# O quebra-cabeça chinês das argolas

Carlos A. Martinhon

Universidade Federal Fluminense, Departamento de Ciência da Computação
Rua Passo da Pátria, 156 - Bloco E - 3 andar - Boa Viagem
24210-240, Niterói, RJ, Brasil
*e-mail*: mart@dcc.ic.uff.br

**Resumo** O quebra-cabeça chinês das argolas consiste em um conjunto de fixo de argolas presas caprichosamente à uma haste fixa de metal. Obedecendo-se a um conjunto de regras bem definidas, o objetivo é determinar a sequência correta de inserções e remoções das argolas de maneira que todas as argolas presas à haste de metal sejam retiradas. Apesar de aparentemente complicado, a manipulacão das argolas pode ser representada convenientemente através de uma sequência de dígitos binários 0-1 correspondendo, respectivamente, aos movimentos de remoção e inserção das argolas. Neste trabalho, estudamos um procedimento recursivo para a resolução deste problema e calculamos o total de movimentos em funcão da quantidade de argolas. Veremos que este interessante quebra-cabeça representa uma situação particular dos Códigos de Gray, conhecido na literatura por suas importantes aplicações em códigos corretores de erros, compressão de dados, processamento de imagens, entre outras.

**Palavras Chaves:** Anéis de Cardano, Recursividade, Indução, Códigos de Gray.

## 1 Introdução

O quebra-cabeça chinês das argolas também conhecido como o problema dos anéis de Cardano foi trazido à Europa no século XVI quando Girolamo Cardano o apresentou pela primeira vez em seu livro *De Subtilitate*, publicado em 1550. Conta a lenda que ele foi desenvolvido por um general chinês que o deu de presente à sua esposa para ocupá-la enquanto ele estivesse fora em combate. O quebra-cabeça consiste em um conjunto de fixo de argolas presas caprichosamente à uma haste fixa de metal. Obedecendo-se a um conjunto de regras bem definidas, o objetivo é determinar a sequência correta de inserções e remoções das argolas de maneira que todas as argolas presas à haste de metal sejam retiradas (veja as Figuras (1.a) e (1.b)).

O quebra-cabeça chinês das argolas pode ser resolvido através de um procedimento recursivo e se enquadra na mesma categoria de um outro famoso quebra-cabeça conhecido como Torres de Hanói, criado por Edouard Lucas em 1883 [2]. Assim, como o problema das Torres de Hanói, o número de movimentos (inserções e remoções) no quebra-cabeça chinês das argolas cresce exponencialmente com o número de argolas. No caso de um conjunto com $n = 7$ argolas,

serão necessários 85 movimentos para a separação completa da haste de metal do conjunto de argolas. Se $n = 8$ este numero sobe para 170. Para $n = 9$ ele sobe para 341, e assim sucessivamente. Uma questão interessante é determinar o número total de movimentos em função do número $n$ de argolas. Pode-se mostrar, por exemplo, que para $n = 65$ esse número sobe para espantosos 18.446.744.073.709.551.616 movimentos! Como discutido em [11], assumindo-se que haja uma pessoa bastante habilidosa e que seja capaz de executar um movimento por segundo, seriam necessários, pelo menos, 56 bilhões de anos para resolvê-lo completamente, ou seja, quase 4 vezes a idade estimada do universo que é de aproximadamente 13 bilhões de anos! Infelizmente, a lenda não relata com quantas argolas era composto o primeiro quebra-cabeça, e portanto, não podemos determinar exatamente o quão difícil era a tarefa a ser confrontada pela esposa do general chinês...

