# A MATEMÁTICA DOS CÓDIGOS DE BARRAS

C. POLCINO MILIES

## 1. Introdução

Hoje em dia, muitos produtos são identificados com um código numérico. O progresso da tecnologia, que tornou relativamente baratos e acessíveis aparelhos de leitura óptica e computadores, tornou também uso de este tipo de códigos bastante freqüente. Por exemplo, os produtos que compramos num supermercado estão identificados por um código de barras, como o que mostramos na Figura 1.1. Ele não é mais do que um número, assignado ao produto para sua identificação, escrito de forma a permitir uma leitura rápida no caixa. Note que imediatamente abaixo das barras, aparece o mesmo número escrito em algaritmos correntes, de forma que o leitor humano também possa ler o número.


Figura 1. Código de barras

Porém, algumas vezes acontece que, ao passar um produto pela leitora ótica (por exemplo, quando a embalagem está húmida ou enrrugada), esta não consegue realizar a leitura. O que vemos então é que a pessoa que está no caixa tenta pasar o produto em sentido contrário, ou inverte o produto, de modo que o código de barras fique de cabeça para baixo, e tenta passa-lo mais uma vez. Se nem asim dá certo, então ele próprio lê o código e o digita rapidamente.

Naturalmente, estas atitudes sugerem algumas perguntas. Em primeiro lugar, uma vez que o desenho das barras é totalemente simétrico para a máquina, que o lê usando um feixe de luz transversal, ao passá-lo "de ponta cabeça"; ela não deveria ler o número na ordem contrária? E, o que é pior, o operador do caixa, ao digitar o número rapidamente, não poderia cometer um erro e nós acabarmos pagando por um produto muito mais caro que aquele que estamos comprando?

Na verdade, isso não ocorre. Tanto quando lido numa ordem, como na ordem contrária, o código sempre é interpretado de forma correta. Mais ainda, quando o operador comete algum erro de digitação - e todos nós já vimos isso acontecer alguma vez - a máquina simplesmente emite um som, para avisar que houve um erro!

O objetivo destas notas é explicar como e porqué isso acontece. Começaremos contando um pouco da história destas idéias na próxima seção. No capítulo seguinte vamos explicar como é feita a tradução dos números em termos de barras e como a leitora distingue entre esquerda e direita; na seção §3 vamos explicar as idéias matemáticas que fazem com que eventuais erros possam ser detectados. No capítulo final, mostraremos brevemente como estas idéias podem ser extendidas para se obter códigos mais sofisticados.

## 2. Um pouco de história

A idéia de utilizar máquinas para realizar cálculos é, na verdade, bastante antiga. Já em 1642, **Blaise Pascal** construiu a primeira máquina de calcular de que se tem notícia. Ela funcionava com engrenagens

mecânicas e era capaz de realizar apenas somas. Em 1694, **Wilhelm Leibniz** aprimorou o invento de Pascal e criou uma máquina capaz de realizar também multiplicações. Do ponto de vista destas notas, o que é realmente interessante é a forma de transmitir dados à máquina já que, praticamente desde os começos da automação, isto foi feito com cartões perfurados que são antepassados diretos dos códigos de barra.

De forma muito esquemática, os estágios do processo que nos interessa são os seguintes:

- Em 1728, **B. Bouchon** concebeu a idéia de cifrar informações em folhas de papel perfurado. A descoberta verdadeiramente importante veio em 1801, quando **Joseph-Marie Jacquard** (1752-1834) construiu um tear que era comandado por cartões perfurados e que foi, talvez, a primeira máquina programável. O tear de Jacquard era capaz de realizar todos os movimentos necessários e ele foi o primeiro a produzir tecidos com padrões figurativos. Essa máquina deu ímpeto à revolução tecnológica da era industrial e foi a base para o desenvolvimento do moderno tear automático.
- Em 1857, **Sir Charles Wheatstone** utilizou fitas de papel para armazenar dados, seguindo o mesmo princípio básico dos cartões perfurados, mas com a vantagem de poder alimentar dados de forma contínua.
- Em 1822, **Charles P. Babbage** (1792-1871), um professor de matemática de Cambridge e um dos fundadores de *Analytical Society* (grupo de professores que modificaria o ensino da matemática na Inglaterra e que tiveram influência fundamental na criação da álgebra abstrata) inventou um instrumento de cálculo mais sofisticado, que denominou *Máquina Diferencial*. Nessa essa época, ele observou que "as operações matemáticas repetitivas poderiam ser desenvolvidas com mais agilidade e confiabilidade pelas máquinas que pelos homens". Mais adiante, em 1833, projetou uma *Máquina Anaítica*, isto é, uma máquina capaz de executar todas as operações aritméticas, de fazer comparações e analisar seus próprios resultados, que era programada através de *cartões perfurados*. Ele foi o primeiro a perceber que uma máquina de computar deveria ter um dispositivo de entrada, uma memória (que ele chamou de *mill* = *moinho*) e um dispositivo de saída. Sua máquina, em particular, seria alimentada por duas séries de cartões perfurados: uma com os dados e outra com as operações a serem executadas. Por causa disto, ele é considerado o pai do computador digital.

  Suas idéias despertaram o interesse de Ada August, condessa de Lovelace e filha de Lord Byron, que foi a primeira programadora da história. No período de 1842-1843 ela taduziu do italiano um artigo de Luigi Menebrea sobre a máquina da Babbage e, num apêndice, detalhou um método completo para calcular números de Bernoulli com a máquina.

  Babbage conseguiu convencer o governo britânico a financiar seu projeto mas, apesar dos esforços de anos e de vários investimentos governamentais, a máquina jamais chegou a ser construida. De acordo ao plano original, ela seria movida a vapor e de tamanho maior do que uma locomotiva.
- A próxima personagem importante nesta história é **Hermann Hollerith** (1860-1951) que obteve um doutorado em estatística na Columbia University em 1879 e, logo em seguida, foi empregado pelo Bureau de Censos dos EUA para trabalhar com seu professor, William P. Trowbridge, no censo de 1880. Foram necessários dez anos de trabalho para tabular completamente todos os dados recolhidos. Durante este período, Hollerith deu também aulas durante algum tempo no Massachussets Institute of Technology e trabalhou no Escritório de Patentes de Washington. Sobretudo, ele empregou seu tempo projetando uma máquina que pudesse tabular dados automaticamente. Para isso, ele utilizou novamente a idéia dos cartões perfurados de Jacquard, escrevendo dados em oito colunas que utilizavam o sistema de numeração binária. Esses cartões eram então lidos por sua máquina que utilizava sensores elétricos. Quando foi realizado um novo censo, em 1890, apesar dele ser mais sofisticado e coletar mais dados que o anterior, a invenção de Hollerith pode tabular todos os dados em apenas seis semanas.

  Sua criação teve sucesso immediato e ele deixou seu emprego para fundar uma companhia dedicada ao desenvolvimento de máquinas semelhantes, a Tabulating Machine Company que, com o decorrer dos anos, se transformou na atual IBM.
- O desenvolvimento de computadores eletrônicos ganhou mais força a partir da segunda guerra mundial, quando foi percebido seu potencial estratégico.

Muitos autores consideram que o Atanasoff-Berry Computer (ABC), desenvolvido nos EUA no período 1937-42 é o primeiro computador eletrônico digital. Porém, ele não tinha capacidade de programação geral e servia apenas para resolver sistemas lineares, além de ter outras limitações técnicas.

Em 1941 os alemães desenvolveram o Z3, desenhado por Konrad Zuse. Era eletromecânico, mas com objetivos mais gerais e totalmente programável mediante fitas perfuradas.

Em 1944, Tommy Flowers desenhou, na Inglaterra, o computador Colossus, totalmente eletrônico, concebido para decodificar mensagens interceptadas aos alemães.

O computador mais famoso deste período foi o ENIAC (Electronic Numerical Integrator and Computer), desenhado por John Mauchly e J. Presper Eckert, da Universidade de Pennsylvania e construído na Escola Penn Moore de Engenharia Eletrônica daquela universidade. Embora ele fosse um computador de propósitos gerais, ele foi desenhado originalmente para calcular tabelas de fogo de artilharia para o Laboratório de Pesquisas Balísticas, durante a guerra, mas sua construção só foi completada três meses após o fim da guerra. Os primeiros problemas computados pelo ENIAC foram relacionados à construção da bomba de hidrogênio. Tanto a entrada quanto a saída de dados era feita através de cartões perfurados.

