UNIVERSIDADE DO VALE DO RIO DOS SINOS - UNISINOS
UNIDADE ACADÊMICA DE GRADUAÇÃO
LICENCIATURA EM MATEMÁTICA

ISAAC CANABARRO NUNES

APLICAÇÃO DA ÁLGEBRA LINEAR NA RESOLUÇÃO DO JOGO LIGHTS OUT

SÃO LEOPOLDO

2023

Isaac Canabarro Nunes

APLICAÇÃO DA ÁLGEBRA LINEAR NA RESOLUÇÃO DO JOGO LIGHTS OUT

Trabalho de Conclusão de Curso apresentado como requisito parcial para a obtenção do título de Licenciado em Matemática, pelo Curso de Licenciatura em Matemática da Universidade do Vale do Rio dos Sinos - UNISINOS.

Orientador: Prof. Dr. Rogério Ricardo Steffenon

São Leopoldo

2023

Resumo

O presente trabalho propõe abordar, através da perspectiva Matemática o jogo Lights Out que consiste em uma tabela de formato retangular em que há luzes ligadas e desligadas, o objetivo é desligar todas. E para isso o trabalho, inicialmente, estabelece uma base de conhecimentos e resultados prévios, tais como: tópicos de Álgebra Linear e operações com Classes Residuais. Posteriormente, há a modelagem matemática do jogo, transformando-o em uma equação matricial e a partir de resultados conhecidos de Álgebra Linear foi possível demonstrar que na configuração $2 \times 2n$, $n \in \mathbb{N}$ sempre admite solução única. Ademais, foi desenvolvido um programa em Python que permite encontrar a solução para qualquer configuração inicial dada.

Palavras-chaves: Ligths Out. Álgebra Linear. Equação Matricial.

Sumário

1 Introdução

O presente trabalho de pesquisa tem por finalidade estudar sobre aplicações de tópicos de Álgebra Linear na resolução do Jogo *Lights-Out* na configuração $2 \times n$, onde $n \in \mathbb{N}$. Para contribuir com essa investigação, será utilizado a programação em Python que é uma ferramenta de suma importância em diversos campos atualmente. Ademais, vale salientar que o problema de pesquisa é "dada qualquer configuração de luzes no Jogo *Lights-Out* de tamanho $2 \times n$ em que $n \in \mathbb{N}$, é possível construir estratégias, a fim de resolver o jogo?".

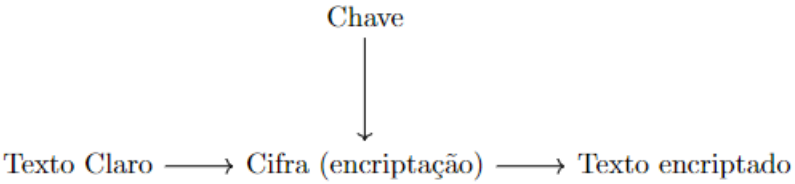
1.1 Justificativa

A Álgebra Linear é um campo da Matemática deveras importante que trata de matrizes, vetores, sistemas lineares, transformações lineares, etc. Apesar de apresentar conceitos extremamente abstratos ela fornece uma gama de aplicações em diferentes áreas, tais como: engenharia, economia, computação.

Faz-se necessário evidenciar que essa pesquisa irá abordar predominantemente a solução de sistemas lineares no corpo dos inteiros módulo 2, denotado por $\mathbb{Z}_2$, que será abordado com ênfase no decorrer do texto. Um fato interessante, é que a solução de sistemas em $\mathbb{Z}_2$ é de grande importância para o ramo da criptografia e da criptoanálise.

De acordo com (PELLEGRINI, 2022) na Criptoanálise, a cifra que é uma ferramenta criptográfica usada para garantir sigilo em comunicações, isto é, em uma mensagem qualquer, representada por uma sequência de bits [1] , ou seja, uma sequência de 0's e 1's, é misturada a chave secreta (que é outra sequência de bits), de forma que um intruso não possa identificar mais a mensagem (e nem possa, é claro, obter a chave secreta). Quando a mensagem chegar ao destinatário, a chave secreta é novamente usada para decodificar a mensagem.

[1] As sequências de bits, são sequências do espaço $\mathbb{Z}_2$, isto é, formadas por 0's e 1's. Outro ponto importante, as sequências de n bits, são sequências com n termos.



Fonte: PELLEGRINI, 2022.

Figura 1 – Diagrama representando o processo de encriptação.

O método da criptoanálise algébrica consiste em representar um criptossistema como um sistema de equações, que no caso será um sistema linear em $\mathbb{Z}_2$ e a solução desse sistema poderá ser uma chave ou mensagem secreta. (PELLEGRINI, 2022).

Portanto, percebe-se que o potencial de estudarmos essa área é gigantesco, uma vez que a capacidade de guardar informações é imprescíndivel.

1.2 Objetivos

O objetivo geral deste trabalho é elaborar um programa em Python que permita conjecturar em quais configurações o Jogo *Lights-Out* de tamanho $2 \times n$ possui solução, posteriormente demonstrar analiticamente tal conjectura. Os objetivos específicos para se chegar a isto são:

1) Expor conceitos relacionados com a existência de solução de sistemas lineares.

2) Exibir tópicos de Aritmética Modular, com ênfase em $\mathbb{Z}_2$, corpo dos inteiros módulo 2.

3) Verificar a existência de solução em diferentes variações do Jogo *Lights-Out* utilizando tópicos de Álgebra Linear.

4) Utillizar programação em Python para mostrar as soluções.

1.3 Organização do Texto

Serão apresentados no capítulo 2 uma pequena contextualização histórica sobre sistemas lineares e determinantes. A fim de criar uma base de conhecimentos para a compreensão do foco da pesquisa no capítulo 3 e 4, vão ser apresentados resultados sobre sistemas lineres e classes residuais, respectivamente. Mais especificamente, o capítulo 3 trata de sistemas lineares e métodos para sua resolução além de alguns tópicos etc. Também vamos evidenciar tópicos sobre conjuntos geradores e matrizes, em que o objetivo é exibir os resultados que permitem associar a resolução de uma equação matricial, com a dimensão do

espaço nulo. Como os elementos das matrizes aqui estudadas estão em $\mathbb{Z}_2$, iremos estudar as operações algébricas em $\mathbb{Z}_2$. Antes de chegar ao centro da pesquisa, será comentado no capítulo 5 de forma sucinta sobre Python, visto que utilizaremos essa linguagem de programação para encontrar as soluções.

Finalmente, chegamos ao foco desse trabalho, capítulo 6, onde vamos explicar como o jogo funciona e, posteriormente, modelá-lo, para que seja possível transformar o jogo em uma equação matricial. Depois da modelagem trataremos de dois casos específicos, o jogo nas configurações 2×2 e 5×5, em que o segundo dará o ponta pé inicial de como podemos trabalhar com a equação matricial modelada. Por fim, apresentaremos resultados que permitem concluir que o jogo na configuração $2 \times 2n$, em que $n \in \mathbb{N}$ sempre tem solução única.

2 Uma breve contextualização histórica

Os sistemas lineares e o determinante apresentam uma importância significativa, uma vez que são extremamente úteis para a resolução de problemas em diversos contextos e aplicações.

Conforme (SOUZA, SABINO, SABINO, 2017) o estudo de sistemas de equações lineares deu origem inicialmente ao estudo de determinantes e posteriormente ao das matrizes. As primeiras evidências desta utilização foram encontradas em tabletas babilônicas feitas de argila por volta de 300 a.C. Além disso, o livro Nove Capítulos sobre a Arte Matemática, publicado na China entre 200 a.C. e 100 a.C., mostra representações dos coeficientes de sistemas lineares em barras de bambu e é considerado uma das primeiras fontes a mencionar a ideia de matrizes. Embora os sistemas de equações lineares, matrizes e determinantes possam parecer assuntos distintos, sua história mostra que estão interligados.

A primeira ideia de determinante foi desenvolvida pelo matemático japonês Seki Kowa em 1683, e em sua obra ele escreveu vários exemplos de sistemas de equações em forma matricial. No mesmo ano, o matemático alemão Gottfried Wilhelm Leibniz também começou a utilizar determinantes para resolver sistemas de equações lineares, estabelecendo a condição de compatibilidade de um sistema de três equações a duas incógnitas em termos do determinante de ordem 3, formado pelos coeficientes e pelos termos independentes. Portanto, ambos resolviam equações lineares, porém de maneiras diferentes.

Além disso, quem também contribuiu para a teoria dos determinantes foi o escocês Colin Maclaurin, que em 1730 escreveu "Um Tratado sobre Álgebra" que mais tarde, em 1748, foi publicado como livro foi publicado e nele encontra-se o "teorema geral" para eliminação de incógnitas de sistemas lineares, onde traz demonstrações para matrizes de ordem 2, 3 e 4. Este teorema é o que conhecemos hoje por regra de Cramer, pois o matemático suíço Gabriel Cramer publicou o livro "Introdução à Análise de Curvas Algébricas", em 1750, que apresentava resultados para matrizes de ordem n (SOUZA, SABINO, SABINO, 2017).

O termo determinante só foi cunhado em 1801 quando utilizado por Gauss. Cauchy foi o primeiro a usar determinantes no sentido moderno em 1812 e, com isso, foi responsável por boa parte da teoria inicial sobre esse novo conceito, que se disseminou a partir de 1841 quando Jacobi o popularizou utilizando-os no contexto de funções de várias variáveis. Por fim, no século XIX a teoria dos determinantes havia se desenvolvido a ponto de livros inteiros serem dedicados a ela, contudo apesar de sua belíssima história os determinantes

hoje têm um interesse mais teórico do que prático, posto que ele perdeu espaço para métodos númericos mais eficazes na resolução de sistemas lineares (POOLE, 2004). Portanto, notamos que a história desses três conceitos se entrelaçam e proporocionam uma rica ferramenta Matemática.

3 SISTEMAS DE EQUAÇÕES LINEARES, MATRIZES E DETERMINANTES

Este capítulo tratará primordialmente resultados relacionados a sistemas lineares e matrizes, em que o objetivo é evidenciar proposições que garantem a existência da solução de sistemas lineares. Ademais, vale salientar que o trabalho supõem que o leitor tenha familiaridade com alguns tópicos de Álgebra Linear, portanto, alguns resultados não vão ser demonstrados.

