TRUQUES E MAGIA COM CÓDIGOS ALGÉBRICOS

Estefani Moraes Moreira e Jorge Picado
Universidade de Coimbra
moraesfani@gmail.com e picado@mat.uc.pt

A teoria algébrica dos códigos é uma área excitante da matemática que usa técnicas e estruturas algébricas abstratas, clássicas e modernas, na conceção de sistemas de identificação e códigos detetores e corretores de erros. Este artigo apresenta alguns truques de magia que ilustram de forma recreativa o alcance e a eficácia de códigos algébricos simples.

Two weekends in a row I came in and found that all my stuff had been dumped and nothing was done. I was really aroused and annoyed because I wanted those answers and two weekends had been lost. And so I said, 'Dammit if the machine can detect an error, why can't it locate the position of the error and correct it?'

Richard Hamming (1949)

1. DOIS TRUQUES DE MAGIA

Comecemos com um truque que provavelmente alguns leitores conhecerão. O mágico pede a um voluntário entre a assistência que pense num número entre 0 e 15:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Em seguida, pede ao voluntário que diga se o número em que pensou está em cada um dos cartões da figura 1 ('Sim' ou 'Não').

Imediatamente o mágico adivinha o número. Como? Simplesmente somando o algarismo mais à esquerda da linha de cima em cada um dos cartões com resposta afirmativa!

Vejamos porquê. Este truque baseia-se na representação binária dos números naturais. Em vez de 'Sim' ou 'Não', podemos representar as respostas com a ajuda dos algarismos binários 1 e 0. Os números foram dispostos nos quatro cartões de modo a que, e aqui reside a eficácia do truque, cada sequência de sins e nãos (uns e zeros) corresponda precisamente a um dos primeiros 16 naturais:

N N N N = 0000 = 0 | S N N N = 1000 = 8
N N N S = 0001 = 1 | S N N S = 1001 = 9
N N S N = 0010 = 2 | S N S N = 1010 = 10
N N S S = 0011 = 3 | S N S S = 1011 = 11
N S N N = 0100 = 4 | S S N N = 1100 = 12
N S N S = 0101 = 5 | S S N S = 1101 = 13
N S S N = 0110 = 6 | S S S N = 1110 = 14
N N S S = 0111 = 7 | S S S S = 1111 = 15

Como cada número *abcd* em representação binária corresponde ao número (em representação decimal)

$$a \times 2^3 + b \times 2^2 + c \times 2^1 + d \times 2^0,$$

colocando os números $2^3 = 8$, $2^2 = 4$, $2^1 = 2$ e $2^0 = 1$ numa posição especial em cada cartão (por exemplo, na linha de cima, no lugar mais à esquerda), o mágico tem um processo muito rápido para passar da representação binária dada pela sequência de respostas para o número a adivinhar: somando estes números em cada um dos cartões com resposta afirmativa.

É evidente que o truque funciona de modo semelhante

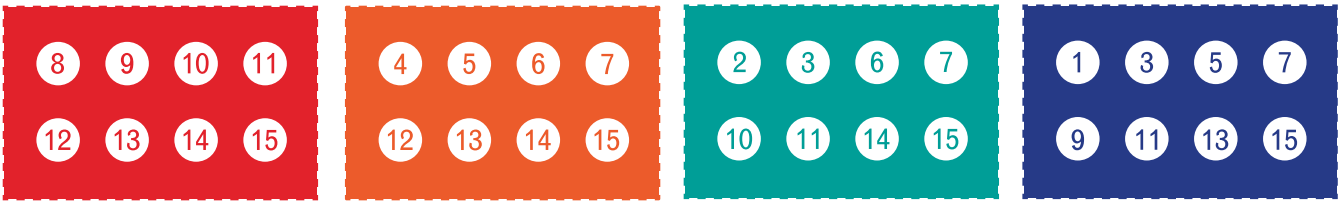


Figura 1: Cartões do primeiro truque.

para qualquer potência de 2 (isto é, para os números entre 0 e $2^n - 1$); nesse caso serão necessários n cartões nos quais se dispõem os números de acordo com a sua representação binária, 2^{n-1} números em cada cartão. Habitualmente este truque usa seis cartões e os números variam entre 0 e 63.

Se o voluntário mentir em algum dos cartões, o truque não funciona. Será possível conceber truques deste tipo em que se dê a possibilidade ao voluntário de mentir? Sim! É esse o nosso objetivo com este artigo: divulgar os truques produzidos por Todd Mateer [6] a partir de uma ideia original de Richard Ehrenborg [1]. Faremos isso no resto desta secção e nas secções 4, 5 e 6, com material retirado desses artigos. Nas secções 2 e 3 explicaremos as ideias matemáticas da teoria algébrica de códigos que permitem que os truques funcionem.

Comecemos pelo truque seguinte, muito mais elaborado do que o anterior, retirado de [6] .

O mágico pede novamente a um voluntário entre a assistência que pense num número entre 0 e 15. O mágico exibe então sete cartões de cor (figura 2) e pede ao voluntário que pense numa dessas cores e responda sequencialmente se o número em que pensou aparece em cada cartão ('Sim' ou 'Não'; mas desta vez é-lhe permitido que no cartão da cor escolhida possa mentir).

Além destes sete cartões, o mágico tem ainda outro cartão, o *cartão-base* (figura 3).