O aparelho pesava 27 toneladas, usava 17.468 válvulas e precisou, para sua construção, de mais de cinco milhões de soldas feitas a mão. Ocupava todo um galpão e consimuia 150 kW de energia elétrica. Costuma-se dizer que, quando estava em uso, provocava apagões na cidade de Pennsylvania mas isto não deve ser verdade pois possuia alimentação independente da rede elétrica. Quando estava em operação, elevava a temperatura do local a 50 graus. Ele foi desativado em 2 de outubro de 1955.

FIGURA 2. O computador ENIAC

Outro computador de grande porte construído nesse período foi o Mark I. O projeto, concebido por Howard Aiken da Universidade de Harvard, iniciou-se em 1939 mas foi concluído apenas em 1943, na IBM. Foi tansladado a Harvard onde foi mostado publicamente, pela primeira vez, em 1944 e foi batizado oficialmente como Harvard-IBM Automatic Sequence Controlled Calculator (ASCC). Media 15.5 m de comprimento, 2.40 m de altura e aproximadamente 60 cm de largura.

A partir de então, os progressos da tecnologia permitiram diminuir gradativamente o tamanho (e o custo!) dos computadores até popularizá-los definitivamente. Foi também a tecnologia que permitiu usar feixes de luz e scanners para transmitir dados direta e rapidamente aos computadores, criando assim condições para a utilização da codificação que nos interessa estudar aqui.

## Códigos de barras

A primeira patente de um código de barras foi atribuida em 1952 a Joseph Woodland e Bernard Silver. Seu código consistia num padrão de circunferências concéntricas de espessura variável. Ao dar entrada ao pedido de patentes, eles descreviam seu invento como *uma classificação de artigos através de identificação de padrões*.

Em torno de 1970, uma firma de assessoria, a McKinsey & Co., junto com a Uniform Grocery Product Code Council [1] definiu um formato numérico para identificar produtos e pediu a diversas companhias que elaborassem um código adequado. Dentre as firmas contactadas, a que acabou apresentando a proposta vencedora foi a IBM e o código foi criado por George J. Laurer[2]

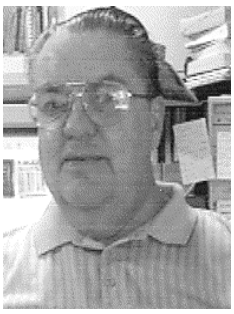

FIGURA 3. George J. Laurer

O código proposto, formalmente aceito em maio de 1973, passou a ser conhecido como código UPC (Universal Product Code) e foi adotado nos Estados Unidos e Canadá. Ele consistia de uma seqüência de 12 dígitos, traduzidos para barras da forma que analizaremos detalhadametne na próxima seção. Existem várias versões sucesivas do UPC, com pequenas modificações. Posteriormente foi solicitado a Laurer que ampliasse o código, para permitir uma maior difusão do sistema, de modo a identificar também o país de orígem de cada produto classificado. Baseado no UPC-A, ele acabou criando um novo código, com 13 dígitos, que foi adotado em dezembro de 1976 com o nome EAN (European Article Numbering system). Alguns países adotam este mesmo sistema, dando-lhe outro nome. Por exemplo, no Japão o sistema é conhecido como JAN (Japanese Article Numbering system).

## 3. Códigos de barras

### 3.1. **Escrevendo com barras.**

Vamos estudar inicialmente o código UPC, que é mais simples. Se observamos o código de barras da figura 3.1, notamos imediatamente que ele é formado por listras brancas e pretas alternadas, de espessura variável. Há, na verdade, quatro espessuras possíveis para essas listras, que podem ser classificadas como finas, médias, grossas ou muito grossas.

Vamos utilizar o símbolo 0 para indicar uma listra branca fina, o símbolo 00 para uma listra branca média, 000 para uma listra branca grossa e 0000 para uma muito grossa. Da mesma forma, vamos representar por 1, 11, 111 e 1111, uma listra preta fina, média, grossa ou muito grossa, respectivamente.

Assim, as primeiras quatro listras da figura (sem contar, é claro as listras que servem de limite e que aparecem mais compridas na figura), que são uma listra branca grossa, uma preta média, uma branca fina e uma preta fina respectivamente, podem ser representadas pela seqüência 0001101.

Como já dissemos, o código de barras representa uma série de números. A cada número lhe é assignado um espaço de espessura fixa, que corresponde sempre a uma seqüência de *sete* dígitos iguais a 1 ou 0. Por exemplo, a seqüência 0001101 que achamos acima representa o número 0, o primeiro do código da figura. O seguinte número do código, o 7, é representado pela seqüência 0111011.

[1] a plavra *Grocery* en inglês, equivale aproximadamente a nossa expressão *secos e molhados*.

[2] Estes dados, bem como a foto do autor, foram obtidos no site do próprio Laurer, que diz ter tido a colaboração de outras duas pessoas, mas não cita os nomes. O endereço do site é:
http://www.bellsouthpwp.net/l/a/laurergj/upc_work.html


FIGURA 4. Código UPC

Agora que começamos a compreender a forma de escrever com barras, já podemos responder à primeira das nossas perguntas: como a leitora distingue a direita de esquerda, quando o artigo pode ser passado em uma ou outra direção?

A resposta é muito engenhosa e também bastante simples. Os dígitos são codificados de maneia *diferente* quando estão do lado direito ou esquerdo do código de barras. Isto é feito conforme à seguinte tabela:

| dígito | do lado esquerdo | do lado direito |
|---|---|---|
| 0 | 0001101 | 1110010 |
| 1 | 0011001 | 1100110 |
| 2 | 0010011 | 1101100 |
| 3 | 0111101 | 1000010 |
| 4 | 0100011 | 1011100 |
| 5 | 0110001 | 1001110 |
| 6 | 0101111 | 1010000 |
| 7 | 0111011 | 1000100 |
| 8 | 0110111 | 1001000 |
| 9 | 00010011 | 1110100 |

Note que a codificação de um dado número, à direita, se obtém da sua codificação à esquera, trocando cada 0 por 1 e reciprocamente. Agora, o mecanismo de reconhecimento fica claro se notamos que *cada seqüência do lado esquerdo tem um número ímpar de dígitos iguais a 1 e, conseqüentemente, cada uma das que estão à direita tem um número par*. Assim, verificando a paridade de cada seqüência de sete dígitos, a maquina "sabe" imediatamente de que lado está lendo o código.

A elaboração do código EAN se deparou com um problema bastante delicado. Era necessário adicionar um dígito à cada código, de modo a permitir a identificação do país de orígem do produto, mas se desejava fazer isto de uma forma tal que a mesma máquina leitora pudesse ler indistintamente códigos UPC e EAN.

Se observamos a figura abaixo, que representa o mesmo código numérico escrito em ambos sistemas, veremos que, a primeira vista, parecem diferentes pois, no número escrito para o leitor humano, vemos que há um 0 a mais, escrito no início da seqüência. Porém, se observamos o código de barras, vemos que é exatamente o mesmo.

O que foi feito é o seguinte. Os países que utilizavam o código UPC antigo, EUA e Canadá, são identificados com um 0, na frente, e o resto da codificação é feito utiliándo-se o sistema anterior.

Para outros países, os primeiros dois ou três dígitos, identificam o país. Por exemplo, o código de barras de todos os produtos produzidos no Brasil começa com a seqüência 789, que é a que identifica o país.[3] Como era necessário adicionar um dígito e também manter o o mesmo padrão de tamanho do código de barras, para não ter que modificar todas as leitoras, a idéia utilizada foi fazer com que o novo dígito *estivesse implícito na forma de escrita de todos os outros*. Para isso, não foi modificada a codificação

[3]Uma tabela completa, com os números identificatórios de cada país, pode ser encontrada na página internet http://www.barcodeisland.com/ean13.phtml


FIGURA 5. Os codigos UPC-A e EAN-13

do lado direito (permitindo assim que as leitoras continuassem a identificar o lado corrspondente) mas *a codificação do lado esquerdo varia, dependendo do dígito inicial*.