Apesar de aparentemente complicado, a manipulacão das argolas pode ser representada convenientemente através de uma sequência de dígitos binários 0-1 correspondendo, respectivamente, aos movimentos de remoção e inserção das argolas. A solução do quebra-cabeça está intimamente ligada aos *Códigos de Gray* introduzida na década de 1930 pelo engenheiro Frank Gray. Em um código de Gray binário, sequências consecutivas diferem apenas em um único *bit*. Esse tipo de comportamento está presente também no quebra-cabeça chinês das argolas, já que as regras existentes para movimentação das argolas (discutidas mais adiante) impedem que duas ou mais argolas sejam inseridas ou removidas simultaneamente. Os códigos de Gray podem ser encontrados em inúmeras aplicações importantes, como na compressão de dados [8,4], processamento de sinais [7], estrutura de dados [6], entre outras. Finalmente, para uma leitura mais geral dedicada à discussão e solução deste e outros quebra-cabeças interessantes consulte o livro de R. Ball [1].

## 2 Resolvendo o quebra-cabeça

No quebra-cabeça chinês das argolas, a compreensão dos movimentos de inserção e retirada das argolas é razoavelmente complicada, mesmo para quem possa manipulá-lo efetivamente! Você poderá construir facilmente um quebra-cabeça (como nas Figuras (1.a) e (1.b)) dispondo-se apenas de um fio de arame e pedaços de madeira. Tente construir um e divirta-se! Uma outra alternativa bastante simples e que possibilita uma compreensão rápida do problema é através de uma representação binária conveniente. Assim, na sequência binária $s = (0, 1, 0, 0, 0, 0, 1)$ apenas a primeira e anti-penúltima argolas (da direita para a esquerda) estarão presas à haste de metal. Em Megalakaki e Tijus [10], os autores discutem alguns aspectos interessantes envolvidos no processo de aprendizado a partir de uma representação matemática conveniente.

As Figuras Figuras 1.a e 1.b ilustram as sequências inicial e final, respectivamente. Partindo-se da sequência inicial, os movimentos de inserção e remoção das argolas deverão obedecer às seguintes regras:

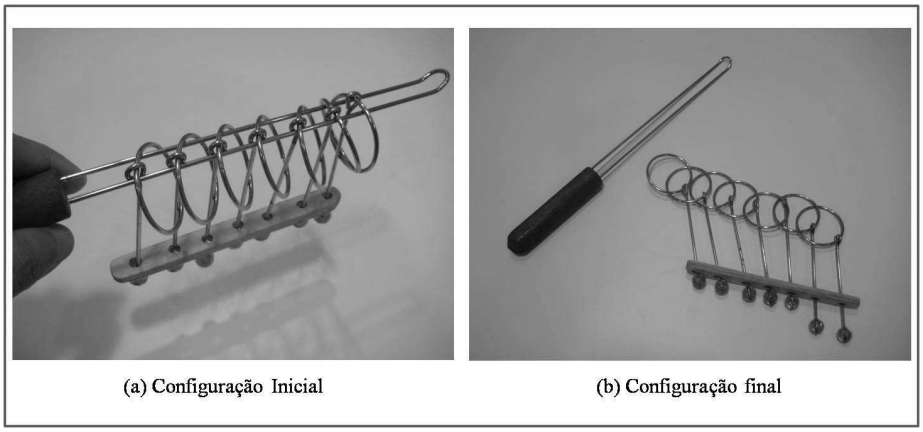


**Figura 1.** (a) Configuracão inicial com $n = 7$ argolas $s_i = (1, 1, 1, 1, 1, 1, 1)$. (b)Configuracão final com $n = 7$ argolas $s_f = (0, 0, 0, 0, 0, 0, 0)$.

1. A argola 1 (primeira argola) pode ser inserida ou retirada a qualquer momento;
2. Partindo-se da direita para a esquerda, pode-se inserir ou retirar a segunda argola logo após primeira argola presa à haste de metal;

Para melhor exemplificar as regras 1 e 2 acima, considere a seguinte sequência arbitrária com 7 argolas: $s = (0, 1, 1, 1, 0, 0, 0)$. Quais os próximos movimentos a serem executados? Da regra 1, conclui-se que $s' = (0, 1, 1, 1, 0, 0, 1)$, ou seja, a primeira argola é inserida. Da regra 2, conclui-se que $s$" $= (0, 1, 0, 1, 0, 0, 0)$, ou seja, a segunda argola logo após a primeira argola presa à haste de metal é retirada (sempre olhando-se da direita para a esquerda).