3.1 Sistemas de Equações Lineares

Definição: Uma equação linear nas variáveis $x_1, x_2, \ldots, x_n$ é uma equação que pode ser escrita na forma $a_1x_1 + a_2x_2 + \cdots + a_nx_n = b$, onde b e os coeficientes $a_1, a_2, \ldots, a_n$ são números reais.

Definição: Um sistema de equações lineares S de m equações e n incógnitas $x_1, x_2, \ldots, x_n$ é uma lista de equações

$$S : \begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2, \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m, \end{cases}$$

onde b_i e a_{ij} são constantes (em geral números reais) para $1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant n$.

Uma solução do sistema é uma lista $(s_1, s_2, \ldots, s_n)$ de números reais que torna as equações simultaneamente verdadeiras quando substituímos x_1 por s_1, x_2 por s_2, ..., x_n por s_n.

Usaremos a sigla SEL para designar sistema de equações lineares.

O sistema S pode ser reescrito usando notação matricial.

Sejam

$$\mathrm{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \quad \mathrm{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathrm{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}.$$

Com isso o sistema S pode ser escrito na forma $\mathrm{Ax} = \mathrm{b}$.

A matriz $\mathrm{A} = [a_{ij}]_{m\times n}$ é chamada matriz dos coeficientes, a matriz coluna $\mathrm{x} = [x_i]_{n\times 1}$ é matriz das incógnitas e a matriz coluna $\mathrm{b} = [b_i]_{m\times 1}$ matriz dos termos independentes.

Teorema: Para um sistema de equações lineares temos três possibilidades:

(i) **Solução única**: sistema possível (compatível) e determinado.

Somente uma lista $(s_1, s_2, \ldots, s_n)$ satisfaz simultaneamente as m equações.

(ii) **Nenhuma solução**: sistema impossível (incompatível).

Nenhuma lista satisfaz as equações simultaneamente.

(iii) **Infinitas soluções**[1]: sistema possível (compatível)e indeterminado.

Infinitas listas da forma $(s_1, s_2, \ldots, s_n)$ satisfazem simultaneamente as m equações.

3.1.1 Métodos Diretos para Resolução de Sistemas Lineares

A palavra *escalonar* vem da palavra latina *scala*, que significa escada ou degraus. Escalonar uma matriz significa dar a ela a forma de escada.

Definição: Uma matriz está na **forma escalonada por linhas** quando satisfaz as seguintes propriedades:

(i) Todas as linhas que consistem inteiramente de zeros estão na parte inferior da matriz.

(ii) Em cada linha não nula, o primeiro elemento não nulo (chamado de *elemento líder*) está em uma coluna à esquerda de qualquer outro elemento líder abaixo dele.

As propriedades acima garantem que os elementos líderes fiquem posicionados formando uma escada.

Definição: As seguintes operações elementares com as linhas podem ser realizadas em uma matriz:

[1] Uma observação importante é que a existência de infinitas soluções não vale para corpos finitos.

(i) Trocar duas linhas de posição. ($L_i \leftrightarrow L_j$ significa trocar as linhas i e j)

(ii) Multiplicar uma linha por uma constante não nula. (kL_i significa multiplicar a linha i pelo número real k)

(iii) Somar um múltiplo de uma linha com outra linha. ($L_i + kL_j$ significa somar k vezes a linha j à linha i e trocar a linha i pelo resultado)

O processo de aplicar operações elementares com linhas para transformar uma matriz em uma matriz escaloada é chamado de escalonamento.
Definição: Duas matrizes A e B são ditas **linha equivalentes** se existir uma sequência de operações elementares com as linhas de A que converta A em B.

O Método de Eliminação de Gauss

Quando um escalonamento é aplicado à matriz completa de um sistema de equações lineares, criamos um sistema equivalente que pode ser resolvido por substituição de trás para a frente. O processo inteiro é conhecido como método da eliminação de Gauss, ou método de eliminação gaussiana.

(i) Escreva a matriz ampliada do sistema de equações lineares.

(ii) Use operações elementares com as linhas para reduzir a matriz ampliada à forma escalonada por linhas.

(iii) Usando substituição de trás para frente, resolva o sistema equivalente que corresponde à matriz linha-reduzida.

Definição: Uma matriz está na **forma escalonada reduzida por linhas** se ela satisfaz as seguintes propriedades:

(i) Todas as linhas que consistem inteiramente de zeros estão na parte inferior da matriz.

(ii) O elemento líder em cada linha não nula é igual a 1 (chamado **1 líder**).

(iii) Cada coluna que contém um 1 líder tem zeros em todas as outras posições.

No exemplo abaixo veremos como funciona a eliminação gaussiana.

Exemplo:
$$\begin{cases} -2y + 3z = 5 \\ 4x + 3y + z = 2 \\ 2x + 4y - z = 1 \end{cases}$$

$$\left[\begin{array}{ccc|c} 0 & -2 & 3 & 5 \\ 4 & 3 & 1 & 2 \\ 2 & 4 & -1 & 1 \end{array}\right] \xrightarrow{L_1 \leftrightarrow L_3} \left[\begin{array}{ccc|c} 2 & 4 & -1 & 1 \\ 4 & 3 & 1 & 2 \\ 0 & -2 & 3 & 5 \end{array}\right] \xrightarrow{L_2-(2L_1)} \left[\begin{array}{ccc|c} 2 & 4 & -1 & 1 \\ 0 & -5 & 3 & 0 \\ 0 & -2 & 3 & 5 \end{array}\right]$$

$$\left[\begin{array}{cccc|c} 2 & 4 & -1 & \vdots & 1 \\ 0 & -5 & 3 & \vdots & 0 \\ 0 & -2 & 3 & \vdots & 5 \end{array}\right] \xrightarrow{L_3-(2/5L_2)} \left[\begin{array}{cccc|c} 2 & 4 & -1 & \vdots & 1 \\ 0 & -5 & 3 & \vdots & 0 \\ 0 & 0 & 9/5 & \vdots & 5 \end{array}\right]$$

Assim o sistema equivalente fica $\begin{cases} 2x + 4y - z = 1 \\ -5y + 3z = 0 \\ \frac{9}{5}z = 5 \end{cases}$.

Fazendo a retrossubstituição obtemos $z = \frac{25}{9}, y = \frac{15}{9}$ e $x = -\frac{13}{9}$. Portanto o sistema tem solução única $\left(-\frac{13}{9}, \frac{15}{9}, \frac{25}{9}\right)$.

Definição: O **posto** de uma matriz é o número de linhas não nulas de qualquer uma de suas formas escalonadas por linhas.

Definição: Um sistema de equações lineares é dito homogêneo se o termo independente em cada equação é igual a zero.

3.2 Conjuntos Geradores

Um vetor $\mathbf{v}$ em $\mathbb{R}^n$ pode ser escrito como uma n-upla $\mathbf{v} = (x_1, x_2, \ldots, x_n)$ ou como uma matriz coluna $\mathbf{v} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$.

Definição: Um vetor $\mathbf{v}$ é uma combinação linear dos vetores $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ se existirem escalares $c_1, c_2, \ldots, c_k$ tais que $\mathbf{v} = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots + c_k\mathbf{v}_k$. Os escalares $c_1, c_2, \ldots, c_k$ são chamados coeficientes da combinação linear.

Teorema: Um sistema de equações lineares **Ax**=**b** é possível se, e somente se, **b** é uma combinação linear das colunas de **A**.

Definição: Seja $S = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k\}$ é um conjunto de vetores de $\mathbb{R}^n$. O conjunto de todas as combinações lineares de $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ é denominado conjunto gerado por $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ e é denotado por ger$(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k)$ ou ger(S) ou span$(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k)$ ou span(S).

Em outras palavras, span(S)= $\{\alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \cdots + \alpha_k\mathbf{v}_k : \alpha_i \in \mathbb{R}\}$.

3.2.1 Dependência Linear

Definição: Um conjunto de vetores $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ é linearmente dependente se existirem escalares $c_1, \ldots, c_k$, pelo menos um dos quais **não** nulo, tais que $c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \ldots + c_k\mathbf{v}_k = \mathbf{0}$.

Um conjunto de vetores não linearmente dependente é chamado **linearmente independente**.

Teorema: Um conjunto de vetores $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ de $\mathbb{R}^n$ é linearmente dependente se, e somente se, pelo menos um dos vetores pode ser escrito como combinação linear dos demais.

Teorema: Sejam $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m$ vetores (coluna) de $\mathbb{R}^n$ e $\mathbf{A}$ a matriz $n \times m$ $[\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m]$ que tem esses vetores como suas colunas. Então $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m$ serão linearmente dependentes se, e somente se, o sistema linear homogêneo cuja matriz completa for $[\mathbf{A} \vdots \mathbf{0}]$ tiver uma solução não trivial, ou seja, tiver infinitas soluções.

Demonstração:
Os vetores $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m$ serão LD se, e somente se, existirem escalares $c_1, c_2, \ldots c_m$ não todos nulos tais que $c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \ldots + c_m\mathbf{v}_m = \mathbf{0}$. Mas isso é equivalente a dizer que o vetor não nulo $\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix}$ é uma solução do sistema homogêneo cuja matriz completa é $\left[\mathbf{v}_1\, \mathbf{v}_2\, \ldots\, \mathbf{v}_m \vdots \mathbf{0}\right]$. $\square$

3.2.2 Base, Dimensão e Subespaços

Definição: Um conjunto $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k\}$ de vetores de $\mathbb{R}^n$ é dito uma base de $\mathbb{R}^n$ se:

(i) $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k\}$ é um conjunto linearmente independente.

(ii) $\text{span}(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k) = \mathbb{R}^n$.

Definição: Um subespaço de $\mathbb{R}^n$ é um conjunto S de vetores de $\mathbb{R}^n$ tal que:

(i) O vetor nulo pertence a S. ($\mathbf{0} \in S$)

(ii) Se $\mathbf{u}$ e $\mathbf{v}$ pertencem a S, então $\mathbf{u} + \mathbf{v}$ também pertence a S. (S é fechado para a soma de vetores)

(iii) Se $\mathbf{u}$ pertence a S e α é um escalar, então $\alpha\mathbf{u}$ também pertence a S.

Definição: Uma base de um subespaço S de $\mathbb{R}^n$ é um conjunto de vetores de S que gera S é que é linearmente independente.

Definição: Se S é um subespaço de $\mathbb{R}^n$, então o número de vetores em uma base de S é chamado dimensão de S e denotado por $\dim(S)$.

Definição: Seja $\mathbf{A}$ uma matriz $m \times n$.