Um dígito do lado esquerdo pode ser agora codificado com um número par ou ímpar de dígitos iguais a 1, de acordo com a seguinte tabela:

| dígito | lado esquerdo par | lado esquerdo ímpar | lado direito |
|---|---|---|---|
| 0 | 0001101 | 0100111 | 1110010 |
| 1 | 0011001 | 0110011 | 1100110 |
| 2 | 0010011 | 0011011 | 1101100 |
| 3 | 0111101 | 0100001 | 1000010 |
| 4 | 0100011 | 0011101 | 1011100 |
| 5 | 0110001 | 0111001 | 1001110 |
| 6 | 0101111 | 0000101 | 1010000 |
| 7 | 0111011 | 0010001 | 1000100 |
| 8 | 0110111 | 0001001 | 1001000 |
| 9 | 00010011 | 0010111 | 1110100 |

Finalmente, para cada dígito inicial escolhe-se uma alternância diferente de pares e ímpares de acordo com o seguinte critério:

| Dígito inicial | $1^o$ | $2^o$ | $3^o$ | $4^o$ | $5^o$ | $6^o$ |
|---|---|---|---|---|---|---|
| 0 | ímpar | ímpar | ímpar | ímpar | ímpar | ímpar |
| 1 | ímpar | ímpar | par | ímpar | par | par |
| 2 | ímpar | ímpar | par | par | ímpar | par |
| 3 | ímpar | ímpar | par | par | par | ímpar |
| 4 | ímpar | par | ímpar | ímpar | par | par |
| 5 | ímpar | par | par | ímpar | ímpar | par |
| 6 | ímpar | par | par | par | ímpar | ímpar |
| 7 | ímpar | par | ímpar | par | ímpar | par |
| 8 | ímpar | par | ímpar | par | par | ímpar |
| 9 | ímpar | par | par | ímpar | par | ímpar |

Vamos ver um exemplo. Uma barra de cerais produzida no Brasil é identificada pelo código 7895000266241. Como corresponde, começa com a seqüência 789, de modo que o primeiro dígito, que estará implícito na

codificação dos demais, é sete. Conseqüentemene, deve-se usar, do lado esquerdo, a seguinte ordem de codificação (obtida na tabela acima):

**ímpar, par, ímpar, par, ímpar, par.**

Consultando então a tabela de codificação do EAN-13 obtemos:

$$8 \mapsto 0110111 \quad 9 \mapsto 0010111 \quad 5 \mapsto 0111001$$
$$0 \mapsto 0001101 \quad 0 \mapsto 0001101 \quad 0 \mapsto 0001101$$

Para os dígitos do lado direito não temos que nos preocupar com paridade, e obtemos, diretamente da tabela, a seguinte codificação:

$$2 \mapsto 1101100 \quad 6 \mapsto 1010000 \quad 6 \mapsto 1010000$$
$$2 \mapsto 1101100 \quad 4 \mapsto 1011100 \quad 1 \mapsto 1100110$$

Por tanto, o código de barras correspondente é:


FIGURA 6

Um último comentário à respeito deste código. Como já dissemos, os primeiros dois ou três dígitos do código de barras (dependendo do caso) servem para identificar o país de orígem do produto. Os cinco ou quatro dígitos que restam, até as barras centrais, identificam o fabricante. Os primeiros cinco dígitos do lado direito identificam o produto específico, desse fabricante. O último dígito, chamado **dígito de verificação**, é adicionado no final do processo de elaboração do código, de acordo a um método que veremos adiante.

Falta ainda responder a nossa segunda pergunta: com faz a máquina para detectar quando um operador apressado comete um erro de digitação? Isto será o assunto da nossa próxima seção.

## 4. A detecção de erros

Para compreender como funciona o processo de detecção de erros precisamos entender, inicialmente, como se atribui a cada produto, o dígito de verificação.

Suponhamos que um determinado produto está identificado, no sistema EAN-13, por uma dada seqüência de dígitos $a_1a_2 \ldots a_{12}a_{13}$. Como os primeiros dígitos identificam o país de origem, o fabricante e o produto específico, os primeiros doze dígitos da seqüência, estão determinados naturalmente, por um método padrão, a cargo de uma autoridade classificadora em cada país. Denotaremos o décimo terceiro dígito, de verificação, por $x$.

Para facilitar nossa exposição, vamos escrever esta seqüência como um vetor

$$\alpha = (a_1, a_2, \ldots, a_{11}, a_{12}, x).$$

O sistema EAN-13, se utiliza de um vetor fixo, que chamaremos, **vetor de pesos** que é:

$$w = (1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1).$$

Calcula-se, então o "produto escalar" de ambos vetores:

$$\alpha \cdot \omega = (a_1, \ldots, a_{12}, x) \cdot (1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1) =$$
$$= a_1 + 3a_2 + a_3 + 3a_4 + a_5 + 3a_6 + a_7 + 3a_8 + a_9 + 3a_{10} + a_{11} + 3a_{12} + x.$$

Agora, o dígito de verificação $x$ se escolhe de forma tal que a soma acima seja múltiplo de 10, isto é, tal que

$$\alpha \cdot \omega = 0 \ \ (mod\ 10).$$

Por exemplo, no caso do código da figura 6, os números que indicam o país de origem, o fabricandte e o produto são 789500026624. Vamos ver como foi determinado o dígito de verificação. Chamando este dígito de $x$ e fazendo o "produto escalar" com o vetor de pesos, temos:

$$7 + 3 \times 8 + 9 + 3 \times 5 + 0 + 3 \times 0 + 0 + 3 \times 2 + 6 + 3 \times 6 + 2 + 3 \times 4 + x = 99 + x.$$

Conseqëntemente, deve-se tomar $x = 1$.

Vejamos agora um exemplo de como funciona a detecção de erros. Um livro do autor [14] recebeu o código de barrras 9781402002380. Suponhamos que, por um erro de digitação no quarto dígito, este número é transmitido como $\alpha = 9782402002380$. Ao fazer a verificação de leitura, o computador que recebeu a informação faz a operação $\alpha \cdot \omega$ e obtém:

$$9 + 3 \times 7 + 8 + 3 \times 2 + 4 + 3 \times 0 + 2 + 3 \times 0 + 0 + 3 \times 2 + 3 + 3 \times 8 + 0 = 73.$$

Como o resultado não é um multiplo de 10, ele avisa que foi cometido algum erro.

O código UPC é muito semelhante. Como utiliza apenas 12 dígitos (pois usa apenas um para identificar o país de orígem do artigo, enquanto o EAN utiliza-se de dois), e o vetor de pesos utilizado pelo UPC também tem um dígito a menos; ele é:

$$\omega \ = \ (3, 1, 3, 1, 3, , 1, 3, 1, 3, 1, 3, 1).$$

O leitor notará que, se o digitador comete apenas UM erro de digitação, trocando um dos dígitos $a_i$ por um outro valor, então necessáriamente o produto $\alpha \cdot \omega$ *não será* congruente a 0 em módulo 10 e assim será possível detectar que o erro foi cometido. Se mais de um erro for cometido na digitação, o fato provavelmente ainda será detectado, mas já não podemos ter certeza, pois eles poderiam se "compensar" mutuamente e a soma poderia ainda continuar sendo um multiplo de 10.

O leitor pode-se perguntar qual é a função do vetor de pesos $\omega$. De fato, se a escolha do dígito de verificação $x$ fosse feita simplesmente de modo que

$$a_1 + a_2 + \cdots + a_{12} + x \equiv 0 \ (mod\ 10),$$

ainda assim UM erro de digitação seria detectado. Acontece que há um outro tipo de erro de digitação muito comum, que consiste em digitar todos os números corretamente, mas trocar a ordem de dois dígitos consecutivos.

Suponha que, ao digitar o número 9 788531 404580 do nosso primeiro exemplo, tenha se cometido esse tipo de erro, e que o número de fato digitado fosse 9 788351 404580. Ao efetuar a verificação ter-se-ia:

$$\begin{aligned}(9, 7, 8, 8, 5, 3, 1, 4, 0, 4, 5, 8, 0)(1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1) &= \\ &= \ 9 + 21 \ + \ 8 \ + \ 24 \ + \ 3 \ + \ 15 \ + \ 1 \ + \ 12 \ + \ 12 \ + \ 5 \ + \ 24 \\ &= \ 134 \ \not\equiv \ 0 \ (mod\ 10)\end{aligned}$$

Desta forma, o erro seria detectado.

