Comunicação de Dados para um Sistema de Telemetria de Baixo Custo**. Dissertação de Mestreado. Disponível em http://docs.computacao.ufcg.edu.br/posgraduacao/dissertacoes/2005/Dissertacao_MauricioMarinhoFormiga.pdf

HAMMING, Richard W. **Error Detecting and Error Correcting Codes.** The Bell System Technical Journal vol.XXVI. April, 1950. Disponível em http://www.sns.ias.edu/~tlusty/courses/InfoInBio/Papers/Hamming1950.pdf

HEFEZ, Abramo. **Curso de Álgebra**. v.1. 3ª ed. Rio de Janeiro: IMPA, 2002.

HEFEZ, Abramo; VILLELA, Maria Lúcia Torres. **Códigos Corretores de Erros.** 2 ed. Rio **de** Janeiro: IMPA, 2008.

MILIES, César Polcino. **A Matemática dos Códigos de Barras**. Campina Grande: UFCG, 2006. Disponível em www.mat.ufg.br/bienal/2006/mini/polcino.pdf

POOLE, David. **Álgebra Linear**. 8ª ed. São Paulo: Cengage Learning, 2014.

SHANNON, Claude E. **A Mathematical Theory of Communication.** The Bell System Technical Journal vol.XXVII. July, 1948. Disponível em http://worrydream.com/refs/Shannon%20-%20A%20Mathematical%20Theory%20of%20Communication.pdf