1. O espaço linha de $\mathbf{A}$ é o subespaço $\text{lin}(\mathbf{A})$ de $\mathbb{R}^n$ gerado pelas linhas de $\mathbf{A}$.
2. O espaço coluna de $\mathbf{A}$ é o subespaço $\text{col}(\mathbf{A})$ de $\mathbb{R}^m$ gerado pelas colunas de $\mathbf{A}$.
3. O espaço anulado de $\mathbf{A}$ é o subespaço $\text{anul}(\mathbf{A})$ de $\mathbb{R}^n$ formado pelas soluções do sistema linear homogêneo $\mathbf{Ax} = \mathbf{0}$. A dimensão do espaço anulado de $\mathbf{A}$ é denominada a nulidade de $\mathbf{A}$.
4. O espaço anulado à esquerda de $\mathbf{A}$ é o subespaço $\text{anul}(\mathbf{A}^T)$ de $\mathbb{R}^m$ formado pelas soluções do sistema linear homogêneo $\mathbf{A}^T\mathbf{y} = \mathbf{0}$.

Teorema: Sejam $\mathbf{A}$ e $\mathbf{B}$ matrizes equivalentes por linhas. Então $\text{lin}(\mathbf{A}) = \text{lin}(\mathbf{B})$.

Teorema: Sejam $\mathbf{A}$ uma matriz $m \times n$ e N o conjunto solução do sistema linear homogêneo $\mathbf{Ax}= \mathbf{0}$. Então $N = \text{anul}(\mathbf{A})$ é um subespaço de $\mathbb{R}^n$.
Demonstração: Observe que um vetor solução $\mathbf{x}$ precisa ser um vetor de $\mathbb{R}^n$ para que $\mathbf{A}$x esteja definido e $\mathbf{0} = \mathbf{0}_m$ é um vetor de $\mathbb{R}^m$. Assim $N = \{\mathbf{x} \in \mathbb{R}^n \, : \, \mathbf{A}x = \mathbf{0} = \mathbf{0}_m\}$

(espaço anulado por $\mathbf{A}$, isto é, $N = \text{anul}(\mathbf{A})$.

(i) $\mathbf{0}_n \in N$, pois $\mathbf{A0} = \mathbf{0}$.

(ii) Sejam $\mathbf{u}$, $\mathbf{v} \in N$. Logo $\mathbf{A}u = \mathbf{0}$ e $\mathbf{A}v = \mathbf{0}$. Então $\mathbf{A}(\mathbf{u}+\mathbf{v}) = \mathbf{A}u + \mathbf{A}v = \mathbf{0} + \mathbf{0} = \mathbf{0}$, logo $\mathbf{u} + \mathbf{v} \in N$.

(iii) Sejam $\mathbf{u} \in N$ e $a \in \mathbb{R}$. Logo $\mathbf{A}(a\mathbf{u}) = a\mathbf{A}u = a \cdot \mathbf{0} = \mathbf{0}$ e então $a\mathbf{u} \in N$. □

Teorema: Seja $\mathbf{A}$ uma matriz cujos elementos são números reais. Para qualquer sistema de equações lineares $\mathbf{Ax} = \mathbf{b}$, vale exatamente uma das condições abaixo:

(a) Não tem solução. (b) Tem uma única solução. (c) Tem infinitas soluções[2].
Demonstração: Se o sistema $\mathbf{A}x = \mathbf{b}$ não tem solução ou se tem uma solução não temos nada a provar. Agora supõe que o sistema tem pelo menos duas soluções distintas $\mathbf{x}_1$ e $\mathbf{x}_2$, ou seja, $\mathbf{A}x_1 = \mathbf{b}$ e $\mathbf{A}x_2 = \mathbf{b}$.

[2] Uma observação importante é que a existência de infinitas soluções não vale para corpos finitos.

Seja $\mathbf{x}_0 = \mathbf{x}_1 - \mathbf{x}_2$, então $\mathbf{A}x_0 = A(\mathbf{x}_1 - \mathbf{x}_2) = \mathbf{A}x_1 - \mathbf{A}x_2 = \mathbf{b} - \mathbf{b} = \mathbf{0}$ e assim $\mathbf{x}_0 \in$ anul($\mathbf{A}$). Como anul($\mathbf{A}$) é um subespaço de $\mathbb{R}^n$ segue que $t\,\mathbf{x}_0 \in$ anul($\mathbf{A}$), para todo $t \in \mathbb{R}$.

Agora vamos considerar os infinitos vetores $\mathbf{x}_1 + t\,\mathbf{x}_0$. Logo $\mathbf{A}(\mathbf{x}_1 + t\,\mathbf{x}_0) = \mathbf{A}x_1 + \mathbf{A}x_0 = \mathbf{b} + \mathbf{0} = \mathbf{b}$. □

Teorema: Seja $\mathbf{A}$ uma matriz de ordem $m \times n$. Então os espaços linha e coluna de $\mathbf{A}$ têm a mesma dimensão.
Demonstração: Seja $\mathbf{B}$ a matriz na forma escalonada por linhas de $\mathbf{A}$. Vimos que lin ($\mathbf{A}$) = lin ($\mathbf{B}$) e assim dim (lin ($\mathbf{A}$)) = dim (lin ($\mathbf{B}$)) = número de linhas não nulas de $\mathbf{B}$ = número de $\mathbf{1}$'s líderes de $\mathbf{B}$. Vamos chamar esse número de r.

Em geral, col ($\mathbf{A}$) $\neq$ col ($\mathbf{B}$), mas as colunas de $\mathbf{A}$ e de $\mathbf{B}$ têm as mesmas relações de dependência.

Logo dim (col ($\mathbf{A}$)) = dim (col ($\mathbf{B}$)) e como existem r $\mathbf{1}$'s líderes, $\mathbf{B}$ tem r colunas que são os vetores unitários canônicos $\mathbf{e}_1$, $\mathbf{e}_2$, ..., $\mathbf{e}_r$ de $\mathbb{R}^m$. Esses vetores são uma base de col($\mathbf{B}$) e assim dim(col($\mathbf{B}$)) = r . □

Definição: O posto de uma matriz $\mathbf{A}$ é a dimensão de seus espaços linha e coluna e é denotado por posto($\mathbf{A}$).

Teorema do Posto: Se $\mathbf{A}$ é uma matriz $m \times n$, então posto ($\mathbf{A}$) + nulidade ($\mathbf{A}$) = n.
Demonstração: Seja $\mathbf{B}$ a matriz escalonada reduzida por linhas de $\mathbf{A}$ e suponha que posto($\mathbf{A}$) = r. Então $\mathbf{B}$ tem r $\mathbf{1}$'s líderes e assim há r variáveis dependentes e $n - r$ variáveis livres na solução $\mathbf{A}$x = $\mathbf{0}$. Como dim(anul($\mathbf{A}$)) = $n - r$, temos que posto($\mathbf{A}$) + nulidade($\mathbf{A}$) = $r + (n - r) = n$. □

3.2.3 Matrizes Invertíveis

Definição: Uma matriz A de ordem $n \times n$ é invertível se, e somente se, existe uma matriz B, também de ordem n, de modo de que: $AB = BA = I_n$. A matriz B chama-se inversa de A e denota-se por A^{-1}.

Proposição: Se uma matriz A quadrada de ordem n é invertível, então sua inversa é única.
Demonstração: Suponhamos que existem matrizes B e C tais que $AB = BA = I_n$ e $AC = CA = I_n$.

$$B = BI_n = B(AC) = (BA)C = I_nC = C.\square$$

Teorema das Matrizes Invertíveis[3]: Seja A uma matriz $n \times n$. As seguintes afirmações são equivalentes:

a) A é invertível.

b) Para cada **b**, $Ax = b$ tem uma única solução.

c) $Ax = 0$ tem apenas a solução trivial.

d) Posto (A)= n.

e) Nulidade (A)=0.

[3] O teorema das Matrizes Invertíveis que pode ser encontrada em qualquer livro de Álgebra Linear, normalmente, apresenta uma série de afirmações equivalentes, no presente texto serão exibidos apenas aquelas úteis para esse contexto.

4 CONGRUÊNCIAS

Neste capítulo será apresentada uma sucinta revisão sobre congruências, classes residuais, bem como conceitos relacionados a corpos e grupos. Fora utlizado como referência principal o livro *Álgebra Moderna* (DOMINGUES, Hygino H.; IEZZI, Gelson).

4.1 Congruência

Definição: Dois números inteiros a e b são congruentes módulo m se os restos de sua divisão euclidiana por m são iguais. Quando os inteiros a e b são congruentes módulo m, escrevemos $a \equiv b \pmod{m}$.

Exemplo: Note que $25 \equiv 13 \pmod{2}$, pois o resto da divisão de 25 e 13 por 2 é igual a 1.

4.2 Classes Residuais

Definição: Denominamos classe residual módulo m do elemento $a \in \mathbb{Z}$, denotada por $[a]$, como sendo o subconjunto

$$[a] = \{x \in \mathbb{Z} : x \equiv a \pmod{m}\}.$$

Sendo assim, o conjunto de todas as classes residuais módulo m será denotado por $\mathbb{Z}_m$, e representado da seguinte forma:

$$[\mathbb{Z}_m] = \{[0], [1], [2], \ldots, [m-1]\}$$

Exemplo: Para $m = 2$ que é de fato o que vamos utilizar nesta pesquisa, temos:

$$[0] = \{x \in \mathbb{Z} : x \equiv 0 \pmod{2}\}$$

$$[1] = \{x \in \mathbb{Z} : x \equiv 1 \pmod{2}\}$$

Portanto, $[a] = [0]$ se, e somente se, a é par; e $[a] = [1]$ se, e somente se, a é ímpar.

4.3 Operações com Classes Residuais

Vamos definir a adição e a multiplicação de classes residuais, bem como apresentar suas propriedades.

Definimos a adição e multiplicação de classes módulo m por:

$$[a] + [b] = [a + b]$$

$$[a] \cdot [b] = [a \cdot b]$$

Propriedades: Para todos $[a], [b]$ e $[c] \in Z_m$ temos as propriedades da adição:

1) $([a] + [b]) + [c] = [a] + ([b] + [c])$ (associativa).
   Demonstração: $[a] + ([b] + [c]) = [a] + [b + c] = [a + (b + c)] = [(a + b) + c] = [a + b] + [c] = ([a] + [b]) + [c]$

2) $[a] + [b] = [b] + [a]$ (comutatividade).
   Demonstração: $[a] + [b] = [a + b] = [b + a] = [b] + [a]$

3) $[a] + [0] = [a]$, para todo $[a] \in \mathbb{Z}_m$ (existência do elemento neutro).