Suponha agora que, ao digitar o número 9 781402 002380 do nosso segundo exemplo, tenha se cometido um erro desse mesmo tipo, e que o número de fato digitado fosse 9 781402 002830. Ao efetuar a verificação ter-se-ia:

$$\begin{aligned}(9, 7, 8, 1, 4, 0, 2, 0, 0, 2, 8, 3, 0) \cdot (1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1) &= \\ &= \ 9 + 21 + 8 + 3 + 4 + 2 + 6 + 3 + 24 \\ &= \ 80 \ \equiv \ 0 \ (mod\ 10)\end{aligned}$$

Este exemplo mostra que o sistema de detecção adotado acima *não tem a capacidade de detectar todo erro de transposição* cometido. Pode-se demostrar que a transposição de dois dígitos consecutivos $a_i$ e $a_{i+1}$ não é detectada, neste sistema de codificação, se e somente se $|a_i - a_{i+1}| = 5$ (veja o exercício 4).

## EXERCÍCIOS

(1) (i) Um determinado produto deve ser identificado, no código UPC pelo número 7 897595 90071. Determine o dígito de segurança correspondente.
(ii) Faça o mesmo para um produto cujo número é 7 894900 01152.

(2) Vamos definir um sistema de detecção de erros da seguinte maneira. A cada número de 12 dígitos $a_0 \dots a_{11}$ vamos asignar um dígito de verificação $a_{12}$ de forma tal que $\sum_{i=0}^{12} a_i \equiv 0 \ (mod\ 10)$.
(i) Achar o dígito de verificação que deve se adicionar ao número 7 234435 01297.
(ii) Provar que toda vez que apenas um número é alterado na digitação, este sistema é capaz de detectar o erro.
(iii) Mostrar que este sistema não é capaz de detectar qualquer erro de transposição.

(3) Nós afirmamos no texto que se apenas um erro de digitação for cometido, alterando um dos dígitos $a_i$ para um outro valor $b_i$, então o sistema UPC sempre será capaz de detectar o erro. Dê uma demonstração cuidadosa deste fato.

(4) (i) Mostrar que uma *transposição adjacente*; isto é, um erro do tipo

$$\dots a_i a_{i+1} \dots \mapsto \dots a_{i+1} a_i \dots$$

é detectada pelo sistema EAN-13 se e somente se $|a_i - a_{i+1}| \neq 5$. (Sugestão: note que $|w_i - w_{i+1}| = 2$).
(ii) Mostre que um erro de *transposição não adjacente* do tipo

$$\dots a_i a_{i+1} a_{i+2} \dots \mapsto \dots a_{i+2} a_{i+1} a_i \dots$$

não pode ser detectado pelo sistema EAN-13.
(iii) Mostrar que um erro de transposição em que dois dígitos não adjacentes $a_i$ e $a_j$ são trocados não pode ser detectado pelo sistema se a diferença $i - j$ é par.
(iv) Mostrar que, num erro como o descrito no item anterior, se a diferença $i - j$ é ímpar, então o erro pode ser detectado pelo sistema EAN-13 se e somente se $|a_i - a_j| \neq 5$.

## 5. Códigos numéricos

Como observamos na seção anterior, existem diversos tipos de erros que podem ser cometidos ao digitar um *vetor de identificação*. Os erros num único dígito e as transposições são, de longe, os mais frequentes. Autores como D.F. Beckley [1] e J. Verhoeff [16] investigaram sistematicamente os erros cometidos por operadores humanos. No quadro abaixo damos as frequencias relativas obtidas por Verhoeff, que citamos abreviando quadro publicado por H.H. Schulz [15] e também por G.B. Belyavskaya, V.I. Izbash e V.A. Shcherbacov [2].

| Tipo de erro | | Frequência relativa % |
|---|---|---|
| erro único | $\dots$ a $\dots \mapsto \dots$ b $\dots$ | 79 |
| transposição adjacente | $\dots$ ab $\dots \mapsto \dots$ ba$\dots$ | 10.2 |
| transposição alterna | $\dots$ abc$\dots \mapsto \dots$ cba$\dots$ | 0.8 |
| erro gêmeo | $\dots$ aa$\dots \mapsto \dots$ bb$\dots$ | 0.6 |
| erro gêmeo alternado | $\dots$ aba$\dots \mapsto \dots$ cbc $\dots$ | 0.3 |
| outros | | 9.1 |

Tabela 1. Tipos de erros e suas frequências segundo Verhoeff

Para descrever vários dos métodos existentes, vamos introduzir um pouco de linguagem geral. Denotaremos por $\mathcal{A}$ o conjunto de valores que podem assumir os dígitos utilizados na codificação. Por exemplo, no caso do código UPC da seção anterior, esse conjunto é

$$\mathcal{A} \ = \ \{x \in \mathbf{Z} \mid 0 \leq x \leq m-1\}.$$

O vetor com os dados $\alpha' = (a_1, \dots a_{n-1})$ será chamado de **vetor de informação** e o vetor, já acrecido do dígito de verificação será chamado de **número ou vetor de indentificação**

**Definição 5.1.** *Sejam $\omega = (w_1, \dots w_n)$, com $w_i \in \mathcal{A}$, $1 \leq i \leq n$ um* **vetor de pesos** *e $c \in \mathcal{A}$ um inteiro fixado. Dados dois inteiros positivos $m$ e $n$ e um conjunto de números $a_1, \dots a_{n-1}$ tais que $a_i \in \mathcal{A}$, $1 \leq i \leq n-1$, define-se o* **número de verificação** *$a_n$ como o único elemento de $\mathcal{A}$ que verifica a equação:*

$$\sum_{i=1}^{n} a_i w_i \ \equiv \ c \pmod{m}.$$

*Um sistema de codificação assim definido será denotado por $\mathcal{C} = (\mathcal{A}, m, n, c, \omega)$.*

Note que freqüentemente $\mathcal{A} = \{0, 1, \dots, m-1\}$. Neste caso, tomando classes módulo $m$, temos que $a_n$ é o único elemento de $\mathcal{A}$ que verifica:

$$\overline{a}_n \ = \ \overline{w}_n^{-1} \left( \overline{c} - \sum_{i=1}^{n-1} \overline{a}_i \overline{w}_i \right).$$

**Exemplo 5.2.**

Um sistema usado em alguns bancos (mas não todos) é o seguinte: o número de conta de um cliente é composto de 9 dígitos, sendo que o último é o dígito de verificação. Na nossa notação, o sistema pode ser descrito como $\mathcal{C} = (\mathcal{A}, 10, 2, 0, \omega)$ onde $\mathcal{A}$ é o conjunto dos dígitos de 0 a 9 e $\omega = (7, 3, 9, 7, 3, 9, 7, 3, 9)$. Por exemplo, o número de uma conta num certo banco é 95-005541-9. Podemos verificar que

$$\begin{aligned}(9,5,0,0,5,5,4,1,9) \cdot (7,3,9,7,3,9,7,3,9) = & \\ = \ & 63 + 15 + 15 + 45 + 28 + 3 + 81 \\ = \ & 250 \equiv 10 \pmod{10}.\end{aligned}$$

Nosso próximo Teorema descreve a capacidade que tem um sistema definido desta forma, para detectar os diversos tipos de erros mais freqüentes.