Após cada resposta, o mágico vai empilhando os cartões sobre o cartão-base; em caso de resposta negativa, coloca o cartão virado ao contrário depois de rodado de acordo com a figura 4 (observe que os números do cartão-base que ficariam cobertos pelo cartão na sua orientação original aparecem à vista após a rotação, e vice-versa).

Por exemplo, a figura 5 mostra, à esquerda, a colocação dos cartões após uma resposta afirmativa no cartão vermelho e, à direita, o resultado após respostas sem mentira à escolha do número 14 nos primeiros quatro cartões (ou seja: sim, sim, sim, não).

No final das sete respostas, o mágico adivinha o número e se o espetador mentiu ou não; caso tenha mentido, adivinha ainda o cartão acerca do qual o espetador mentiu (ou seja, a cor escolhida). De facto, no final uma de duas situações pode ocorrer: ou exatamente um número no cartão-base está à vista ou todos os números estão cobertos. No primeiro caso, o espetador nunca mentiu e o número à vista é o número pensado. No outro caso, só um número do cartão-base está coberto por exatamente um cartão: esse é o número pensado e a cor desse cartão é a cor do cartão acerca do qual o voluntário mentiu.

Mais uma vez, não há magia nenhuma, só matemática! Neste caso, a matemática dos códigos corretores de erros, que explicaremos sucintamente de seguida.

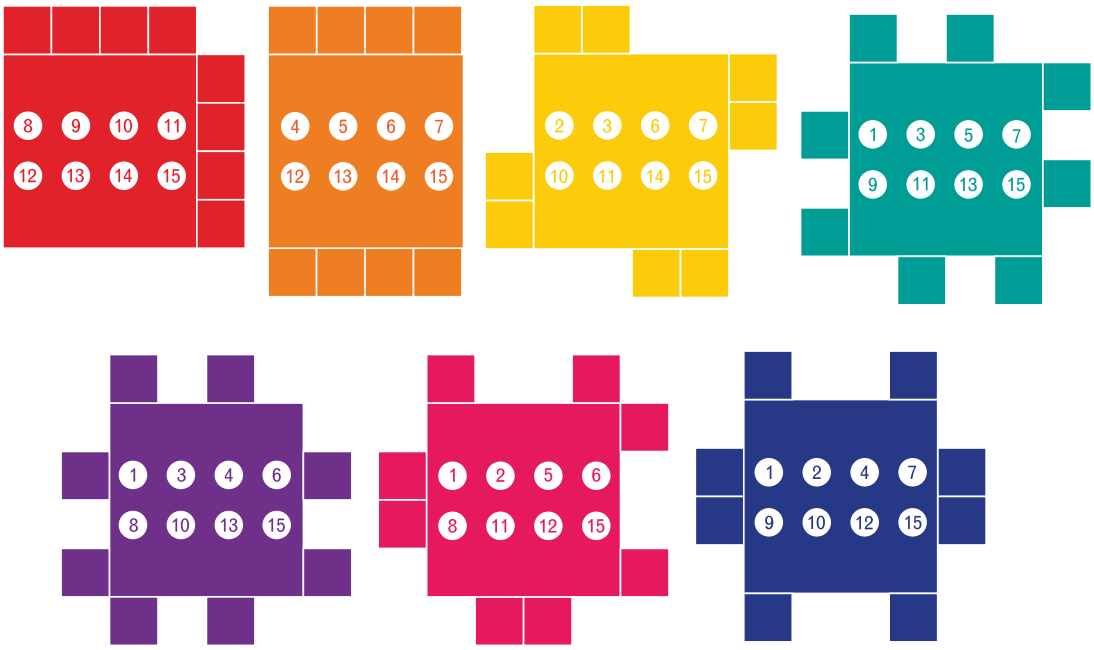


Figura 2: Cartões do segundo truque.

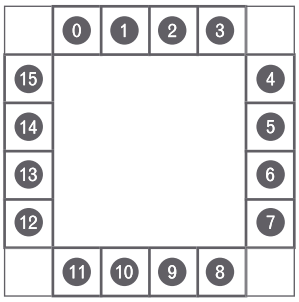


Figura 3: Cartão-base.

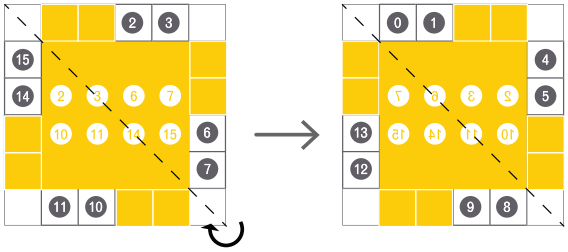


Figura 4: Empilhamento do cartão no caso de resposta negativa.

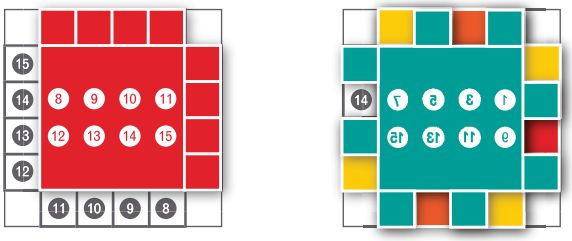


Figura 5: Exemplos de empilhamento de cartoes.

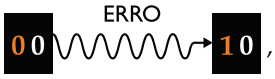
2. COMO DETETAR E CORRIGIR ERROS: MAGIA?!