4) $[a] + [-a] = [0]$ (existência de simétrico).

Propriedades: Para todos $[a], [b], [c] \in Z_m$ temos as propriedades da multiplicação:

1) $([a] \cdot [b]) \cdot [c] = [a] \cdot ([b] \cdot [c])$ (associativa).

2) $[a] \cdot [b] = [b] \cdot [a]$ (comutatividade).

3) $[a] \cdot [1] = [a]$ (existência de unidade).

4) $[a] \cdot ([b] + [c]) = [a] \cdot [b]) + [a] \cdot [c] =$ (distributiva).

4.4 Corpo

Nesta seção vamos apresentar de uma maneira sucinta sobre a noção de corpo.

Definição: Um corpo é um conjunto K no qual estão definidas duas operações, adição e multiplicação, satisfazendo os seguintes axiomas:

1) $a + b = b + a$, para todo $a, b \in K$ (comutativa da adição).

2) $a + (b + c) = (a + b) + c$, para todo $a, b, c \in K$ (associoativa da adição).

3) Existe $0 \in K$, tal que $a + 0 = a$, para todo $a \in K$ (elemento neutro da adição).

4) Dado $a \in K$, existe $(-a) \in K$ tal que $a + (-a) = 0$ (existência do oposto).

5) $(ab)c = a(bc)$, para todo $a, b, c \in K$ (associativa da multiplicação).

6) $ab = ba$, para todo $a, b, c \in K$ (comutativa da multiplicação).

7) Existe $1 \in K$ tal que $a \cdot 1 = a$, para todo $a \in K$ (elemento neutro da multiplicação).

8) Para todo $a \in K, a \neq 0$ existe um único $b \in K$ tal que $a \cdot b = 1$. Denotamos b por a^{-1}. (elemento neutro da multiplicação).

9) $a(b + c) = ab + ac$, para todo $a, b, c \in K$ (distributiva da multiplicação em relação à adição).

Exemplo: O conjunto dos números reais, com as operações usuais de adição e multiplicação é um corpo.

Exemplo: O conjunto dos números inteiros $\mathbb{Z}$, com as operações usuais de adição e multiplicação não é um corpo, pois $2 \in \mathbb{Z}$, porém não existe $b \in \mathbb{Z}$ tal que $2 \cdot b = 1$.

Teorema: Se p é primo, então $\mathbb{Z}_p$ é corpo.
Em particular, para $p = 2$.

Observação: Isso implica que $\mathbb{Z}_p$ se comporta em muitos aspectos como os números reais, além disso podemos usar muitos resultados que foram enunciados para os $\mathbb{R}$. Por exemplo, escalonamento para resolver sistemas lineares.

Resultado importante: Se A é uma matriz quadrada com entradas em um corpo K, então A é invertível se, e somente se, det (A) é invertível em K. Em $\mathbb{Z}_m$, com $m > 1$, uma matriz A é invertível se, e somente se, mdc (det (A), m)=1. Em nossa pesquisa como o foco é trabalhar em $\mathbb{Z}_2$, para $m = 2$, temos que se det $(A) = 1 \pmod 2$, então A é invertível em $\mathbb{Z}_2$, já que $mdc(1, 2) = 1$.

5 Python

5.0.1 O que é Python?

Conforme o artigo Python e Orientação a Objetivos publicado na plataforma Alura, tem-se que: "Python é uma linguagem de programação interpretada, orientada a objetos, de alto nível e com semântica dinâmica [...] Python suporta módulos e pacotes, que encoraja a programação modularizada e reuso de códigos".

Dentre a sua gama de possibilidades, Python também pode ser utilizado para resolver cálculos, dentre eles: sistemas lineares.

5.0.2 Sistema de Equações Lineares em Python

A maneira mais prática de resolver sistemas desse tipo em Python é utilizando a função np.linalg.solve. Essa função resolve sistemas lineares na forma:

$$Ax = b$$

O método numérico utilizado na fnção np.linalg.solve é uma decomposição LU com pivotamento particial e permutação das linhas (KAGGLE, 2020).

6 Lights Out

Boa parte desse capítulo utiliza ideias da tese de mestrado [2], que faz uma análise de alguns casos específicos, são eles: $2\times2, 2\times3, 3\times3, 4\times4$ e 5×5. Já no artigo [11] são feitas duas análises do jogo, o trabalho é estruturado em duas colunas em que a da esquerda é tratado como um problema lógico e a da direita como um problema de Álgebra Linear, e esta que será mais importante para os nossos objetivos. Além disso, vale destacar outras duas publicações feitas por Goldwasser et al. em que [4] trata o jogo como um problema da ciência da computação e apresenta alguns resultados mais espefícicos sobre o caso 3×3, já a publicação [5] exibe uma séries de resultados para os jogos no formado $k \times n$. E por fim, usando a tese [13] que também abordou o jogo na perspectiva $k \times n$. A presente pesquisa tem como foco evidenciar resultados teóricos sobre a existência da solução dos jogos $2 \times n$, onde $n \in \mathbb{N}$ e usufruir do Python, a fim de mostrar a solução.

Lights Out é um jogo que foi desenvolvido em 1995 pela Tiger Eletronics. Ele é composto por 25 botões, dos quais há alguns acesos e outros apagados e o objetivo do jogo é acionar a menor quantidade de botões, a fim de que todas as luzes sejam desligadas. Posteriormente, também foram lançadas outras versões, tais como: o Mini *Lights Out* que possui o formato 4×4 e o *Lights Out Deluxe* no formato 6×6.

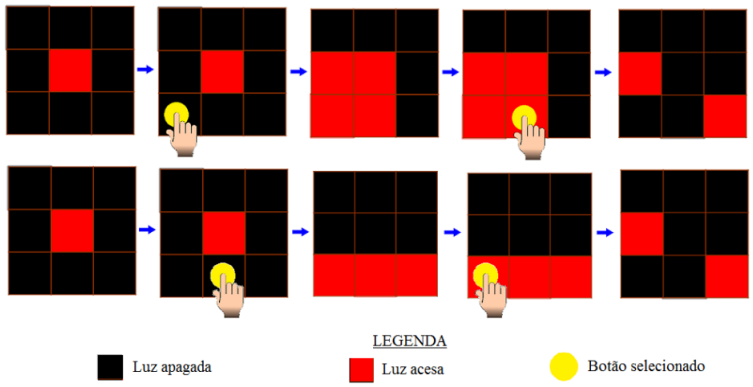
Fonte: DELGADO (2007 apud Agostin, 2020)

Figura 2 – Jogo Lights Out.

O jogo possui 25 botões que podem ser representados pelos elementos de uma matriz 5×5 em que cada um representa o estado de uma luz, isto é, acesa ou apagada. Sendo assim, quando um botão é acionado, seu estado é alterado, assim como todos os botões vizinhos verticais e horizontais. Dessa forma, dada uma configuração inicial o objetivo do jogo é apagar todas as luzes.

Abaixo segue um exemplo em que na configuração inicial está acesa apenas a luz central e exibe uma série de movimentos mostrando o funcionamento do jogo.

Definição: Um jogo será dito solucionável se dada qualquer configuração inicial existe uma sequência de botões que podemos pressionar para desligar todas as luzes.



Fonte: Agostin (2020).

Figura 3 – Funcionamento do Jogo na versão 3x3

Veremos mais adiante que:

a) Pressionar um botão duas vezes é equivalente a não pressioná-lo.

b) A ordem em que os botões são acionados não faz diferença.

6.1 Ideia da modelagem do Jogo Lights Out na configuração 2×2

Consideremos, inicialmente, que o jogo contenha duas linhas e duas colunas.

1	2
3	4

Vamos verificar o que acontece quando pressionamos cada botão, sendo que inicialmente todas as luzes estão desligadas. Esta investigação será necessária para definir a matriz associada ao jogo.

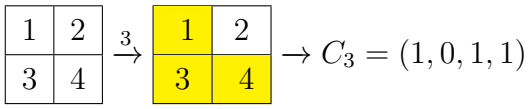
Denotaremos por C_1 o vetor que representa estado que se encontra o jogo a partir do momento que pressionamos o botão 1, supondo todas as luzes desligadas, coforme figura abaixo. As três primeiras componentes de C_1 são iguais a 1 e isso indica que as luzes 1, 2 e 3 estarão acesas. Já a quarta componente é igual a 0 que indica que a luz 4 está desligada.

$$\begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 & 4 \\ \hline \end{array} \xrightarrow{1} \begin{array}{|c|c|}\hline \cellcolor{yellow}1 & \cellcolor{yellow}2 \\ \hline \cellcolor{yellow}3 & 4 \\ \hline \end{array} \rightarrow C_1 = (1,1,1,0)$$

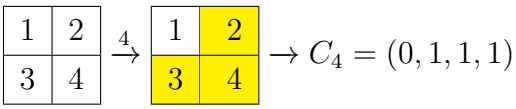
Analogamente, definimos C_2 o vetor que representa estado que se encontra o jogo a partir do momento que pressionamos o botão 2, supondo todas as luzes desligadas inicialmente.

$$\begin{array}{|c|c|}\hline 1 & 2 \\ \hline 3 & 4 \\ \hline \end{array} \xrightarrow{2} \begin{array}{|c|c|}\hline \cellcolor{yellow}1 & \cellcolor{yellow}2 \\ \hline 3 & \cellcolor{yellow}4 \\ \hline \end{array} \rightarrow C_2 = (1,1,0,1)$$

Analogamente, os vetores C_3 e C_4 representam estado que se encontra o jogo a partir do momento que pressionamos os botões 3 e 4, respectivamente, supondo todas as luzes desligadas inicialmente.



$$\rightarrow C_3 = (1, 0, 1, 1)$$



$$\rightarrow C_4 = (0, 1, 1, 1)$$

Para transformar o jogo em um problema de Álgebra Linear vamos definir a matriz A, onde o vetor C_i é a i-ésima coluna de A:

$$A = \begin{bmatrix} C_1 & C_2 & C_3 & C_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

6.2 Modelagem do Jogo Lights Out

As ideias da modelagem desta seção estão baseadas na tese de [2].

Como há apenas dois estados que cada tecla do jogo se encontra, ligada ou desligada, podemos representá-las por elementos de $\mathbb{Z}_2$ em que 1 representa quando a luz estiver na posição ligada e 0 caso estiver desligada.