**Teorema 5.3.** (Capacidade de detecção) *Sejam $m$ um inteiro positivo e $\omega = (w_1, \dots w_n)$ um vetor de pesos. Suponhamos que um vetor de identificação $\alpha = (a_1 \dots, a_n)$ (onde assumimos que $0 \leq a_i < n$, para todo índice $i$, $1 \leq i \leq n$) satisfaz a condição*

$$\alpha \cdot \omega = a_1 w_1 + \cdots + a_n w_n \ \equiv \ c \ (mod\ m).$$

*Então:*

(1) *Todo erro consistente numa única alteração na posição $i$-ésima será detectadose e somente se $mdc(w_i, m) = 1$.*
(2) *Todo erro de transposição da forma*

$$\dots a_i \dots a_j \dots \mapsto \dots a_j \dots a_i \dots$$

*será detectado se e somente se $mdc(w_i - w_j, m) = 1$.*

**Demonstração.** Suponhamos inicialmente que o dígito $a_i$, na posição $i$, foi trocado por um outro valor $b_i$ e vamos denotar por $\beta$ o vetor resultante deste erro. É claro que o erro não será detectado se e somente se

$$\alpha \cdot \omega - \beta \cdot \omega \ \equiv \ 0 \ (mod\ m).$$

Mas $\alpha \cdot \omega - \beta \cdot \omega = (a_i - b_i)w_i$, de modo que o erro não será detectado se e somente se $m \mid (a_i - b_i)w_i \equiv 0 \ (mod\ m)$ ou, se denotamos por $\overline{x}$ a classe de um inteiro $x$ em $\mathbf{Z}_m$, se e somente se $(\overline{a_i} - \overline{b_i})\overline{w_i} = \overline{0}$ em $\mathbf{Z}_m$.

Se $mdc(w_i, m) = 1$ tem-se que $\overline{w_i}$ é inversível em $\mathbf{Z}_m$, donde a condição acima implica que $\overline{a_i} = \overline{b_i}$, logo $a_i \equiv b_i \ (mod\ m)$ e, como ambos os números são menores que $m$, isto só aconteceria se $a_i = b_i$. Logo, o erro será detectado.

Por outro lado, se $mdc(w_i, m) = d \neq 1$, dado $a_i$ tem-se que dentre os números $b_i = a_i + m/d$ e $b_i = a_i - m/d$, um deles verifica a condição $0 \leq b_i < m$ e o erro que substitui $a_i$ por esse número não pode ser detectado. Isto completa a demonstração de $(i)$.

Suponhamos agora que foi cometido um erro do tipo

$$\alpha = \ldots a_i \ldots a_j \ldots \quad \mapsto \quad \alpha' \ = \ \ldots a_j \ldots a_i \ldots .$$

Note que, neste caso podemos calcular a diferença

$$\alpha \cdot \omega - \alpha' \cdot \omega \ = \ (a_i w_i + a_j w_j) - (a_j w_i - a_i w_j) \ = \ (a_i - a_j)(w_i - w_j).$$

Assim, este erro não será detectado se e somente se

$$(a_i - a_j)(w_i - w_j) \ \equiv \ 0 \ (mod\ m).$$

Daqui em diante, o argumento é inteiramente análogo ao anterior. □

A luz deste teorema, resulta claro que a melhor forma de ter certeza que o sistema de codificaçào será capaz de detectar todos os erros únicos e todos os erros de transposição (contigua ou não) é tomar, para o valor do módulo $m$, um número primo. De fato, existem vários sistemas em uso que procedem desta forma.

**Exemplo 5.4.**

Um sistema universalmente adotado para a clasificação de livros é o ISBN (**International Standard Book Number**). Ele trabalha módulo 11, mas para facilitar a notação, utiliza também como conjunto de valores $\mathcal{A}$ os dígitos de 0 a 9 e os vetores de identificação tem 10 componentes. Ele pode ser descrito, na nossa notação, por $(\mathcal{A}, 11, 10, 0, \omega)$ com $\omega = (10, 9, 8, 7, 6, 5, 4, 3, 2, 1)$.

Por exemplo, o livro do autor [14] mencionado acima tem o número ISBN 1-4020-0238-6. O dígito final, de verificação é 6 porque

$$\begin{aligned}(1, 4, 0, 2, 0, 0, 2, 3, 8, 6) \cdot (10, 9, 8, 7, 6, 5, 4, 3, 2, 1) = \\ &= \ 54 + 48 + 63 + 6 + 45 + 16 + 27 + 12 + 4 \\ &= \ 275 \equiv 0 \ \ (mod\ 11)\end{aligned}$$

Este método, porém, tem um pequeno inconveniente, que se comprenderá melhor analizando mais um exemplo. Um conhecido livro de álgebra [13] tem como vetor de informação do vetor de seu código ISBN 0-387-96035. Vamos tentar determinar o dígito de verificação. Ele será um número $a \in \mathcal{A}$ tal que

$$(0, 3, 8, 7, 9, 6, 0, 3, 5, a) \cdot (10, 9, 8, 7, 6, 5, 4, 3, 2, 1) \equiv 0 \ \ (mod\ 11).$$

Efetuando as operações necessárias, obtemos que

$$243 + a \equiv 0 \ \ (mod\ 11).$$

ou

$$a \equiv -243 \ \ (mod\ 11).$$

Como $243 \equiv 1 \ (mod\ 11)$ e $-1 \equiv 10 \ (mod\ 11)$ temos que $a = 10$. Porém, no conjunto dos dígitos de 0 a 9, não temos nenhum que represente o número 10. Devemos introduzir então mais um símbolo para representar este número. A convenção usual é utilizar o símbolo X e assim, ó código ISBN que aparece neste livro é ISBN 0-387-96035-X.

Finalmente observamos que, se tomamos o número $m$ de modo que seja primo e o conjunto $\mathcal{A}$ é formado por inteiros menores do que $m$ - como em todos os exemplos acima - como cada componente $w_i$ do vetor de pesos é prima com $m$, resulta que multiplicar por $w_i$, em módulo $m$, equivale a definir uma permutação do conjunto $\mathcal{A}$ (isto é, uma bijeção de $\mathcal{A}$ em si mesmo). Isto sugere um método mais geral de definir o vetor de pesos.

Dado um vetor de informação $\alpha' = (a_1, \ldots, a_{n-1})$ podemos escolher $n$ permutações $\delta_1, \ldots \delta_n$ do conjunto $\mathcal{A}$, definir um "vetor de pesos" por $\gamma = (\delta_1, \ldots, \delta_n)$, fixar um número $c \in \mathcal{A}$ e escolher o dígito de verificação $a_n$ de modo que verifique a equação:

$$\gamma(\alpha) = \delta_1(a_1) + \cdots \delta_n(a_n) \ \equiv \ c \ \ (mod\ m).$$

Neste caso, o dígito de verificação fica definido por:

$$a_n \ = \ \delta_n^{-1} \left( c - \sum_{i=1}^{n-1} \delta_i(a_i) \right).$$

Este tipo de codificação também é usado na prática, como mostra o seguinte.

**Exemplo 5.5.**

Um código usado pela IBM utiliza como conjunto $\mathcal{A}$ os dígitos de 0 a 9; o valores $m = 10$, um valor qualquer $c \in \mathcal{A}$ e a permutação

$$\delta = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 0 & 2 & 4 & 6 & 8 & 1 & 3 & 5 & 7 & 9 \end{pmatrix}.$$

No caso de um vetor de identificação com um número ímpar de componentes $n$, se utiliza o vetor de pesos

$$\gamma \ = \ (I, \delta, I, \delta, \ldots, \delta, I).$$

Assim, a equação de verificação resulta:

$$a_n + \delta(a_{n-1}) + a_{n-2} + \delta(a_{n-3}) \cdots \ \equiv c \ \ (mod\ 10)$$

e o dígito de verificação é dado pela fórmula:

$$a_n = c - (\delta(a_{n-1}) - a_{n-2} - \delta(a_{n-3}) - \cdots) \ \ (mod\ 10).$$

No caso em que o número de componentes do vetor de identificação é par utiliza-se o vetor de pesos

$$\gamma \ = \ (\delta, I, \delta, I, \ldots, \delta, I)$$

e resulta uma equação de verificação análoga, com uma fórmula similar para o cálculo do dígito de verificação.