A teoria dos códigos corretores de erros nasceu em 1945 nos Laboratórios Bell com o artigo [10] de C. Shannon sobre a teoria matemática da transmissão de informação. R. Hamming foi o outro pioneiro com o decisivo artigo [2]. Nos últimos 70 anos, os códigos algébricos revelaram-se uma das mais importantes aplicações da álgebra abstrata, base dos sistemas de comunicação modernos. São hoje usados em quase todos os sistemas de identificação, sistemas automáticos de telecomunicações, equipamentos de gravação e reprodução de informação, equipamentos óticos e *scanners*. É devido a eles que conseguimos comunicar com tanta eficácia a longa distância e ter larguras de banda tão grandes em redes de comunicações sem fios.

Qual é a ideia básica dos códigos algébricos? Por exemplo, consideremos o seguinte código binário, que designaremos por $\mathcal{C}_1$, que permite dar as instruções de comando a um leitor de DVD, através de um comando à distância:

Instruções	PLAY	REW	FORWARD	STOP
Código $\mathcal{C}_1$	00	01	10	11

Suponhamos que carregamos na tecla PLAY do comando, a que corresponde a palavra 00 do código; o comando transmite esta palavra ao leitor de DVD, mas se, porventura, nessa comunicação ocorrer o erro singular



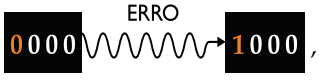
o leitor receberá a palavra 10, e como esta faz parte de $\mathcal{C}_1$ (corresponde à instrução FORWARD), aquele não terá nenhuma maneira de detetar o erro e executará a instrução FORWARD!

$\mathcal{C}_1$ é um exemplo de código definido sobre o alfabeto (corpo) $\mathbb{F}_2 = \{0,1\}$, constituído por todas as palavras de comprimento 2 nesse alfabeto. Trata-se de um código muito pobre pois nem sequer detecta erros *singulares* como o exemplo acima ilustra.

Qual é o truque de Hamming para melhorar o código? Muito simples. O que é que fazemos habitualmente quando não entendemos bem o que outra pessoa nos está a dizer? Pedimos que repita. Façamos isso no código $\mathcal{C}_1$, duplicando a informação em cada palavra:

Instruções	PLAY	REW	FORWARD	STOP
Código $\mathcal{C}_2$	00 00	01 01	10 10	11 11

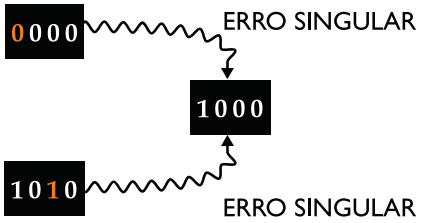
Agora, ao ser transmitida a instrução PLAY (ou seja, a palavra 0000), se ocorrer o mesmo erro singular de há pouco,



como a palavra recebida não faz parte de $\mathcal{C}_2$, o leitor de DVD pode concluir imediatamente que ocorreu algum erro na transmissão. Neste caso, o código $\mathcal{C}_2$ já deteta este erro singular (e é fácil de ver que deteta qualquer outro

erro singular). Bastou acrescentar alguma redundância às palavras do código para resolver o problema.

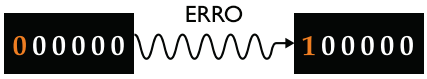
Será $\mathcal{C}_2$ capaz de corrigir esse erro, isto é, de identificar a palavra original (*assumindo que na transmissão só poderão ocorrer, quando muito, erros singulares*)? É claro que não, pois existem duas palavras em $\mathcal{C}_2$ que poderiam ser as originais:



Mas se triplicarmos a informação, com o código $\mathcal{C}_3$ definido pela tabela

Instruções	PLAY	REW	FORWARD	STOP
Código $\mathcal{C}_3$	00 00 00	01 01 01	10 10 10	11 11 11

além de qualquer erro singular ser detetável, também pode ser corrigido automaticamente (assumindo novamente que *na transmissão só poderão ocorrer, quando muito, erros singulares*). Por exemplo, o erro singular



é evidentemente detetado e corrigido; a única palavra de $\mathcal{C}_3$ que poderia ter dado origem à palavra 100000 é a palavra 000000:

Palavra de $\mathcal{C}_3$	000000	010101	101010	111111
Palavra recebida	100000	100000	100000	100000
N.º de erros	1	4	2	5

É claro que se puderem ocorrer erros duplos no canal de comunicação, $\mathcal{C}_3$ já não corrige o erro singular acima: a palavra original poderia muito bem ser a palavra 101010.

Assim, esta ideia de construir códigos corretores de erros só funciona se conhecermos *a priori* um limite para o número de erros que pode ocorrer no respetivo canal de comunicação. Ou, então, se adotarmos o seguinte princípio de bom senso, o chamado *princípio do vizinho mais próximo*:

> A palavra original correspondente a uma palavra recebida com erros deve ser a palavra do código "mais próxima" da palavra recebida.

(Ou seja, assumimos que é mais provável que o menor número de erros possível tenha ocorrido na transmissão.) Daqui em diante, assumimos sempre este princípio. (Na secção seguinte, tornaremos precisa a noção de distância implícita no termo "mais próxima".)