Posto isso, vamos estabelecer uma estratégia para resolver o jogo em qualquer formato (não somente na forma de um quadrado), utilizando tópicos de Álgebra Linear. Observamos os botões como entradas de uma matriz $M = [m_{ij}]_{k\times n}$. Chamamos de $S = (m_{11}, m_{12}, ..., m_{1n}, m_{21}, m_{22}, ..., m_{2n}, ..., m_{kn})$ o vetor **coluna que representa a configuração inicial** (de luzes acesas ou apagadas). Seja C_j, o vetor coluna que aponta quais luzes são acesas quando o botão j é acionado, considerando que todas as luzes estão inicialmente apagadas, com $j = 1, 2, ..., q$ e onde q denota o número de botões que o jogo possui e $x_i = 0, 1$ com $i = 1, ..., q$ que indica se o botão i foi acionado ou não.

A configuração inicial do jogo S é um vetor contendo apenas os números 0's e 1's. Analogamente, C_j com $j = 1, 2, ..., q$ são vetores cujas coordenadas também são 0 ou 1, pois expressam a configuração de todos os botões quando é selecionado o botão j. Assim, x_jC_j com $j = 1, 2, ..., q$ pode indicar que o botão j foi pressionado, quando $x_j = 1$ e, portanto, temos a configuração de quais botões foram acesos e apagados nesse processo, ou apenas o vetor nulo se $x_j = 0$, que indica que o botão j não foi pressionado.

Dessa forma, precisamos determinar os valores de x_j com $j = 1, 2, ..., q$, **para que quando adicionarmos a configuração inicial** com cada x_jC_j, o resultado seja o vetor nulo. Em outras palavras, devemos resolver a seguinte equação:

$$S + x_1C_1 + x_2C_2 + \cdots + x_qC_q = \mathbf{0},$$

onde $\mathbf{0}$ é o vetor nulo, ou seja, representa todas as luzes desligadas.

Note que, $S \equiv -S \pmod 2$, portanto, podemos reescrever a equação acima como

$$x_1C_1 + x_2C_2 + \cdots + x_qC_q = S.$$

Definimos A a matriz formada pelos vetores coluna $C_1, ..., C_q$, ou seja,

$$A = \begin{bmatrix} C_1 & C_2 & C_3 & C_4 & \dots & C_q \end{bmatrix}.$$

Se X é a matriz coluna formada por $x_1, x_2, ..., x_q$, podemos representar a equação acima no formato matricial: $AX = S$. [1]

[1] Note que a partir da equação acima e a sua representação matricial, também é possível a partir de uma configuração inicial que está toda desligada fazer movimentos até chegar na configuração B desejada. Porém, o objetivo do jogo é fazer o processo inverso. Apenas para fins de curiosidade a matriz A é chamada nos outros artigos de *Matrix Push*.

Note que, se o jogo é de tamanho $k \times n$, a matriz A associada é quadrada de ordem kn, pois como há k linhas e n colunas, então há kn botões e para cada um haverá um vetor coluna. Como estamos interessados em jogos $2 \times n$ e $n \times 2$, as matrizes serão de ordem $2n$.

Agora provaremos o seguinte resultado importante:

Proposição:

(a) Pressionar um botão duas vezes é equivalente a não pressioná-lo.

(b) A ordem em que os botões são acionados não faz diferença.

Demonstração:

(a) Considerando a equação

$$x_1C_1 + x_2C_2 + \cdots + x_qC_q = S$$

lembre que x_j é 0 ou 1, o que implica que clicar duas vezes no mesmo botão k seria

$$x_1C_1+x_2C_2+\cdots+1\cdot C_k+1\cdot C_k+\cdots+x_qC_q = x_1C_1+x_2C_2+\cdots+2\cdot C_k+\cdots+x_qC_q$$

Mas, $2 \equiv 0 \pmod 2$ e assim temos que a expressão fica

$$x_1C_1 + x_2C_2 + \cdots + 0C_k + \cdots + x_qC_q.$$

Portanto provamos que clicar duas vezes em um botão é a mesma coisa que não clicar.

(b) Note que nosso objetivo é demonstrar que $1 * C_i + 1 * C_j = 1 * C_j + 1 * C_i$ o que é óbvio, uma vez que a soma de vetores e ou matrizes, vale a comutatividade, portanto, $1 * C_i + 1 * C_j = C_i + C_j = C_j + C_i = 1 * C_j + 1 * C_i$. $\square$

6.2.1 Jogo 2×2

Consideremos, inicialmente, que o jogo contenha duas linhas e duas colunas.

1	2
3	4

Já construímos acima a matriz A.

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

No exemplo abaixo resolvemos o jogo com determinada configuração inicial.

Exemplo: Considere a seguinte configuração inicial

1	2
3	4

Note que o vetor S que representa a configuração inicial é:

$$S = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Nosso objetivo é resolvermos a equação $AX = S$ e um dos métodos que podem ser utilizados é o escalonamento, escrevendo como matriz completa.

$$\Big[A \mid S\Big] = \left[\begin{array}{cccc|c} 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{array}\right]$$

Obtemos a seguinte matriz equivalente

$$\left[\begin{array}{cccc|c} 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{array}\right]$$

Ou seja, $x_1 = 0, x_2 = 1, x_3 = 0, x_4 = 1$. Além disso, é fácil notar olhando para a matriz equivalente que o $\det A = 1 \pmod 2$, ou seja, existe solução e é única.

1	2
3	4

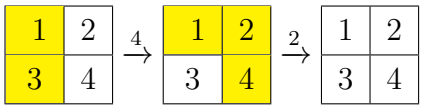
$\xrightarrow{2}$

1	2
3	4

$\xrightarrow{4}$

1	2
3	4

E a ordem que os botões são pressionados não interfere na solução



Apesar do método do escalonamento funcionar, à medida que trabalhamos com jogos em configurações maiores ele não se torna prático, uma vez que para cada caso se torna necessário refazer o escalonamento, logo, ele não permite uma generalização.

6.3 Dedução da solução da configuração 4×2

Antes de começarmos a desenvolver os cálculos faz-se necessário evidenciar que esta pesquisa propõe um estudo do jogo na configuração $2 \times n$, porém as matrizes quando trabalhamos em uma configuração $k \times 2$ permitem uma melhor manipulação algébrica. Além disso, será demonstrado um resultado que nos fornece que a solubilidade da configuração $2 \times n$ e $n \times 2$ são as mesmas, ou seja, se existe solução na primeira, então existe na segunda e vice e versa.

O jogo possui quatro linhas e duas colunas.

1	2
3	4
5	6
7	8

A ideia é estabelecer um resultado teórico que permita concluir se essa configuração é solucionável ou não, isto é, se dada qualquer configuração inicial existe solução. Para isso, é necessário conhecermos quem é a matriz A associada que é de ordem 8.

$$A = \begin{bmatrix} C_1 & C_2 & C_3 & C_4 & C_5 & C_6 & C_7 & C_8 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Note que podemos escrever a matriz A em blocos:

$$A = \begin{bmatrix} R & I & O & O \\ I & R & I & O \\ O & I & R & I \\ O & O & I & R \end{bmatrix}, \text{ onde } R = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ e } O = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

Observe que I é a matriz identidade e O é a matriz nula todas de ordem 2.

Lembrando que o nosso objetivo é afirmar a existência de soluções da equação $AX = S$, para o jogo 4×2 em que S é o vetor associado a configuração inicial, portanto o vetor S possui 8 coordenadas.

Apesar de ser possível fazer o escalonamento não é viável devido ao tamanho da matriz então, baseado no trabalho [11], ao invés de trabalharmos com um sistema 8×8 podemos

utilizar a matriz acima A (escrevendo ela como 4×4), em seguida podemos escrever os vetores X, S como

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \quad S = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix},$$

dividindo em subvetores x_i e s_i, cada um com duas componentes. Consequentemente, podemos reescrever como um sistema menor da seguinte maneira:

$$\begin{bmatrix} R & I & O & O \\ I & R & I & O \\ O & I & R & I \\ O & O & I & R \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix}$$

Além disso, podemos transformar $AX = S$ em uma equação equivalente $0 = AX + S$, pois $A \equiv -A \pmod 2$ (estamos em $\mathbb{Z}_2$).

Temos que $JX = JX + AX + S$ vale para qualquer matriz J de ordem 4, cujos elementes são matrizes 2×2.

Conforme feito em [11], escolhemos a matriz J indicada abaixo.

$$J = \begin{bmatrix} O & I & O & O \\ O & O & I & O \\ O & O & O & I \\ O & O & O & O \end{bmatrix}$$

Observe que $A + J = \begin{bmatrix} R & O & O & O \\ I & R & O & O \\ O & I & R & O \\ O & O & I & R \end{bmatrix}$

Portanto, a equação acima pode ser reescrita na forma $JX = (A + J)X + S$ e assim obtemos o seguinte sistema

$$\begin{bmatrix} x_2 \\ x_3 \\ x_4 \\ 0 \end{bmatrix} = \begin{bmatrix} R & O & O & O \\ I & R & O & O \\ O & I & R & O \\ O & O & I & R \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix}$$

$$\begin{bmatrix} x_2 \\ x_3 \\ x_4 \\ 0 \end{bmatrix} = \begin{bmatrix} Rx_1 + s_1 \\ x_1 + Rx_2 + s_2 \\ x_2 + Rx_3 + s_3 \\ x_3 + Rx_4 + s_4 \end{bmatrix}$$

É fácil ver que podemos escrever x_2 em termos de x_1 e s_1, assim como podemos escrever x_3, x_4 em termos de x_1 e S. Portanto, podemos reescrever o sistema assim como:

$$\begin{bmatrix} x_2 \\ x_3 \\ x_4 \\ 0 \end{bmatrix} = \begin{bmatrix} B_1 & B_0 & O & O & O \\ B_2 & B_1 & B_0 & O & O \\ B_3 & B_2 & B_1 & B_0 & O \\ B_4 & B_3 & B_2 & B_1 & B_0 \end{bmatrix} \begin{bmatrix} x_1 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix}$$

Em que $B_0 = I, B_1 = R$ e $B_i = B_{i-2} + R \cdot B_{i-1}$, se $i \geqslant 2$.

Observação: O objetivo do presente trabalho é tratar a existência de solução de maneira teórica, porém as soluções serão obtidas utilizando a linguagem Python.