Este é o sistema utilizado para determinar os números nos cartões de crédito. Por exemplo, um determinado cartão tem o número 5745 5195 0431 5412. Vamos a aplicar a ele o sistema de verificação IBM:

$$\begin{array}{cccc cccc cccc cccc} 5 & 7 & 4 & 5 & 5 & 1 & 9 & 5 & 0 & 4 & 3 & 1 & 5 & 4 & 1 & 2 \\ \downarrow \delta & \downarrow I & \downarrow \delta & \downarrow I & \downarrow \delta & \downarrow I & \downarrow \delta & \downarrow I & \downarrow \delta & \downarrow I & \downarrow \delta & \downarrow I & \downarrow \delta & \downarrow I & \downarrow \delta & \downarrow I \\ 1 & 7 & 8 & 7 & 1 & 1 & 9 & 1 & 0 & 4 & 6 & 1 & 1 & 9 & 2 & 2 \end{array}$$

Agora, calculamos:

$$1 + 7 + 8 + 7 + 1 + 1 + 9 + 1 + 0 + 4 + 6 + 1 + 1 + 9 + 2 + 2 \ = \ 60 \ \equiv \ 0 \quad (mod\ 10)$$

Note que esta codificação permite detectar todo erro único de digitação e toda transposição adjacente exceto no caso em que $a_i$ e $a_j$ assumem os valores 0 e 9 ou 9 e 0 respectivamente (veja o Exercício 3). Observe, porém, que este sistema não detecta transposições do tipo

$$\ldots a_i \ldots a_j \ldots \ \mapsto \ \ldots a_j \ldots a_i \ldots$$

quando a diferença $i - j$ é par e também não permite detectar erros gêmeos.

O código IBM generalizado utiliza a mesma permutação, com o vetor de pesos:

$$\gamma \ = \ (\delta^{n-1}, \delta^{n-2}, \ldots, \delta, \delta^0)$$

onde $\delta^0 = I$. A equação é então:

$$\sum_{i=1}^{n} \delta^{i-1}(a_{n+1-i}) \equiv c \ \ (mod\ 10)$$

e

$$a_n = c - \sum_{i=2}^{n} \delta^{i-1}(a_{n+1-i}) \ \ (mod\ 10).$$

Este sistema detecta todo erro único de digitação, toda transposição, adjacente ou não, (exceto no caso já apontado, quando os dígitos envolvidos são 0 e 9) e todo erro gêmeo.

Vimos acima que o código ISBN detecta todo erro único de digitação e todo erro de transposição. É natural se perguntar, então, se existe um código com essa capacidade de detecção, trabalhando na base 10. Infelizmente, a resposta é negativa, como mostra o seguinte teorema.

**Teorema 5.6.** (Gumm [10]) *Se um sistema numérico de detecção de erros, com um módulo par, detecta todo erro único de digitação, então, para todo par de índices $i, j$ existe um erro de transposição entre as posições $i$ e $j$ que não é detectada pelo sistema.*

**Demonstração.** Como vamos trabalhar com os números de 0 a $2m-1$ e tomar congruências em módulo $2m$, vamos considerar nossos dígitos como elementos de $\mathbf{Z}_{2m}$, para simplificar nossos argumentos. Suponhamos que o sistema transforma o vetor $(a_1, \ldots, a_n)$ num outro vetor, que denotaremos $(\sigma_1(a_1), \ldots, \sigma(n(a_n))$. Claramente, se o sistema é capaz de detectar todo erro único de digitação, então a aplicação na posição $i$-ésima $x \mapsto \sigma_i(x)$ deve ser uma permutação de $\mathbf{Z}_{2m}$.

Para que o sistema detecte todo erro de transposição entre as posições $i$ e $j$ é necessário que $\sigma_i(a)+\sigma_j(b) \neq \sigma_j(a) + \sigma_i(b)$, todo par de elementos diferentes $a, b \in \mathbf{Z}_{2m}$. Isto é equivalente a dizer que a aplicação $\sigma = \sigma_i - \sigma_j$ é uma permutação $\mathbf{Z}_{2m}$.

Mas $m \in [0, 2m-1]$ e temos que

$$0 + 1 + 2 + \cdots + 2m - 1 = \frac{(2m-1) \centerdot 2m}{2} = 2m \centerdot m - m \equiv m \ \ (mod\ 2m).$$

Logo:

$$\begin{aligned} m &= \sum_{x \in \mathbf{Z}_{2m}} x = \sum_{x \in \mathbf{Z}_{2m}} \sigma(x) \\ &= b \sum_{x \in \mathbf{Z}_{2m}} (\sigma_i(x) - \sigma_j(x)) = \sum_{x \in \mathbf{Z}_{2m}} \sigma_i(x) - \sum_{x \in \mathbf{Z}_{2m}} \sigma_j(x) \\ &= m - m = 0, \end{aligned}$$

uma contradição. □

## EXERCÍCIOS

(1) Calcular o dígito de verificação para um livro cujo número ISBN tem, como vetor de informação o número 85-314-0458

(2) Use o sistema do exemplo 5.2 para determinar o dígito de verificação do número 13-010765.

(3) Demonstre que o código IBM e o código IBM generalizado tem a capacidade de detecção de erros mencionada no texto. (Sugestão: Para discutir erros de transposição adjacente, note que $\delta(x) = 2x$, se $x \in [0, 4]$ e que $d(x) = 2x - 9$ se $x \in [5, 9]$. Considere separadamente três casos:$(i)$ $a, b \in [0, 4]$, $(ii)$ $a, b \in [5, 9]$ e $(iii)$ $a \in [0, 4], b \in [5, 9]$ ou vice-versa. Mostre que nos casos $(i)$ e $(ii)$ o erro é sempre detectado e que, no caso $(iii)$ o erro só não é detectado se $a = 0$ e $b = 9$ ou $a = 9$ e $b = 0$.

(4) Mostre que no código UPC podem ocorrer 90 erros de transposição adjacente e que o código e capaz de detectar todos eles, exceto quando os pares de números adjacentes são 05, 16, 27, 38, 49 ou aqueles que se obtém invertendo estes.

## 6. Códigos sobre grupos

### 6.1. **O grupo dihedral.**

Na seção anterior apresentamos diversos métodos de detectar erros usando um dígito de verificação. Dentre estes, só o sistema ISBN para livros era capaz de detectar todo erro único de digitação e todo erro de transposição. Ele tinha, porém, o inconveniente de precisar da introdução de um dígito extra, para representar o número 10, que denotamos por $X$.

Em 1969 Verhoeff, na sua tese de doutoramento [16], desenvolveu um método simples, baseado não em cálculos com números inteiros, mas com os elementos de um certo grupo, que também detecta erros únicos de digitação e todos os erros de transposção adjacentes, sem necessidade de símbolos extras. A exposição elementar deste método que damos a seguir aparece em [6] e num texto básico de álgebra, do mesmoo autor [5, Capítulo V].

Consideremos o **grupo dihedral** $D_5$, que pode ser definido como o grupo das isometrias do plano que deixam fixo um pentágono regular dado. Este grupo contém dez elementos. Cinco deles são rotações: a identidade $R_0$, a rotação $R_1$ de ângulo $2\pi/5$ em sentido antihorário, e as rotações $R_2$, $R_3$ e $R_4$ de ângulos $2(2\pi/5)$, $3(2\pi/5)$ e $4(2\pi/5)$ respectivamente. Contém ainda cinco reflexões, em relação aos seus eixos de simetria, que passam por cada um dos vértices e o ponto médio do lado oposto: $S_6$, $S_7$, $S_8, S_9$ e $S_{10}$.

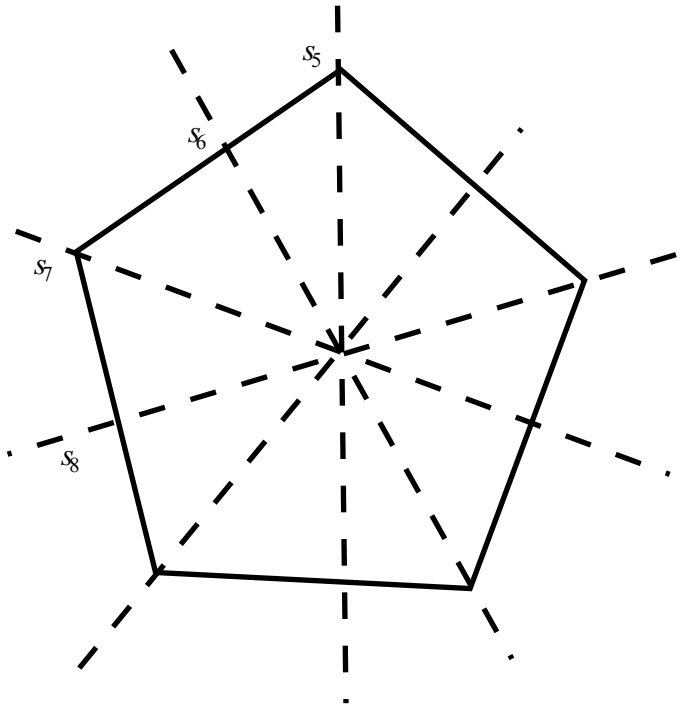


Figura 7. Simetrias do pentágono

Vamos usar simplesmente os subíndices $0, 1, 2, 3, 4$ para designar as rotações correspondentes e os subíndices $5, 6, 7, 8, 9$ para as respectivas reflexões.