3. CÓDIGOS ALGÉBRICOS

Os códigos $\mathcal{C}_1$, $\mathcal{C}_2$ e $\mathcal{C}_3$ são exemplos de códigos binários. Um *código binário de comprimento n* é um subconjunto $\mathcal{C}$ de $(\mathbb{F}_2)^n$. Denotaremos as *palavras* de $\mathcal{C}$ (que têm todas comprimento n) por $a_1a_2\ldots a_n$. Mais geralmente, podemos considerar códigos definidos sobre um corpo finito arbitrário [3], mas todos os exemplos que consideraremos neste artigo são binários.

Precisemos agora a noção de distância entre duas palavras de $(\mathbb{F}_2)^n$:

> A *distância de Hamming* entre duas palavras $\boldsymbol{a} = a_1a_2\ldots a_n$ e $\boldsymbol{b} = b_1b_2\ldots b_n$, que denotamos por $d(\boldsymbol{a},\boldsymbol{b})$, é o número de índices $i \in \{1,2,\ldots,n\}$ para os quais $a_i \neq b_i$.

Por exemplo, $d(1101,0111)=2$. Note que $d(\boldsymbol{a},\boldsymbol{b})$ indica o número de erros ocorridos quando $\boldsymbol{a}$ é a palavra transmitida e $\boldsymbol{b}$ é a palavra recebida.

> É muito fácil provar que a distância de Hamming satisfaz as propriedades usuais de distância (é uma *métrica* em $(\mathbb{F}_2)^n$): $d(\boldsymbol{a},\boldsymbol{b}) \geq 0$, $d(\boldsymbol{a},\boldsymbol{b}) = 0$ se e só se $\boldsymbol{a} = \boldsymbol{b}$, $d(\boldsymbol{a},\boldsymbol{b}) = d(\boldsymbol{b},\boldsymbol{a})$ e $d(\boldsymbol{a},\boldsymbol{b}) \leq d(\boldsymbol{a},\boldsymbol{c}) + d(\boldsymbol{c},\boldsymbol{b})$. Esta última propriedade é a chamada *desigualdade triangular*.

> A *distância mínima* de um código $\mathcal{C}$, que denotamos por $\delta(\mathcal{C})$, é o número
> $$\min\{d(\boldsymbol{a},\boldsymbol{b}) \mid \boldsymbol{a},\boldsymbol{b} \in \mathcal{C},\ \boldsymbol{a} \neq \boldsymbol{b}\}.$$
> Por exemplo, $\delta(\mathcal{C}_1) = 1$, $\delta(\mathcal{C}_2) = 2$ e $\delta(\mathcal{C}_3) = 3$.

É claro que quanto maior for o valor de $\delta(\mathcal{C})$, mais eficiente será o código. Portanto, um dos objetivos na conceção de um código é que tenha as palavras o mais afastadas entre si. Por outro lado, isto limita o número de palavras do código, logo, limita a sua capacidade de armazenar e transmitir informação. Conciliar estes dois

objetivos de sentido oposto, procurando o ponto de equilíbrio entre eles, é o desafio central da teoria dos códigos.

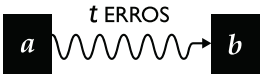
Seja t um número natural. Um código $\mathcal{C}$ *deteta até t erros* se deteta qualquer combinação de t erros em qualquer palavra. Diz-se que $\mathcal{C}$ *corrige até t erros* se corrige qualquer combinação de t erros em qualquer palavra.

A proposição seguinte é um resultado fundamental da teoria algébrica de códigos.

Proposição 3.1. Seja $\mathcal{C}$ um código com distância mínima $\delta(\mathcal{C})$. Então:
(a) $\mathcal{C}$ deteta até t erros se e só se $t \leq \delta(\mathcal{C}) - 1$.
(b) $\mathcal{C}$ corrige até t erros se e só se

$$t \leq \frac{\delta(\mathcal{C}) - 1}{2}.$$

Demonstração. (a) É evidente que existindo duas palavras $\boldsymbol{a}$ e $\boldsymbol{b}$ em $\mathcal{C}$ tais que $d(\boldsymbol{a}, \boldsymbol{b}) = \delta(\mathcal{C})$, se a palavra transmitida for $\boldsymbol{a}$ e acontecerem $\delta(\mathcal{C})$ erros que a transformem em $\boldsymbol{b}$, esses erros nunca serão detetados. Portanto, se $\mathcal{C}$ deteta até t erros então $t < \delta(\mathcal{C})$. Reciprocamente, suponhamos que na transmissão de uma palavra $\boldsymbol{a} \in \mathcal{C}$ ocorreram t erros, resultando na palavra $\boldsymbol{b}$:



(portanto, $d(\boldsymbol{a}, \boldsymbol{b}) = t$). Para provarmos que o código terá a capacidade de detetar o erro, teremos que garantir que $\boldsymbol{b} \notin \mathcal{C}$, o que é fácil: como $d(\boldsymbol{a}, \boldsymbol{b}) = t < \delta(\mathrm{C})$ e $\boldsymbol{a} \in \mathcal{C}$ então $\boldsymbol{b} \notin \mathcal{C}$.

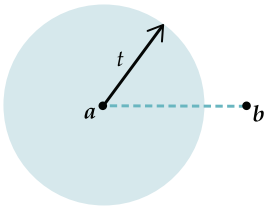
(b) Se $\mathcal{C}$ corrige até t erros, então $2t \leq \delta(\mathcal{C}) - 1$. De facto, $\delta(\mathcal{C}) = 2t$ implicaria a existência de duas palavras $\boldsymbol{a}$ e $\boldsymbol{b}$ diferindo exatamente em $2t$ posições; acontecendo t erros em metade dessas $2t$ posições na transmissão de $\boldsymbol{a}$, nunca seria possível corrigir esses erros, pois poderia ter sido a palavra $\boldsymbol{b}$ a palavra emitida (tendo os t erros ocorrido na outra metade dessas $2t$ posições).