Note que criamos uma equação matricial que usa apenas x_1 e o vetor S que representa a configuração inicial, ou seja, os s_i da matriz acima são conhecidos e assim é suficinete encontrar x_1 e a partir dele descobrir as demais incógnitas. Observando a última linha temos que

$$0 = B_4x_1 + B_3s_1 + B_2s_2 + B_1s_3 + B_0s_4.$$

Como estamos trabalhando em módulo 2 ($-B_4x_1 \equiv B_4x_1 \pmod 2$) segue que

$$B_4x_1 = B_3s_1 + B_2s_2 + B_1s_3 + B_0s_4.$$

Agora temos um sistema para encontrar x_1, pois os s_i e os B_i são conhecidos. Como o nosso objetivo é tratar apenas a existência de solução, [11] e [13] chamam o lado direito da equação de *gathers(s)*, uma vez que não terá tanta importância.

Assim temos que resolver $B_4x_1 = gathers(s)$.

Lembrando que $B_0 = I, B_1 = R$ e $B_i = B_{i-2} + R \cdot B_{i-1}, i \geqslant 2$, temos

$B_3 = B_1 + R \cdot B_2 = R + R(B_0 + R \cdot B_1) = R + R + R \cdot R \cdot B_1 = 2R + R^3 = R^3$

$B_4 = B_2 + R \cdot B_3 = (B_0 + R \cdot B_1) + R^4 = I + R^2 + R^4$

Como $R = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ segue que $R^2 = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$, pois estamos resolvendo módulo 2.

Então $R^4 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ e assim $B_4 = I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

Logo a equação $B_4 x_1 = gathers(s)$ é equivalente a $Ix_1 = gathers(s)$ e então $x_1 = gathers(s)$.

Portanto, existe x_1 e a partir disso conseguimos encontrar x_2, x_3, x_4, mostrando que o jogo 4×2 é **solucionável**.

6.4 Jogo 2×2 em outra perspectiva

Já havíamos criado a matriz A associada a essa configuração anteriormente.

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

Faremos uma análise semelhante a anterior. Note que podemos reduzir ainda mais a matriz A escrevendo-a como uma matriz por blocos

$$A = \begin{bmatrix} R & I \\ I & R \end{bmatrix} \quad R = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \quad J = \begin{bmatrix} O & I \\ O & O \end{bmatrix}$$

Ao invés de usarmos a equação $AX = S$ faremos $JX = (A + J)X + S$, em que

$$\begin{bmatrix} x_2 \\ 0 \end{bmatrix} = \begin{bmatrix} R & O \\ I & R \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}$$

Analogamente, podemos reescrever como

$$\begin{bmatrix} x_2 \\ 0 \end{bmatrix} = \begin{bmatrix} B_1 & B_0 & 0 \\ B_2 & B_1 & B_0 \end{bmatrix} \begin{bmatrix} x_1 \\ s_1 \\ s_2 \end{bmatrix}$$

Através da última linha obtemos $0 = B_2x_1 + B_1s_1 + B_0s_2$. Pelo mesmo motivo anterior podemos escrever a última linha como, $B_2x_1 = gathers(s)$.

$B_2 = B_0 + R \cdot B_1 = I + R^2$

Assim como calculado acima, obtemos de R^2:

$$R^2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Portanto, $B_2 = I$ o que implica x_1 existe, logo, o jogo neste formato é **solucionável**. Ou seja, dada qualquer configuração inicial, sempre há solução.

6.5 Teoremas e Resultados sobre a existência de solução dos jogos $k \times 2$ e $2 \times n$

O presente trabalho concentra no estudo de configurações $2 \times n$, já os trabalhos de [11], [4], [5] e [13] fazem uma análise $k \times 2$, ou seja, eles fixam o número de colunas igual a 2 e variam o número de linhas. Porém em um dos resultados apresentados em seguida veremos que a existência de solução não muda.

Para cada jogo no formato $k \times 2$ as matrizes R, I, O são:

$$R = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, O = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

Lema 1: Sejam $B_0 = I, B_1 = R$ e $B_{k+1} = R \cdot B_{k-1} + B_{k-2}$, para $k \geqslant 1$.
Para todo k inteiro não negativo temos que $B_{2k} = I, R \cdot B_{2k+1} = O, B_{4k+1} = R$ e $B_{4k+3} = O$.

Demonstração: Começamos calculando alguns valores de B_k:

$B_2 = R \cdot B_1 + B_0 = R \cdot R + I = R^2 + I = O + I = I.$
$B_3 = R \cdot B_2 + B_1 = R \cdot I + R = R + R = O.$
$B_4 = R \cdot B_3 + B_2 = R \cdot O + I = O + I = I.$
$B_5 = R \cdot B_4 + B_3 = R \cdot I + O = R + O = R.$

Agora vamos provar o resultado por indução.

Base de indução: Se $k = 0$, então
$B_{2 \cdot 0} = I, R \cdot B_{2 \cdot 0+1} = R \cdot B_1 = R \cdot R = O, B_{4 \cdot 0+1} = B_1 = R$ e $B_{4 \cdot 0+3} = B_3 = O$.

Hipótese de Indução: Supõe que
$B_{2k} = I, R \cdot B_{2k+1} = O, B_{4k+1} = R$ e $B_{4k+3} = O$ vale até um certo $k \geqslant 0$.

Passagem de Indução: Vamos provar para $k + 1$:
$B_{2k+2} = R \cdot B_{2k+1} + B_{2k} =^{HI} O + I = I.$
Assim fica provado que $B_n = I$, se n for par.
$R \cdot B_{2k+3} = R \cdot (R \cdot B_{2k+2} + B_{2k+1}) = R \cdot R \cdot I + R \cdot B_{2k+1} =^{HI} O + O = O.$
$B_{4k+5} = R \cdot B_{4k+4} + B_{4k+3} = R \cdot I + O = R + O = R.$
$B_{4k+7} = R \cdot B_{4k+6} + B_{4k+5} = R \cdot I + R = R + R = O.$
Portanto segue o resultado. □

Teorema 1: Para todo jogo com a configuração $2k \times 2$ possui solução.
Demonstração: Para qualquer jogo de configuração $2k \times 2$, a solubilidade será determinanda pela matriz B_{2k}. Pelo lema 1, $B_{2k} = I$ e então todo estado inicial admite solução. □

Corolário 1: As matrizes associadas aos jogos $(2k+1) \times 2$ não são invertíveis, logo nem sempre há solução para uma determinanda configuração inicial.
Demonstração: Se n é um inteiro positivo ímpar, então $n = 4k+1$ ou $n = 4k+3$. No lema 1 povamos que $B_{4k+1} = R$ e $B_{4k+3} = O$. Então, temos que as matrizes B_n se n é um inteiro positivo ímpar não são invertíveis, logo nem sempre possuem solução. □

O seguinte resultado é de suma importância, uma vez que os artigos pesquisados tratam o jogo no formato $k \times 2$, enquanto que o foco dessa pesquisa é a configuração $2 \times n$.

Proposição: O caso $k \times n$ tem a mesma solubilidade do caso $n \times k$.

Demonstração: Lembre-se de que geramos a matriz A, a matriz de pressionamento de botão, considerando o efeito de apertar cada botão. Ou seja, a rotulagem dos botões, no entanto, foi essencialmente arbitrária.

O tabuleiro $k \times n$ é essencialmente o tabuleiro $n \times k$ girado, então a matriz A correspondente será gerada renomeando os botões, porém as relações entre eles continuam as mesmas. Isso significa que podemos passar de A_n (matriz do jogo $k \times n$) para a matriz A_k (matriz do jogo $n \times k$) através de uma série de trocas de linhas e colunas. Assim, ambas as matrizes possuem a mesma dimensão do espaço nulo e, portanto, representarão sistemas com a mesma solubilidade. □

No exemplo a seguir transformaremos o jogo 3×2 em um jogo 2×3 confirmando o resultado acima e, posteriormente faremos um exemplo em que o objetivo é exibir o corolário 1, isto é, mostrar que quando tratamos um jogo que o número de colunas (linhas) é ímpar, implica que pode ou não haver solução.

Exemplo: O jogo 3×2.

1	2
3	4
5	6

Tem como matriz de pressionamento de botão.

$$A_2 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Girando a placa, obtemos a seguinte rotulagem:

$1 \to 1$	$3 \to 2$	$5 \to 3$
$2 \to 4$	$4 \to 5$	$6 \to 6$

Portanto, isso explica a trocas de linhas e colunas e assim temos como matriz

$$A_2 = \begin{bmatrix} C_1 & C_2 & C_3 & C_4 & C_5 & C_6 \end{bmatrix} \to \begin{bmatrix} C_1 & C_3 & C_5 & C_2 & C_4 & C_6 \end{bmatrix}$$

Ficando com a seguinte matriz

$$A_2 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \to \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Porém, agora note que as matrizes associadas aos jogos sempre são simétricas, portanto, também faremos trocas nas linhas, ou seja, a nova matriz terá como linhas

$$\begin{bmatrix} C_1 \\ C_3 \\ C_5 \\ C_2 \\ C_4 \\ C_6 \end{bmatrix}$$

Isto é,

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \to \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} = A_3$$

E esta última matriz representa o jogo 2×3. E esse resultado permite darmos continuidade ao foco de pesquisa que trata do Ligths-Out nessa nova configuração.

Exemplo: Vamos ilustrar dois exemplos no jogo 2×3 um que não há solução e outro que existe, porém ocorre um fato interessante.

Caso 1: Considere inicialmente a seguinte configuração de luzes

1	2	3
4	5	6

Portanto, nosso objetivo é resolver o sistema $AX = S$, isto é,

$$\Big[A \mid S\Big] = \left[\begin{array}{cccccc|c} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{array}\right]$$

Utilizando operações elementeares na matriz amplicada, obtemos um sistema equivalente:

$$\Big[A \mid S\Big] = \left[\begin{array}{cccccc|c} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

Note que da penúltima linha, temos que $0 = 1$, o que é um absurdo. Portanto, é impossível apagar todas as luzes da configuração dada.

Caso 2: Suponhamos agora que o jogo tenha como configuração inicial:

1	2	3
4	5	6

$$\Big[A \mid S\Big] = \left[\begin{array}{cccccc|c} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{array}\right]$$

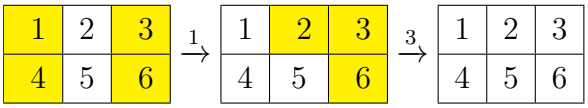
Fazendo o processo de escalonamento, obtemos

$$\Big[A \mid S\Big] = \left[\begin{array}{cccccc|c} 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

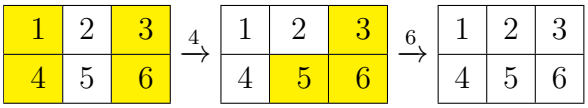
Portanto, a solução do sistema é $S = (x_5 + x_6 + 1, x_5, x_5 + x_6 + 1, x_6, x_5, x_6)$.