Assim, por exemplo, a composição das rotações $R_3$ e $R_4$ seria a rotação de ângulo $3(2\pi/5) + 4(2\pi/5) = 7(2\pi/5) = 2(2\pi/5)$; isto é, a rotação $R_2$. Em vez de escrever então que $R_3 \circ R_4 = R_2$ escreveremos simplesmente que $3 \cdot 4 = 2$.

Da mesma forma, como a composição $S_6 \circ S_5 = R_1$ (lembre que, como se trata de composição de funções, aplicamos primeiro $S_1$ e depois $S_2$) escrevemos $6 \cdot 5 = 1$. Por outro lado, é fácil verificar que $S_5 \circ S_6 = R_0$ donde escrevemos $6 \cdot 5 = 0$.

Procedendo desta forma, obtemos a seguinte tabela de multiplicação para $D_5$.

| $\cdot$ | 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 | 0 | 6 | 7 | 8 | 9 | 5 |
| 2 | 2 | 3 | 4 | 0 | 1 | 7 | 8 | 9 | 5 | 6 |
| 3 | 3 | 4 | 0 | 1 | 2 | 8 | 9 | 5 | 6 | 7 |
| 4 | 4 | 0 | 1 | 2 | 3 | 9 | 5 | 6 | 7 | 8 |
| 5 | 5 | 9 | 8 | 7 | 6 | 0 | 4 | 3 | 2 | 1 |
| 6 | 6 | 5 | 9 | 8 | 7 | 1 | 0 | 4 | 3 | 2 |
| 7 | 7 | 6 | 5 | 9 | 8 | 2 | 1 | 0 | 4 | 3 |
| 8 | 8 | 7 | 6 | 5 | 9 | 3 | 2 | 1 | 0 | 4 |
| 9 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

TABELA 2. A tabela de multiplicação do grupo $D_5$

Consideremos a permutação:

$$\sigma = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 1 & 5 & 7 & 6 & 2 & 8 & 3 & 0 & 9 & 4 \end{pmatrix}.$$

A idéia de Verhoeff consiste em transformar um vetor de informação $(a_1, \ldots, a_{n-1})$ num vetor de codificação, adicionando um dígito de verificação $a_n$ de forma tal que

$$\sigma(a_1) \cdot \sigma^2(a_2) \cdot \cdots \cdot \sigma^{n-1}(a_{n-1}) \cdot a_n = 0 \text{ em } D_5.$$

Note que $\sigma$ é uma permutação de $D_5$. Ela foi escolhida para desenvolver este sistema porque pode-se verificar diretamente que

$$a \cdot \sigma(b) \neq b \cdot \sigma(a) \text{ para todo } a, b \in D_5. \tag{1}$$

Logo abaixo veremos a importância deste fato.

Como $\sigma^i$ também é uma permutação de $D_5$, para todo inteiro positivo $i$, resulta claro que este sistema de codificação detecta todo erro único de digitação.

Note que um erro de transposição adjacente $\ldots a_i a_{i+1} \ldots \mapsto \ldots a_{i+1} a_i \ldots$ será detectado se somente se $\sigma^i(a_i) \cdot \sigma^{i+1}(a_{i+1}) \neq \sigma^i(a_{i+1} s^{i+1}(a_i)$. Mas, vimos na equação 1 que $a \cdot \sigma(b) \neq b \cdot \sigma(a)$ para todo $a, b \in D_5$ o que implica, aplicando a permutação $\sigma^i$ a ambos os membros desta equação, que

$$\sigma^i(a) \cdot \sigma^{i+1}(b) \neq \sigma^i(b) \cdot \sigma^{i+1}(a) \text{ para todo } a, b \in D_5$$

como queriamos demostar.

Uma variante do método de Verhoeff[4] foi usada pelo orgão emissor de dinheiro da Alemanha, o Deutsche Bundesbank. As notas de dinheiro são numeradas num código alfanumérico; isto é, se utilizam tanto letras quanto números. Na Figura 6.1 reproduzimos uma nota de 10 marcos (que já está fora de circulação desde

[4]Citado por Gallian [7].

o advento da moeda unificada da Europa, o Euro).


FIGURA 8. Gauss

O código utilizado pelo banco segue uma variante do método anterior. Eles numeram as notas usando os dígitos de 0 a 9 e também dez letras: A, D, G, K, L, N, U, V e Z. Também se utiliza da tabela da operação do grupo $D_5$, mas ao invés de usar uma permutação e suas potências, utiliza dez permutações diferentes. Para determiná-las damos, na tabela abaixo, em cada fila $i$ os valores da função $\sigma_i$. Em outras palavras, na posição $i, j$ está o valor de $\sigma_i(j)$.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| $\sigma_1$ | 1 | 5 | 7 | 6 | 2 | 8 | 3 | 0 | 9 | 4 |
| $\sigma_2$ | 5 | 8 | 0 | 3 | 7 | 9 | 6 | 1 | 4 | 2 |
| $\sigma_3$ | 8 | 9 | 1 | 6 | 0 | 4 | 3 | 5 | 2 | 7 |
| $\sigma_4$ | 9 | 4 | 5 | 3 | 1 | 2 | 6 | 8 | 7 | 0 |
| $\sigma_5$ | 4 | 2 | 8 | 6 | 5 | 7 | 3 | 9 | 0 | 1 |
| $\sigma_6$ | 2 | 7 | 9 | 3 | 8 | 0 | 6 | 4 | 1 | 5 |
| $\sigma_7$ | 7 | 0 | 4 | 6 | 9 | 1 | 3 | 2 | 5 | 8 |
| $\sigma_8$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| $\sigma_9$ | 1 | 5 | 7 | 6 | 2 | 8 | 3 | 0 | 9 | 4 |
| $\sigma_{10}$ | 5 | 8 | 0 | 3 | 7 | 9 | 6 | 1 | 4 | 2 |

Isto significa, por exemplo, que a permutação $\sigma_5$ é:

$$\sigma_5 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 4 & 2 & 8 & 6 & 5 & 7 & 3 & 9 & 0 & 1 \end{pmatrix}.$$

O número de série da nota da Figura 6.1 é DU7124458G6. Vamos verificar que esta é uma numeração válida. Para poder usar o método de Verhoeff devemos trabalhar apenas com os dígitos de 0 a 9; por causa disso, a cada letra das usadas pelo Deutch Bundesbank deve-se lhe assignar um valor numérico. Isto é feito de acordo com a seguinte tabela.

| A | D | G | K | L | N | S | U | Y | Z |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Usando esta tabela, o número da nota em questão se transforma em 1771244 5826. Aplicamos então ordenadamente as permutações dadas:

| 1 | 7 | 7 | 1 | 2 | 4 | 4 | 5 | 8 | 2 | 6 |
|---|---|---|---|---|---|---|---|---|---|---|
| $\downarrow \sigma_1$ | $\downarrow \sigma_2$ | $\downarrow \sigma_3$ | $\downarrow \sigma_4$ | $\downarrow \sigma_5$ | $\downarrow \sigma_6$ | $\downarrow \sigma_7$ | $\downarrow \sigma_8$ | $\downarrow \sigma_9$ | $\downarrow \sigma_{10}$ | $\downarrow$ |
| 5 | 1 | 5 | 4 | 8 | 8 | 9 | 5 | 9 | 0 | 6 |