Portanto, dado um quebra-cabeça com $n$ argolas, todos presas à haste de metal, o procedimento recursivo $RETIRA(n)$ a seguir exibe a sequência de movimentos necessários para a remoção completa das $n$ argolas. Como movimentos de inserção de argolas também deverão ser executados de maneira combinada, constrói-se um segundo procedimento $INSERE(n)$, representando a inserção de $n$ argolas. Um estudo mais detalhado de rotinas combinadas, ou co-rotinas pode ser encontrado em [5]. Quando uma co-rotina $A$ evoca uma outra co-rotina $B$, a ação de $A$ é suspensa temporariamente até que a ação requisitada em $B$ seja concluída.

No exemplo a seguir, são representados todos os movimentos necessários para a remoção completa de um conjunto com $n = 5$ argolas. Veja o procedimento $RETIRA(n)$.

– **Passo** 7: Retira $n - 2$ argolas
  $s_0 = (1, 1, 1, 1, 1)$, $s_1 = (1, 1, 1, 1, 0)$, $s_2 = (1, 1, 0, 1, 0)$, $s_3 = (1, 1, 0, 1, 1)$, $s_4 = (1, 1, 0, 0, 1)$, $s_5 = (1, 1, 0, 0, 0)$;
– **Passo** 8: Retira $n$-ésima argola:
  $s_6 = (0, 1, 0, 0, 0)$;

**Procedimento 1** : RETIRA(n)

1: **se** $n = 1$ **então**
2: Remove a *primeira* argola;
3: **fim se**
4: **se** $n = 2$ **então**
5: Remove a *segunda* e a *primeira* argolas (nesta ordem);
6: **se não**
7: RETIRA$(n-2)$;
8: Remove *n-ésima* argola;
9: INSERE$(n-2)$;
10: RETIRA$(n-1)$;
11: **fim se**

**Procedimento 2** : INSERE(n)

1: **se** $n = 1$ **então**
2: Insire a *primeira* argola;
3: **fim se**
4: **se** $n = 2$ **então**
5: Insire a *primeira* e *segunda* argolas (nesta ordem);
6: **se não**
7: INSERE$(n-1)$;
8: RETIRA$(n-2)$;
9: Insire a *n-ésima* argola;
10: INSERE$(n-2)$;
11: **fim se**

- **Passo** 9: Insere $n-2$ argolas:
  $s_7 = (0,1,0,0,1)$, $s_8 = (0,1,0,1,1)$, $s_9 = (0,1,0,1,0)$, $s_{10} = (0,1,1,1,0)$, $s_{11} = (0,1,1,1,1)$;
- **Passo** 10: Retira $n-1$ argolas:
  $s_{12} = (0,1,1,0,1)$, $s_{13} = (0,1,1,0,0)$, $s_{14} = (0,0,1,0,0)$, $s_{15} = (0,0,1,0,1)$, $s_{16} = (0,0,1,1,1)$, $s_{17} = (0,0,1,1,0)$, $s_{18} = (0,0,0,1,0)$, $s_{19} = (0,0,0,1,1)$, $s_{20} = (0,0,0,0,1)$, $s_{21} = (0,0,0,0,0)$;

No passo 7 do procedimento $RETIRA(n-2)$, são descritos todos os movimentos para a retirada completa das 3 primeiras argolas (sempre da direita para a esquerda). Em seguida a *n-ésima* argola é removida no passo 8 (argola 5). No passo 9 são inseridas novamente as primeiras 3 argolas. Finalmente, no passo 10 (procedimento $RETIRA(n-1)$), são descritos todos os movimentos para a retirada completa das 4 argolas restantes. Cada uma das etapas $7, 9$ e $10$ é executada recursivamente. Note que serão necessários 21 movimentos para a retirada completa das 5 argolas (sequência $s_{21}$).