Se $x_5 = 0$ há duas possibilidades para x_6.

Se $x_5 = 0$ e $x_6 = 0$, temos como solução $S = (1, 0, 1, 0, 0, 0)$

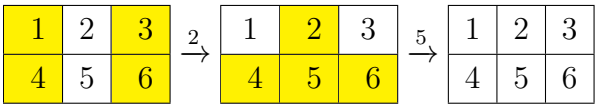


Se $x_5 = 0$ e $x_6 = 1$,temos como solução $S = (0, 0, 0, 1, 0, 1)$

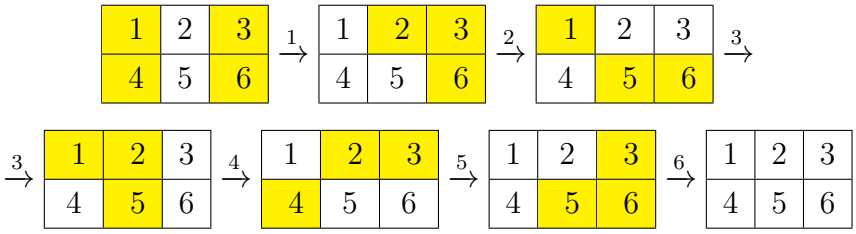


Se $x_5 = 1$ também há duas possibilidades para x_6.

Se $x_5 = 1$ e $x_6 = 0$,temos como solução $S = (0, 1, 0, 0, 1, 0)$



Se $x_5 = 1$ e $x_6 = 1$,temos como solução $S = (1, 1, 1, 1, 1, 1)$



Em que todas permitem solucionar o jogo.

7 Análises e Soluções

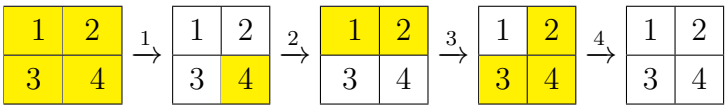
7.1 Jogos e Soluções na configuração 2×2

Neste formato de jogo há $2^4 = 16$ possibilidades de configurações iniciais, serão apresentadas mais algumas e suas respectivas soluções, através do programa em Python.

Exemplo 1: Agora que já compreendemos como funciona as matrizes A e S vamos utilizar do programa em Python para resolver o jogo, que através dos resultados anteriores implica que existe solução.

1	2
3	4

A partir do programa em Python obtemos como solução $x = (1, 1, 1, 1)$



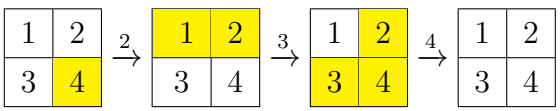
Para que o leitor perceba a flexibilidade que a Álgebra Linear nos proporciona trabalhar com esse tema, vamos apresentar diferentes maneiras de abordar a solução, isto é, usando vetores colunas e a própria modificação nos jogos. Vale ressaltar que os números sobre as flechas representam o botão acionado.

O exemplo a seguir é análogo ao anterior, porém o intuito é apresentar uma outra maneira de exibir o jogo.

Exemplo 2: Seja $S = [1, 1, 1, 1]$, como foi dito acima através do programa em Python encontramos como solução $x = [1, 1, 1, 1]$

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \xrightarrow{1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \xrightarrow{2} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{3} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \xrightarrow{4} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Exemplo 3: $S = [0, 0, 0, 1]$, através da programação em Python encontramos como solução $x = [0, 1, 1, 1]$

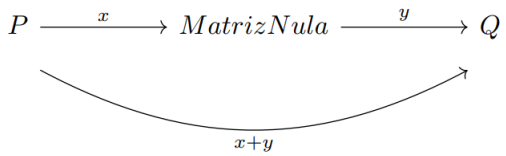


$$\begin{bmatrix}0\\0\\0\\1\end{bmatrix} \xrightarrow{2} \begin{bmatrix}1\\1\\0\\0\end{bmatrix} \xrightarrow{3} \begin{bmatrix}0\\1\\1\\1\end{bmatrix} \xrightarrow{4} \begin{bmatrix}0\\0\\0\\0\end{bmatrix}$$

Observação Importante:

O presente trabalho concentra-se em abordar o objetivo do jogo que consiste em dada uma configuração inicial deseja-se apagar todas as luzes.

Porém, devida a modelagem do jogo, isto é, a equação $Ax = B$, resultados citados anteriormente e uma pequena observação anterior, implicam que dadas uma configuração inicial P e uma configuração final Q (não necessariamente toda nula) desejadas, é possível efetuar uma séries de passos, para que chegue no que se procura.



Fonte: autor.

Figura 4 – Passagem de configurações.

Em que o vetor x, y representam os botões para resolver as configurações P e Q, respectivamente.

Exemplificando, caso se deseje passar da configuração $P = [1, 1, 1, 1]$ para a configuração $Q = [0, 0, 0, 1]$, note que apresentamos acima as sequências de botões, a fim de que ambas chegassem no vetor nulo, ou seja, $x = [1, 1, 1, 1]$ e $y = [0, 1, 1, 1]$, implica que $x + y = [1, 0, 0, 0]$. O que é fácil de verificar.

7.2 Jogos e Solução na configuração 2×4

Já nesse formato há $2^8 = 256$ possibilidades de configurações, pois há 8 botões que possuem dois estados, ligado ou desligado.

Inicialmente vamos exibir o jogo usando a tabela retangular, posto que ela permite uma fácil compreensão visual do processo.

Exemplo 1: Consirando o jogo com a seguinte configuração inicial:

1	2	3	4
5	6	7	8

A partir do programa em Python obtemos como solução $x = (0, 0, 0, 1, 0, 1, 1, 1)$

1	2	3	4	$\xrightarrow{4}$	1	2	3	4	$\xrightarrow{6}$	1	2	3	4	$\xrightarrow{7}$	1	2	3	4	$\xrightarrow{8}$	1	2	3	4
5	6	7	8		5	6	7	8		5	6	7	8		5	6	7	8		5	6	7	8

Neste próximo exemplo além de exibirmos a imagem da solução em Python, utilizaremos da representação como vetores coluna e os números acima de cada flecha referem-se ao botão que foi pressionado.

Exemplo 2: Consideramos a configuração inicial com todas as luzes acesas. Usando o Python para resolver:

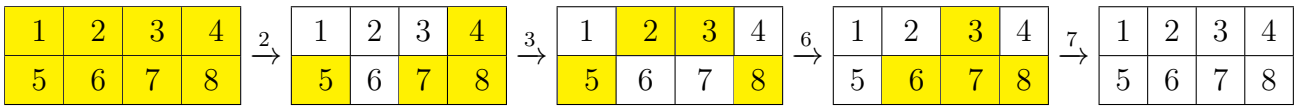
1	2	3	4
5	6	7	8

```
[0, 1, 1, 1, 0, 0, 1, 0]
[0, 0, 1, 1, 0, 0, 0, 1]
[1, 0, 0, 0, 1, 1, 0, 0]
[0, 1, 0, 0, 1, 1, 1, 0]
[0, 0, 1, 0, 0, 1, 1, 1]
[0, 0, 0, 1, 0, 0, 1, 1]
Determinante: 4.999999999999999
Determinante correto: 5.0
Digite o valor de b[1]: 1
Digite o valor de b[2]: 1
Digite o valor de b[3]: 1
Digite o valor de b[4]: 1
Digite o valor de b[5]: 1
Digite o valor de b[6]: 1
Digite o valor de b[7]: 1
Digite o valor de b[8]: 1
Vetor de termos independentes:
[1. 1. 1. 1. 1. 1. 1. 1.]
Solução:  [0.4 0.2 0.2 0.4 0.4 0.2 0.2 0.4]
O menor valor em módulo não nulo é: 0.1999999999999999
novo vertor: [2.000000000000001, 1.0, 1.0000000000000009, 2.00000000000001, 2.0000000000000013, 1.0000000000000007, 1.0000000000000007, 2.000000000000001]
Novo vetor com erros de arredondamento corrigidos: [2.0000000001, 1.0000000001, 1.0000000001, 2.0000000001, 2.0000000001, 1.0000000001, 1.0000000001, 2.0000000001]
Novo vetor x em módulo 2: [0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0]
PS C:\Users\cesar\OneDrive\Área de Trabalho\Python>
```

Fonte: autor.

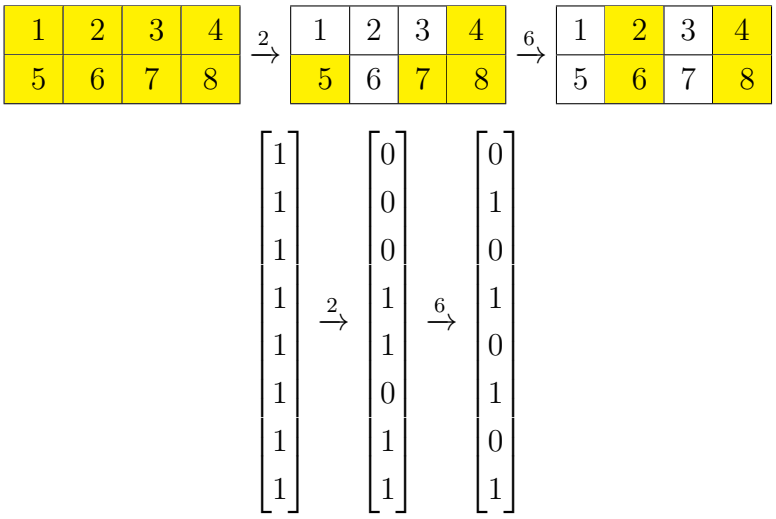
Figura 5 – Solução em Python.