Reciprocamente, suponhamos que na transmissão de uma palavra $\boldsymbol{a} \in \mathcal{C}$ ocorreram t erros, resultando na palavra recebida $\boldsymbol{b}$ (portanto, $d(\boldsymbol{a}, \boldsymbol{b}) = t$). Agora, para provarmos que o código terá a capacidade de corrigir o erro, bastará garantir que mais nenhuma palavra em $\mathcal{C}$ além de $\boldsymbol{a}$ pode ter dado origem à palavra errada $\boldsymbol{b}$, ou seja, que qualquer outra palavra $\boldsymbol{c} \in \mathcal{C}$ está a uma distância de $\boldsymbol{b}$ maior do que t. Isto resulta da desigualdade triangular da distância:

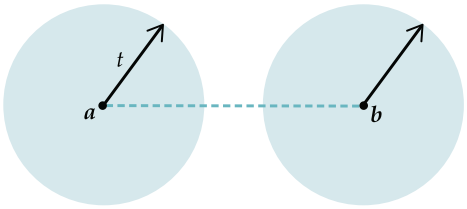
$$d(\boldsymbol{b}, c) \geq d(\boldsymbol{a}, c) - d(\boldsymbol{a}, \boldsymbol{b}) \geq \delta(\mathcal{C}) - t \geq 2t + 1 - t = t + 1.$$

□

Portanto, um código consegue detetar t erros se quaisquer duas palavras do código estiverem a uma distância de Hamming, de pelo menos, $t + 1$:



Por sua vez, um código consegue corrigir até t erros se quaisquer duas palavras do código estiverem a uma distância de Hamming de, pelo menos, $2t + 1$, ou seja, se quaisquer bolas de raio t centradas em palavras distintas forem disjuntas:



Nos exemplos da primeira secção temos:

Código	$\delta(\mathcal{C})$	Tipo de erros que deteta	Tipo de erros que corrige
$\mathcal{C}_1$	1	0	0
$\mathcal{C}_2$	2	singulares	0
$\mathcal{C}_3$	3	duplos	singulares

Suponhamos que, num determinado sistema de comunicação, necessitamos de um código com, no máximo, 2^k palavras. Poderemos então usar todas as palavras binárias $a_1 a_2 \cdots a_k$ de comprimento k. Este código será muito pouco eficiente, uma vez que a distância mínima entre palavras é igual a 1. A proposição 3.1 diz-nos que se quisermos aumentar a eficiência deste código teremos de aumentar a distância mínima entre as suas palavras. Como é que poderemos fazer isso? Muito simplesmente, acrescentando a cada palavra $\boldsymbol{a} = a_1 a_2 \cdots a_k$ um bloco

$$\boldsymbol{c_a} = c_{k+1} \cdots c_n \in (\mathbb{F}_2)^{n-k}$$

tal que $d(\boldsymbol{a}\boldsymbol{c_a}, \boldsymbol{b}\boldsymbol{c_b}) > d(\boldsymbol{a}, \boldsymbol{b})$ para qualquer par de palavras distintas $\boldsymbol{a}, \boldsymbol{b}$.

Os primeiros k símbolos de cada nova palavra

$$\tilde{\boldsymbol{a}} = a_1 a_2 \cdots a_k c_{k+1} \cdots c_n$$

são a *mensagem original* e os $n-k$ símbolos adicionais são os chamados *símbolos de controle*.

Um dos métodos mais utilizados para gerar os símbolos de controle e, por conseguinte, as palavras do código, usa uma matriz binária, i.e., com entradas em $\mathbb{F}_2$,chamada *matriz de controle*,

$$C = (A \mid I_{n-k})$$

onde A é uma matriz $(n-k) \times k$ e I_{n-k} é a matriz identidade de ordem $n-k$. O conjunto $\mathcal{C}$ das palavras é dado pelas soluções do sistema de equações $C\mathbf{a}^T = \mathbf{0}$, sendo $\mathbf{0}$ o vetor nulo de $(\mathbb{F}_2)^{n-k}$. Estes códigos, criados por Hamming nos anos 40 do século passado, chamam-se *códigos (n,k)-lineares* e podem ser estudados usando álgebra linear uma vez que $\mathcal{C}$ é um subespaço vetorial de $(\mathbb{F}_2)^n$ (isto é, qualquer múltiplo de uma palavra é uma palavra do código, a soma de quaisquer duas palavras do código é ainda uma palavra do código). Por exemplo, é fácil ver que, em qualquer código linear $\mathcal{C}$,

$$\delta(\mathcal{C}) = \min\{d(\boldsymbol{a}, \mathbf{0}) \mid \boldsymbol{a} \in \mathbb{C},\ \boldsymbol{a} \neq \mathbf{0}\} \quad \textit{(ou seja, } \delta(\mathcal{C}) \textit{ é igual ao menor número de dígitos não nulos nas palavras diferentes da palavra nula).} \tag{3.2}$$

Para mais informação sobre estes códigos, consulte, por exemplo, o clássico [5] (ainda muito atual) ou os mais recentes[3, 4]. A referência [9] contém muitos dos códigos mais utilizados hoje em dia.