## 3 Calculando o total de movimentos

Considere $T_I(n)$ e $T_R(n)$, respectivamente, o número total de movimentos para a inserção e remoção de $n$ argolas como descrito nos procedimentos (ou co-rotinas) 1 e 2 acima. Do procedimento *RETIRA(n)*, é fácil ver que o total de movimentos para a retirada completa das $n$ argolas, denotado por $T_R(n)$, pode ser expresso pela seguinte relação de recorrência:

$$\begin{aligned} &T_R(0) = 0 \\ &T_R(1) = 1 \\ &T_R(n) = T_R(n-1) + T_R(n-2) + T_I(n-2) + 1, \forall\ n \geq 2. \end{aligned} \tag{1}$$

Analogamente, do procedimento *INSERE(n)* acima, o total de movimentos $T_I(n)$ para inserção de $n$ argolas pode ser expresso por:

$$\begin{aligned} &T_I(0) = 0 \\ &T_I(1) = 1 \\ &T_I(n) = T_I(n-1) + T_I(n-2) + T_R(n-2) + 1, \forall\ n \geq 2. \end{aligned} \tag{2}$$

Note que os valores $T_I(n)$ e $T_R(n)$ acima serão idênticos. Isto pode ser constatado facilmente executando-se toda a sequência de movimentos de retirada das $n$ argolas no sentido inverso. Formalmente, ao subtrair a expressão (2) de (1) obtêm-se que: $T_R(n) - T_I(n) = T_R(n-1) - T_I(n-1) = T_R(n-2) - T_I(n-2) = \ldots = T_R(1) - T_I(1) = 1 - 1 = 0$, e portanto, $T_R(n) = T_I(n), \forall n \geq 1$. Desta forma, fazendo $T(n) = T_R(n) = T_I(n)$ chega-se à seguinte uma relação de recorrência para o número total de movimentos (inserções/remoções):

$$T(0) = 0 \tag{3}$$
$$T(1) = 1 \tag{4}$$
$$T(n) = T(n-1) + 2T(n-2) + 1, \forall\ n \geq 2. \tag{5}$$

A tabela seguinte ilustra o total de movimentos para diferentes valores de $n$. Eles podem ser obtidos substituindo-se simplesmente os diferentes valores de $n$ na relação de recorrência (5) acima:

$$\begin{aligned} &T(0) = 0 \\ &T(1) = 1 \\ &T(2) = T(1) + 2T(0) + 1 = 2 \\ &T(3) = T(2) + 2T(1) + 1 = 5 \\ &T(4) = T(3) + 2T(2) + 1 = 10 \\ &T(5) = T(4) + 2T(3) + 1 = 21 \end{aligned}$$

$$
\begin{aligned}
T(6) &= T(5) + 2T(4) + 1 = 42 \\
T(7) &= T(6) + 2T(5) + 1 = 85 \\
T(8) &= T(7) + 2T(6) + 1 = 170 \\
T(9) &= T(8) + 2T(7) + 1 = 341 \\
&\ldots \qquad\qquad \ldots\ldots \qquad\qquad \ldots
\end{aligned}
$$

Ao desenvolver a relação de recorrência (5) acima, apesar de factível, fica difícil estabelecer diretamente uma única lei de formação que seja válida para todo inteiro $n$ positivo (tente fazê-lo como exercício!). Entretanto, observando-se mais atentamente o total de movimentos em funcão de $n$, pode-se determinar duas relações de recorrência distintas para $n$ par e ímpar separadamente. Dessa forma, pode-se chegar naturalmente à seguinte conjectura:

$$
\begin{aligned}
&T(0) = 0 && (6)\\
&T(n) = 2T(n-1) + 1, \quad \texttt{para } n \texttt{ ímpar} && (7)\\
&T(n) = 2T(n-1), \qquad \texttt{para } n \texttt{ par.} && (8)
\end{aligned}
$$