Agora precismos "multiplicar" estes números, com ajuda da tabela 6.1. Temos:

$$5 \cdot 1 = 9, \quad 9 \cdot 5 = 4, \quad 4 \cdot 4 = 3,$$

$$3 \cdot 8 = 6, \quad 6 \cdot 8 = 3, \quad 3 \cdot 9 = 7,$$

$$7 \cdot 5 = 2, \quad 2 \cdot 9 = 6, \quad 6 \cdot 0 = 6,$$

e finalmente

$$6 \cdot 6 = 0.$$

Este método, porém, tem um inconveniente. Nos cálculos, ele não distingue entre uma letra e o número que lhe é assignado. Assim por exemplo, se a letra K for trocada pelo número 3, o sistema será incapaz de detectar o erro. O mesmo acontece se ocorre uma transposição de 3 e K, ou viceversa. Para evitar este problema, poder-se-ia usar o grupo $D_{18}$, que tem 36 elementos (e portanto os vinte símbolos usado no código alfanumérico das notas corresponderiam a elementos diferentes em $D_5$), com uma permutação adequada. Sugestões nesse sentido apareceram, por exemplo, em trabalhos de Winters, em 1990 [17] e de Gallian e Mullin, em 1995 [8]

Note que a equação 1 foi essencial para podermos mostrar que o método de Verhoeff permite detectar erros de transposição adjacentes. Isto justifica a seguinte definição.

**Definição 6.1.** *Uma permutação $\sigma$ de um grupo $G$ diz-se uma* **aplicação anti-simétrica** *se verifica a seguinte condição:*

$$x\sigma(y) \neq y\sigma(x), \quad \text{para todo par de elementos } x, y \in G.$$

O grupo $D_5$ desempenha um papel importante na elaboração de códigos detectores de erros porque pode-se mostrar que ele é o único grupo de ordem 10 que possui uma aplicação anti-simétrica.

Como as transposições adjacentes se encontram entre os erros mais comuns e como códigos que detectam estes erros podem-se elaborar a partir de grupos com aplicações anti-simétricas, houve vários trabalhos que dedicam especial atenção a este tipo de grupos.

No caso dos grupos abelianos, há um tipo de permutação que é também importante.

**Definição 6.2.** *Uma permutação $\sigma$ de um grupo $G$ diz-se uma* **aplicação completa** *se a função $x \mapsto x\sigma(x)$, para todo $x \in G$, é uma permutação de $G$.*

Pode-se demonstrar que se $G$ é um grupo abeliano, então $G$ possui uma aplicação anti-simétrica se e somente se $G$ possui uma aplicação completa.

Um grupo abeliano de ordem $2m$, com $m$ ímpar não possui aplicações completas [3]. Como consequência imediata, temos que *Um grupo abeliano de ordem $2m$, com $m$ ímpar não possui aplicações anti-simétricas.*

Outros resultados relativos a este tipo de aplicações são os seguintes:

- Um grupo cíclico admite uma aplicação anti-simétrica se e somente se é de ordem ímpar [15].
- Todo grupo solúvel não abeliano admite uma aplicação anti-simétrica [11].
- Todo grupo simples, exceto $\mathbf{Z}_2$, admite uma aplicação anti-simétrica [8].

Foi anunciado por Heiss que todo grupo finito não abeliano também admite uma aplicação anti-simétrica [12].

Para os grupos dihedrais, diversas classes de aplicações anti-simétricas foram achadas em [4] e [9]. Como vimos, isto implica que estes grupos podem ser usados para construir códigos que detectam erros únicos de digitação ou transposições adjacentes. Porém, eles não podem detectar outros erros freqüentes, como mostra o seguinte teorema, devido a Damm [3, Teorema 5]

**Teorema 6.3.** *Seja $m > 2$ um inteiro ímpar. Não existe um sistema de dígito de controle sobre $D_m$ que seja capaz de detectar todas as transposições alternadas, todas os erros gêmeos ou todos os erros gêmeos alternados.*

# EXERCÍCIOS

(1) Determine todos os subgrupos cíclicos de $D_5$.
(2) Ache o centro $\mathcal{Z}(D_5)$ de $D_5$ e determine o quociente $D_5/\mathcal{Z}(D_5)$.
(3) Determine a decomposição em produto de ciclos disjuntos e a paridade da permutação $\sigma$ do método de Verhoeff.
(4) Determine o valor de $x$ para que o número 3572498x seja um número válido no método de Verhoeff.
(5) Mostre que tomando a permutação $\tau = (1\ 4)(2\ 3)(5\ 8\ 6\ 9\ 7)$ no método de Verhoeff, também é possível detectar todo erro único de digitação e toda transposição adjacente.
(6) Determine o valor do dígito $x$ para que o número $AD377345654Kx$ seja um número válido para uma nota emitida pelo Deustche Bank.
(7) Idem, para o número $NZ357x29477L2$.
(8) (Gallian [5]) Seja $\sigma = (1\ 2\ 4\ 8\ 7\ 5)(3\ 6)$. A cada número da forma $a_1a_2\ldots a_n$ (com $n$ ímpar) atribuímos o dígito de controle $-(\sigma(a_1) + a_2 + \sigma(a_3) + a_4 + \cdots + \sigma(a_n))\ \ (mod\ 10)$. Calcule o valor do dígito de controle para o número 3125600196431. Prove qie este método detecta todo erro único de digitação. Determine quais transposições adjacentes não podem ser detectadas por este método.

## Referências


[1] D.F. Beckley, An optimum system with modulo 11, *The Computer Bulletim*, **11** (1967), 213-215.
[2] G.B. Belyavskaya, V.I. Izbash and G.L. Mullen, Check character systems over quasegroups and loops, *Quasigroups and related systems*, **10** (2003), 1-28.
[3] M. Damm, Check digit over groups and anti-symmetric mappings, *Archiv der Math.*, **75** (2000), 413-421.
[4] A. Ecker and G. Poch, Check character systems, *Computing*, **37** (1986), 277-301.
[5] J.A. Gallian, *Contemporary Abstract Algebra*, D.C. Heath and Co., Lexington, 1990.
[6] J.A. Gallian, The Mathematics of Identification Numbers, *The College Math. J.*, **22**, 3 (1991), 194-202.
[7] J.A. Gallian, Error detecting methods, *ACM Computing Surveys*, **28**, 3 (1996), 504-517.
[8] J.A. Gallian and M. Mullin, Groups with antisymmetric mappings, *Archiv der Math.*, **65** (1995), 273-280.
[9] H.P. Gumm, A new class of check-digit methods for arbitrary number systems, *IEEE Trans, Inf. Th.*, **31** (1985), 102-105.
[10] H.P. Gumm, Encoding of numbers to detect typing errors, *Inter. J. Applied Eng. Ed.*, **2** (1986), 61-65.
[11] S. Heiss, Anti-symmetric mappings for finite solvable groups, *Archiv der Math.*, **69** (1997), 445-454.
[12] S. Heiss, Anti-symmetric mappings for finite groups, *preprint*, 1999.
[13] R. Lidl and G. Pilz, *Applied Abstract Algebra*, Undergraduate Texts in Math., Springer Verlag, New York, 1984.
[14] C. Polcino Milies and S.K. Sehgal, *An introduction to Group Rings*, Kluwer Acad. Publ., Dordrecht, 2002.
[15] R.H. Schulz, On check digit systems using anti-symmetric mappings, in *Numbers, Information and Complexity*, Kluwer Acad. Publ., Dordrecht, 2000, 295-310.
[16] J. Verhoeff, Error detecting decimal codes, *Math. Centre Tracts*, Mathematische Centrum, Amsterdam, 1969.
[17] S. Winters, Error detecting codes using dihedral groups, *UMAP J.*, **11** (1990), 299-308.


Há também muitas páginas na internet com informações sobe o assunto. Veja, por exemplo:

http://en.wikipedia.org./wiki/ENIAC

http://en.wikipedia.org/wiki/Harvard_Mark-I

http://en.wikipedia.org/wiki/Ada_Lovelace

http://www-etsi2.ugr.es/alumnos/mili/Harvard20I.htm

http://www.bellsouthpwp.net/l/a/laurergj/upc_work.html

http://www.barcodeisland.com/ean13.phtml

http://www.adams1.com/pub/russadam/barcode1.html

IME/USP - Departamento de Matemática, SP.