Os códigos $\mathcal{C}_2$ e $\mathcal{C}_3$ da secção anterior são códigos lineares binários: $\mathcal{C}_2$ é um código (4,2)-linear, com matriz de controle

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

enquanto $\mathcal{C}_3$ é um código $(6,2)$-linear, com matriz de controle

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

O código $(7,4)$-linear $\mathcal{C}_4$ definido pela matriz de controle

$$\begin{array}{c} A \\ \left(\begin{array}{cccc|ccc} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right) \end{array}$$

é constituído pelas palavras $\mathbf{a} = a_1\, a_2\, a_3\, a_4\, c_5\, c_6\, c_7$ $(a_i, c_j \in \mathbb{F}_2)$, em que os símbolos de controle c_5, c_6, c_7 podem ser calculados resolvendo o sistema $C\mathbf{a}^T = \mathbf{0}$:

$$\begin{cases} a_1 + a_2 + a_4 + c_5 = 0 \\ a_1 + a_3 + a_4 + c_6 = 0 \\ a_2 + a_3 + a_4 + c_7 = 0 \end{cases} \Leftrightarrow$$

$$\Leftrightarrow \begin{cases} c_5 = a_1 + a_2 + a_4 \\ c_6 = a_1 + a_3 + a_4 \\ c_7 = a_2 + a_3 + a_4 \end{cases}$$

Assim, $\mathcal{C}_4$ é formado pelas 16 palavras $\mathbf{a} = (a_1, a_2, a_3, a_4, a_1+a_2+a_4, a_1+a_3+a_4, a_2+a_3+a_4)$. Trata-se de todos os múltiplos e somas (chamadas combinações lineares) das linhas da *matriz geradora*

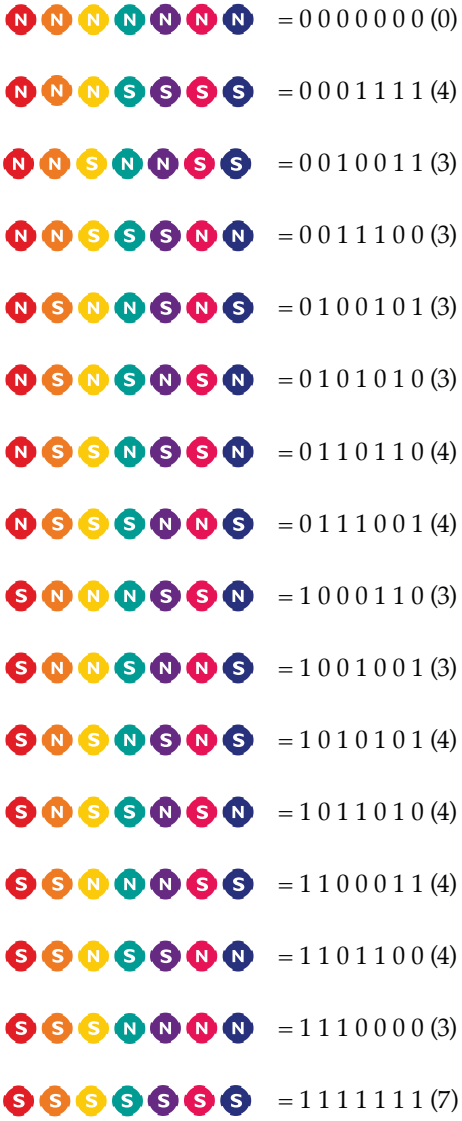
$$G = (I_k \mid A^T) = \begin{array}{c} \\ \left(\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right) \end{array} \begin{array}{c} \\ A^T \end{array}.$$

Da fórmula (3.2) decorre imediatamente que $\delta(\mathcal{C}_4) = 3$, pelo que $\mathcal{C}_4$ corrige erros singulares.

À laia de curiosidade, refira-se que $\mathcal{C}_4$ é precisamente o código que Hamming criou em 1950, nos Laboratórios Bell, para lidar com os erros frequentes cometidos pelo computador Bell Model V, uma das primeiras máquinas da história da computação, na leitura dos cartões perfurados contendo os dados de entrada dos programas. Foi a frustração de Hamming, por ter muitas vezes de reiniciar os seus cálculos no computador, que o levou a dedicar-se à investigação de códigos corretores de erros.

4. EXPLICAÇÃO DO TRUQUE

Estamos agora em condições de perceber o funcionamento do truque com os sete cartões coloridos. Tal como no primeiro truque, cada número de 0 a 15 é representado por uma palavra binária de comprimento 4. Mas agora, como queremos detetar uma possível mentira (corresponde a corrigir um possível erro), precisamos de acrescentar três algarismos de controle de modo a que a distância mínima entre palavras seja 3. O código (7,4)-linear da secção anterior é um exemplo possível e é nele que este truque se baseia. É por isso que o truque usa sete cartões. A disposição dos números nos sete cartões é feita de modo a que cada sequência de sins e nãos corresponda precisamente a uma das 16 palavras do código, que representam os primeiros 16 naturais (a seguir a cada palavra, entre parêntesis, indica-se a distância da palavra à palavra nula, o que confirma que o código tem, de facto, distância mínima 3).