Para provar que as recorrências (7) e (8) são válidas para todo inteiro positivo $n$ basta que se construa uma prova por indução dividindo-a em $n$ par e $n$ ímpar, respectivamente. Considere então a seguinte prova:

**Prova por indução:** Inicialmente, é fácil ver por inspeção que $T(0) = 0$ e $T(1) = 1$ na base da indução.

Para provar que as recorrências (7) e (8) são verdadeiras para todo $n$ ímpar e par, respectivamente, assume-se primeiramente que (7) e (8) sejam verdadeiras para $n - 1$. Dessa forma, para $n \geq 2$ pode-se afirmar que:

$$
\begin{aligned}
&T(n-1) = 2T(n-2) + 1, \quad \texttt{para } n-1 \texttt{ ímpar} && (9)\\
&T(n-1) = 2T(n-2), \qquad \texttt{para } n-1 \texttt{ par.} && (10)
\end{aligned}
$$

Note agora da igualdade (5) acima que:

$$T(n) = T(n-1) + 2T(n-2) + 1, \forall\ n \geq 2.$$

Assim, se $n$ é ímpar (e portanto $n-1$ é par), conclui-se após a substituição de (10) em (5) e da base da indução (pois $T(1) = 1$) que $T(n) = 2T(n-1)+1$, para todo $n$ ímpar. Analogamente, para $n$ par (e portanto $n-1$ ímpar), conclui-se após a substituição de (9) em (5) e da base da indução (pois $T(0) = 0$) que $T(n) = 2T(n-1)$ para todo $n$ par, o que completa a prova por indução. ■

Analogamente, o cálculo do total de movimentos de inserção ou remoção de $n$ argolas pode ser feito para $n$ par e ímpar, separadamente. Assim:

$1^o$ **Caso:** ($n$ `par`). Inicialmente de (6) e (8) tem-se, respectivamente, $T(0) = 0$ e $T(n) = 2T(n-1)$ para $n$ par. Substituindo-se agora $T(n-1) = 2T(n-2)+1$ na recorrência (8) (pois $n-1$ é impar), conclui-se que: $T(n) = 2(2T(n-2)+1)$. Assim, para $n$ par, tem-se a seguinte relação de recorrência descrita apenas em função de valores pares:

$$\begin{aligned} &T(0) = 0, \texttt{ base da recursão} \\ &T(n) = 4T(n-2) + 2, \texttt{ para todo } n \geq 2. \end{aligned} \tag{11}$$

Agora, desenvolvendo (11) obtêm-se:

$$\begin{aligned} &T(n) = 4(4T(n-4)+2)+2 = 4^2T(n-4)+2.4+2 \Rightarrow \\ &T(n) = 4^2(4T(n-6)+2)+2.4+2 = 4^3T(n-6)+2.4^2+2.4+2. \end{aligned} \tag{12}$$

Após $k$ passos chega-se a:

$$T(n) = 4^kT(n-2k) + 2\sum_{i=0}^{k-1} 4^i \tag{13}$$

Ao final do processo espera-se que $n-2k = 0$ (base da recursão). Assim:

$$T(n) = 4^k.T(0) + 2\sum_{i=0}^{k-1} 4^i = 2\sum_{i=0}^{k-1} 4^i. \tag{14}$$

Multiplicando-se agora ambos os lados da igualdade (14) por $(4-1)$ obtêm-se facilmente que $3T(n) = 2(4^k-1)$ (verifique![1]). Como $k = \frac{n}{2}$, chega-se finalmente à expressão desejada:

$$T(n) = 2\left(\frac{4^{\frac{n}{2}}-1}{3}\right) = 2\left(\frac{2^n-1}{3}\right) \Rightarrow T(n) = \frac{2^{n+1}-2}{3}, \tag{15}$$
$$\texttt{para } n \texttt{ par.}$$