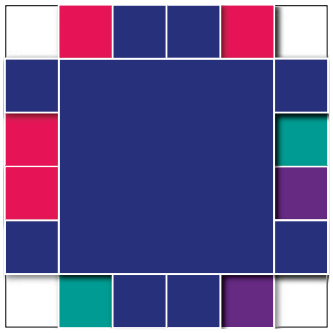


Observe que as palavras correspondentes aos números 1, 2, 4 e 8 são precisamente as linhas da matriz geradora do código e, portanto, o facto enunciado na secção anterior de que as linhas da matriz geradora geram, por combinação linear, todas as outras palavras, corresponde ao facto observado na primeira secção de que os primeiros 16 naturais são múltiplos e somas dos números 1, 2, 4, 8.

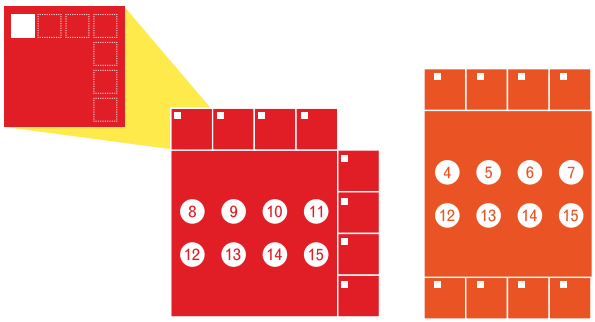
Quanto à explicação para a forma dos cartões de cor, é aquela que, após sobreposição do cartão no cartão-base, permite que fiquem à vista precisamente os números inscritos no cartão de cor (confirme isso, caro leitor, nos cartões da esquerda nas figuras 4 e 5).

5. FACILITANDO A APRESENTAÇÃO DO TRUQUE

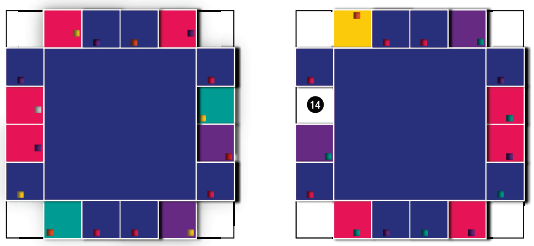
Como vimos, se o voluntário não mentir, no final do truque o número a adivinhar fica à vista no cartão-base. Mas no caso de uma mentira, os números ficam todos cobertos e o número a adivinhar será o que corresponde à palavra que está mais perto de uma das 16 palavras do código, ou seja, aquele que só está coberto por um dos cartões (precisamente o cartão onde o voluntário mentiu). Os restantes 15 números ficam cobertos por, pelo menos, dois cartões. Um mágico muito experiente, conhecendo bem os cartões, conseguirá identificar que número e cartão serão esses (jogando com a espessura dos cartões). No entanto, uma pessoa menos experiente terá dificuldade em apresentar o truque. Por exemplo, se o voluntário escolher o número 14 e o cartão cor-de-rosa e mentir neste, o resultado final será:



Este problema pode ser ultrapassado fazendo uns buraquinhos nos pequenos quadrados de cada cartão. A posição destes buracos terá de variar de cartão para cartão, pelo que precisamos de sete posições distintas: quando colocados um sobre o outro, nenhum par de cartões pode partilhar uma posição comum. Na seguinte figura, mais à esquerda, podemos ver um esquema possível para a colocação dos buracos nos quadrados. Nas duas figuras da direita, podemos ver como posicionar os buracos nos dois primeiros cartões (vermelho e laranja).



Agora, no caso de há pouco (quando o voluntário escolhe o número 14, o cartão cor-de-rosa e mente neste), no final o mágico observa o seguinte (figura da esquerda):



A figura da direita mostra o resultado final na situação análoga em que o voluntário não mente. No primeiro caso, o mágico vê um buraco branco na posição 14 e a cor rosa à volta do buraco, o que significa que é o cartão cor-de-rosa que está imediatamente sobre o cartão-base. No segundo caso, o 14 fica à vista.

Uma simulação computacional deste truque bem como modelos para construir os cartões encontram-se no material suplementar indicado em [6].

6. MAIS UM TRUQUE

A ideia do truque anterior pode estender-se facilmente a um truque que corrija duas mentiras, ou seja, erros duplos [8]. Neste caso, em vez de um só buraco em cada quadrado dos cartões, serão precisos dois buracos ou, equivalentemente, duas ranhuras transversais (ver cartões na figura 6) que permitam que uma segunda cor fique visível quando os cartões são empilhados sobre o cartão-base. No final, duas ranhuras brancas aparecerão na posição do número a adivinhar.

Neste truque, são mostrados à assistência 10 cartões (figura 6) e o mágico tem ainda consigo o cartão-base branco. Um voluntário deverá escolher um número entre 0 e 7 bem como duas das cores dos cartões.

Tal como anteriormente, a pessoa terá de dizer se o número em que pensou está em cada cartão, podendo desta vez mentir em dois dos 10 cartões. O mágico adivinhará o número e os dois cartões.

Caberá agora ao leitor descobrir qual o código corretor de erros que faz funcionar o truque (o código terá oito palavras correspondentes aos oito números, a sua matriz geradora terá de ser uma matriz 3×8 e a sua

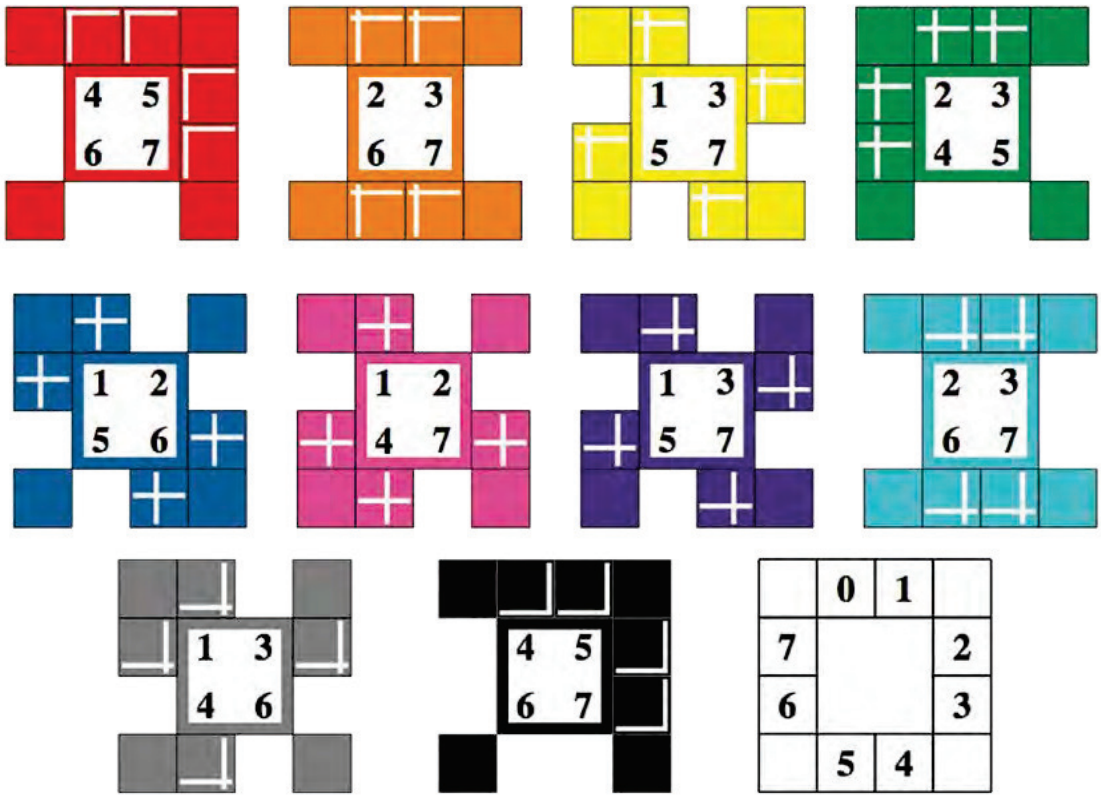


Figura 6: Cartões do terceiro truque.

distância mínima terá de ser 5 para que detete de facto erros duplos).

Para mais truques baseados noutros códigos correctores de erros (sobre outros corpos finitos) consultar [7] e *mthsc.clemson.edu/misc/MAM_2014/MagicErrorCorrectingCodes.html*.

7. REFERÊNCIAS

[1] R. Ehrenborg, "Decoding the Hamming code", *Math Horizons* 13 (Abril, 2006), 16-17.

[2] R. W. Hamming, "Error detecting and error correcting codes", *Bell System Tech. J.* 29 (1950), 147-160.

[3] R. Lidl e H. Niederreiter, *Introduction to Finite Fields and their Applications*, Cambridge University Press, 2000.

[4] S. Ling e C. Xing, *Coding Theory: A First Course*, Cambridge University Press, 2004.

[5] F. J. MacWilliams e N. J. A. Sloane, *The Theory of Error-Correcting Codes*, New York: Elsevier, 1978.

[6] T. Mateer, "A magic trick based on the Hamming Code", *Math Horizons* 21 (Novembro, 2013), 9-11. Material suplementar em: *www.maa.org/publications/periodicals/math-horizons/math-horizons-supplements/supplements-to-a-magic-trick-based-on-the-hamming-code*.

[7] T. Mateer, "A Reed-Solomon code magig trick", *Math. Magazine* 87 (2014) 125-131.

[8] T. Mateer, "A magic trick based on a double error-correcting code", *mthsc.clemson.edu/misc/MAM_2014/ doublemagic.pdf.* Consultado em Agosto de 2014.

[9] J. Picado e H. Pinto, "Sistemas de identificação com algarismos de controlo", em: *Atractor - Matemática Interactiva, www.atractor.pt/mat/alg_controlo*, 2006.

[10] C. E. Shannon, "A mathematical theory of communication I, II", *Bell System Tech. J.*] 27 (1948), 379-423, 623-656.

SOBRE OS AUTORES

Estefani Moraes Moreira é estudante do último ano da Licenciatura em Matemática na Universidade Federal de São Carlos (UFSCar), Brasil. Estudou na Universidade de Coimbra, em dois anos letivos (2012/2013 e 2013/2014), por meio do Programa de Licenciaturas Internacionais (PLI). Interessa-se por jogos matemáticos e também por história da matemática.

Jorge Picado é professor associado do Departamento de Matemática da Universidade de Coimbra (UC). É doutorado e agregado em Matemática Pura pela UC. É membro do Centro de Matemática da UC, onde investiga no grupo de Álgebra, Lógica e Topologia. A sua área de especialização é a topologia "sem pontos". É co-autor do livro *Frames and Locales: topology without points* (Springer, 2012), onde se tenta explicar as ideias fundamentais desta abordagem *locálica* à topologia.