$2^o$ **Caso:** ($n$ `ímpar`). Analogamente, de (6) e (7) tem-se $T(0) = 0$ e $T(n) = 2T(n-1)+1$ para $n$ ímpar, e portanto, $T(1) = 1$ para $n = 1$. Agora, após a substituição de $T(n-1) = 2T(n-2)$ em (7) (pois $n-1$ é par), conclui-se que: $T(n) = 2(2T(n-2))+1$, para todo $n$ ímpar maior ou igual que 3. Portanto:

$$\begin{aligned} &T(1) = 1, \texttt{ base da recursão} \\ &T(n) = 4T(n-2)+1, \texttt{ para } n \texttt{ ímpar maior ou igual que } 3 \end{aligned} \tag{16}$$

[1] De maneira geral, para se resolver a série $S = \sum_{i=0}^{n} a^i$ para $a > 1$, basta multiplicar ambos os lados desta igualdade por $a-1$.

Desenvolvendo-se esta recorrência:

$$T(n) = 4(4T(n-4)+1)+1 = 4^2T(n-4)+4+1 \Rightarrow$$
$$T(n) = 4^2(4T(n-6)+1)+4+1 = 4^3T(n-6)+4^2+4+1. \quad (17)$$

Após $k$ passos obtêm-se:

$$T(n) = 4^kT(n-2k)+\sum_{i=0}^{k-1}4^i \quad (18)$$

Ao final do processo espera-se que $n-2k=1$ (base da recursão). Assim:

$$T(n) = 4^k.T(1)+\sum_{i=0}^{k-1}4^i = \sum_{i=0}^{k}4^i. \quad (19)$$

Multiplicando-se ambos os lados da igualdade por $(4-1)$ tem-se que $3T(n) = 4^{k+1}-1$. Como $k=\frac{n-1}{2}$ (já que $n-2k=1$) conclui-se finalmente que:

$$T(n) = \frac{4^{k+1}-1}{3} = \frac{2^{2k+2}-1}{3} \Rightarrow T(n) = \frac{2^{n+1}-1}{3}, \quad (20)$$
$$\texttt{para todo } n \texttt{ ímpar.}$$

Uma vez concluída a análise $n$ par e ímpar separadamente, pode-se determinar facilmente a seguinte fórmula geral:

$$T(n) = \frac{2^{n+1}-2+\left(\frac{(-1)^{n+1}+1}{2}\right)}{3}, \texttt{ para todo } n \geq 0.$$

## 4 Códigos de Gray

O quebra-cabeça chinês das argolas pode ser visto como caso particular de uma teria mais geral denominada Códigos de Gray. Em um código de Gray binário com $n$ bits, são geradas todas as $2^n$ sequências possíveis com $n$ bits de maneira que duas sequências consecutivas diferem apenas em um único *bit*.

Considere novamente o quebra-cabeça como descrito anteriormente juntamente com a sequência inicial $s_i = (1,0,\ldots,0)$ com $n$ *bits*. Neste caso, apenas a $n$-ésima argola (ou última argola) esta presa à haste de metal. Note que para a separação completa da haste de metal do conjunto de argolas será necessário que se insira novamente as primeiras $n-1$ argolas (a retirada direta da $n$-ésima argola não é possível por violar as regras 1 e 2 descritas acima, vide Seção 2).

Isto pode ser implementado simplesmente através da execução dos procedimentos $INSERE(n-1)$ seguido do procedimento $RETIRA(n)$ como acima. Desta forma, o número total de movimentos para $n$ par será expresso por:

$$T(n) = \frac{2^n - 1}{3} + \frac{2^{n+1} - 2}{3} = 2^n - 1. \tag{21}$$

Da mesma forma, para $n$ ímpar obtêm-se:

$$T(n) = \frac{2^n - 2}{3} + \frac{2^{n+1} - 1}{3} = 2^n - 1. \tag{22}$$

Portanto, partindo-se de $s_i = (1, 0, \ldots, 0)$, seriam necessários $T(n) = 2^n - 1$ movimentos para a separação completa da haste de metal do conjunto de $n \geq 1$ argolas. A Tabela 1 ilustra uma sequência completa de movimentos (representada de trás para a frente) juntamente com a representação decimal e binária para um exemplo com $n = 4$ *bits*. Note que todas as $2^n$ sequências são geradas neste caso. Ao contrário da representação binária, nos códigos de Gray duas sequências consecutivas sempre diferem em um único *bit*. Isto ocorre mesmo para a primeira e a última sequência da cadeia, sequências $s_0 = (0, 0, 0, 0)$ e $s_{15} = (1, 0, 0, 0)$ respectivamente.

**Tabela 1.** Comparação entre a representação decimal, binária e código de Gray para todas as sequências com 4 dígitos

| *Decimal* | **código binário** | **código de Gray** |
|---|---|---|
| 00 | 0000 | 0000 |
| 01 | 0001 | 0001 |
| 02 | 0010 | 0011 |
| 03 | 0011 | 0010 |
| 04 | 0100 | 0110 |
| 05 | 0101 | 0111 |
| 06 | 0110 | 0101 |
| 07 | 0111 | 0100 |
| 08 | 1000 | 1100 |
| 09 | 1001 | 1101 |
| 10 | 1010 | 1111 |
| 11 | 1011 | 1110 |
| 12 | 1100 | 1010 |
| 13 | 1101 | 1011 |
| 14 | 1110 | 1001 |
| 15 | 1111 | 1000 |

Uma questão interessante é a conversão da representação binária para código de Gray e vice-versa (tente fazê-lo como exercício!). Um estudo mais detalhado

envolvendo códigos de Gray foge ao escopo deste trabalho. Entretanto, para aqueles leitores interessados, aplicações e demais questões envolvendo códigos de Gray podem ser encontradas em [3,5,9].

## Referências


1. W. W. R. Ball. *Mathematical Recreations and Essays*, Macmillan, 305-310, New York, 1947.
2. A. K. Dewdney. *Computer Recreations: Yin and yang: recursion and iteration, the Towers of Hanoi and the Chinese Rings*, Scientific American, 251, no. 5, 1928. Reprinted, with Addendum, in The Armchair Universe: An Exploration of Computer Worlds, W. H. Freeman and Co., New York, pp. 186-199, 1988.
3. R. W. Doran. *The Gray Code*, Journal of Universal Computer Science, vol. 13, n. 11, 1573-1597, 2007.
4. D. Dimitrov, T.Dvorak, P. Gregor, R. Screkovski. *The Gray code compression*, Preprint series, Vol. 47, 1085, 2009.
5. D. E. Knuth, F. Ruskey. *Efficient Coroutine Generation of Constrained Gray Sequences*, Tech. Report, dedicated to the memory of Ole-Johan Dahl), 2004.
6. M. Loeb, A. L. Tharp. *Gray Code Chaining: A High Performance Hashing Algorithm for Limited Storage Applications*, International Conference on Information Technology (ITNG'07), IEEE, 2007.
7. G. S. Manku, J. Sawada. *A Loopless Gray Code for Minimal Signed-Binary Representations*, Lecture Notes in Computer Science, V.3669, pp. 438447, 2005.
8. D. Richards. *Data compression and Gray-code sorting*, Information Processing Letters, 22, 201-205, 1986.
9. C. Savage. *A survey of combinatorial Gray codes*, SIAM Review, 39, 605-629,1997.
10. C. Tijus, O. Megalakaki. *Exploration et planification dans le problème des anneaux chinois : la découverte des règles à partir des propriétés*, L'année psychologique, V. 105, N. 4, p. 625-647, 2005.
11. M. Wertheim *Celebrating Puzzles, in* $18,446,744,073,709,551,616$ *Moves (or So)*, New York Times, July 28, 2006.