Portanto, o vetor solução é $x = [0, 1, 1, 0, 0, 1, 1, 0]$.



$$\begin{bmatrix}1\\1\\1\\1\\1\\1\\1\\1\end{bmatrix} \xrightarrow{2} \begin{bmatrix}0\\0\\0\\1\\1\\0\\1\\1\end{bmatrix} \xrightarrow{3} \begin{bmatrix}0\\1\\1\\0\\1\\0\\0\\1\end{bmatrix} \xrightarrow{6} \begin{bmatrix}0\\0\\1\\0\\0\\1\\1\\1\end{bmatrix} \xrightarrow{7} \begin{bmatrix}0\\0\\0\\0\\0\\0\\0\\0\end{bmatrix}$$

Exemplo 3: Suponhamos que o objetivo é partir de todas as luzes ligadas e chegar na configuração $Q = [0, 1, 0, 1, 0, 1, 0, 1]$. Primeiramente, nosso objetivo será saber $Ay = Q$. Encontramos usando Python $y = [0, 0, 1, 0, 0, 0, 1, 0]$. Como já temos $x = [0, 1, 1, 0, 0, 1, 1, 0]$ Agora, se a gente quiser sair da configuração com todas as luzes ligadas e chegar na configuração Q, faremos $x + y = [0, 1, 0, 0, 0, 1, 0, 0]$.



7.3 Regularidade não demonstrada

Usando o código em Python para resolver o sistema obtemos também o determinante de cada matriz associada ao jogo. Segue a tabela dos determinantes de cada matriz:

Jogo $2 \times n$	**Tamanho da matriz** A	**det** A
2×2	4×4	-3
2×3	6×6	0
2×4	8×8	5
2×5	10×10	0
2×6	12×12	-7
2×7	14×14	0
2×8	16×16	9
2×9	18×18	0
2×10	20×20	-11
2×11	22×22	0

8 CONCLUSÃO

Foi deveras interessante partir de um simples jogo, Lights Out, modelar matematicamente e buscar resolvê-lo, uma vez que eu tenho um certo apreço por jogos e pela investigação. É notório que o campo da Álgebra Linear apresenta conceitos que com um grau de abstração considerável, contudo é um ramo extremamente importante e com uma gama muito grande de aplicações. Sendo assim, este trabalho visa proporcionar uma divertida aplicação e que permite visualizar esses resultados teóricos.

Neste trabalho foi possível concluir que os jogos no formato retangular $2 \times 2n$ sempre tem solução, uma vez que as matrizes associadas a este jogo sempre são invertíveis, em contrapartida os casos $2 \times (2n+1)$ nem sempre possuem solução ou possuem mais de uma solução, visto que as matrizes associadas não são invertíveis.

Como sugestões de leituras sobre este assunto, no artigo de [13] é tratado também do caso mais geral $k \times n$, além disso faz uma relação com a sequência de Fibonacci. Mais sobre estes resultados, podem ser encontrado no trabalho de [4] e [5] que exibe outros teoremas e corolários bem mais específicos.

Referências

[1] ANTON, Howard; RORRES, Chris. **Álgebra linear: com aplicações**. 10.ed. Porto Alegre: Bookman, 2012.

[2] D' AGOSTIN, Izabele. **Aplicações de Álgebra Linear**. 102f. Dissertação-Programa de Mestrado Profissional em Matemática em Rede Nacional-PROFMAT, Universidade Tecnológicaa Federal do Paraná, Curitiba, 2020.

[3] DOMINGUES, Hygino H.; IEZZI, Gelson. **Álgebra Moderna**. 4. ed. São Paulo: Atual, 2003.

[4] GOLDWASSER, J.; KLOSTERMEYERS, W.; TRAPP, G.; ZHANG, C. **Setting Swichtes in a Grid**. West Virginia University. 13p. 1996.

[5] GOLDWASSER, J; KLOSTERMEYERS, W.; TRAPP, G. **Characterizing Switch Settings Problems**. Linear and Multilinear Algebra, vol. 43. 1-3, p.121-134, 1997.

[6] JOÃO, Antônio. **Modelagem do Jogo Lights Out usando Sistemas Lineares**. 86 p. Dissertação-Programa de Mestrado Profissional em Matemática, Universidade Federal de Santa Catarina, Florianópolism 2016.

[7] LAY, David C; LAY, Steven R.; MACDONALD, Judi J. **Álgebra linear e suas aplicações**. 5.ed. Rio de Janeiro: LTC, 2018.

[8] LIMA, Elon Lages. **Álgebra Linear**. 8. ed. Rio de Janeiro: IMPA, 2012.

[9] PELLEGRINI, Jerônimo C. **álgebra linear com aplicações**,2022. Disponível em: *<http://aleph0.info/cursos/al/notas/al.pdf>*. Acesso em: 28 de maio de 2023.

[10] POOLE, David. **Álgebra linear: uma introdução moderna**. 2.ed. São Paulo: Cengage Learning, 2016.

[11] SÁNCHEZ, Óscar Marín; FLORES, Cristóbal Pareja, **Two Reflecet Analyses of Ligths Out**. *Mathemactis Magazine*, 2001.

[12] SOUSA, Fábio; SABINO, Elizabeth; SABINO, Elizete. **Abordagem História e Conceitual sobre os Sistemas Lineares e sua Relação com Matrizes e Determinantes**. 15p. Jornada em Estudos em Matemática, Marabá, 2017.

[13] WILSON, Tamar Elise. **Lights Out: Determining solvability on rectangular boards**. 47f. South Hadley, 2009.

9 APÊNDICE

Apêndice A - Código do desenvolvimento do Sistema de Equações

Para transformar os jogos $2 \times n$ em uma matriz e a partir disso elaborar uma equação matricial foi utilizado o código abaixo, uma observação que vale salientar que ele pede para o usuário inserir o tamanho da matriz, ou seja, o jogo 2×2 a matriz é de ordem 4, o jogo 2×4 a matriz é de ordem 8...

```
#Cria a matriz associada ao jogo de maneira automática, basta indicar o tamanho da matriz
#O usuário insere a matriz B que representa a configuração do jogo;
#Fornece a solução;


import numpy as np
import random

# Definir o tamanho da matriz
n = int(input ("Digite o tamanho da matriz: "))


# Criar uma matriz nxn nula

matriz = [[0 for j in range(n)] for i in range(n)]

#for i in range (0,n):
  #for j in range (0,n):
    #matriz [i][j]= int (input (f'Digite um valor para [{i},{j}]: '))

for i in range (0,n):
    if (i==0):
        for j in range (0,n):
            if (j==i):
                matriz [i][j]= 1
            elif (j==i+1):
                matriz [i][j]= 1
            elif (j==n/2):
                matriz [i][j]= 1
    elif (i==n-1):
        for j in range (0,n):
            if (j==i):
```

```
                matriz [i][j]= 1
            elif (j==i-1):
                matriz [i][j]= 1
            elif (j==(n/2)-1):
                matriz [i][j]= 1
    elif (i==(n/2) - 1):
        for j in range (0,n):
            if (j==i):
                matriz [i][j]= 1
            elif (j==i-1):
                matriz [i][j]= 1
            elif (j==n-1):
                matriz [i][j]= 1
    elif (i==(n/2)):
        for j in range (0,n):
            if (j==i):
                matriz [i][j]= 1
            elif (j== 0):
                matriz [i][j]= 1
            elif (j==i+1):
                matriz [i][j]= 1
    else:
        if (i <= (n/2) -1):
            for j in range (0,n):
                if (j==i):
                    matriz [i][j]= 1
                elif (j==i-1):
                    matriz [i][j]= 1
                elif (j==i+1):
                    matriz [i][j]= 1
                elif (j==i+(n/2)):
                    matriz [i][j]= 1
        else:
            for j in range (0,n):
                if (j==i):
                    matriz [i][j]= 1
                elif (j==i-1):
                    matriz [i][j]= 1
                elif (j==i+1):
                    matriz [i][j]= 1
                elif (j==i-(n/2)):
```

```
                    matriz [i][j]= 1


# Calcular o determinante da matriz usando a biblioteca NumPy
determinante = np.linalg.det(np.array(matriz))

# Imprimir a matriz e o determinante
print("Matriz:")
for linha in matriz:
    print(linha)
#print("Determinante:", determinante) #colocar um if para informar se há solução única ou

determinante_rounded = round(determinante, 2)  # arredonda para 2 casas decimais
print("Determinante correto:", determinante_rounded)



# Cria um vetor de termos independentes com valores inseridos pelo usuário
b = np.zeros(n)
for i in range(n):
    b[i] = float(input(f"Digite o valor de b[{i+1}]: "))


#Cria um vetor  de termos indepentes que possui apenas o elemento 1
c= np.zeros(n)
for j in range(n):
    c[j] = 1



# Imprime a matriz e o vetor
#print("Matriz: ")
#print(matriz)
print("Vetor de termos independentes: ")
print(b)
#print("A matriz auxiliar é:", c)

# Resolve o sistema linear
x = np.linalg.solve(matriz, b)
y = np.linalg.solve(matriz, c)
# Imprime a solução
```

```
print("Solução: ", x)
#print("Solução auxiliar: ", y)

# Encontra o menor elemento não nulo em módulo do vetor y
valores_nao_nulos = [valor for valor in y if valor != 0]
if valores_nao_nulos:
    valores_absolutos = [abs(valor) for valor in valores_nao_nulos]
    menor_valor_absoluto = min(valores_absolutos)
    print("O menor valor em módulo não nulo é:", menor_valor_absoluto)
else:
    print("Não há valores não nulos no vetor x")


# Divide todos os elementos do vetor x pelo menor valor absoluto encontrado
x = [valor/menor_valor_absoluto for valor in x]

print ("novo vertor:", x)

for i in range(len(x)):
    if abs(x[i] - round(x[i])) < 1e-6:  # verifica se a diferença é próxima a 0.999999
        x[i] -= 1e-10  # subtrai uma quantidade muito pequena
        x[i] = round(x[i])  # arredonda o valor resultante
        x[i] += 1e-10  # adiciona novamente a quantidade muito pequena
#print ("Novo vetor com erros de arredondamento corrigidos:", x)

# Divide todos os elementos do vetor x pelo menor valor absoluto encontrado
x_mod_2 = [(valor // menor_valor_absoluto) % 2 for valor in x]
print("SOLUÇÃO FINAL (vetor x em mod 2):", x_mod_2)
```

ApêndiceB - Código que gera o jogo automaticamente

O código abaixo permite que o usuário troque o número de colunas, porém a configuração inicial é gerada aleatóriamente.

```
#Jogo que gerado aleatoriamente, porem e possivel modificar o tamanho de 2xn
#Na linha 62 permite modificar o número de colunas

import tkinter as tk
import random

class LightOut:

    def __init__(self, master, cols=10):
        self.master = master
        self.rows = 2
        self.cols = cols
        self.buttons = []
        self.create_board()

    def create_board(self):
        for i in range(self.rows):
            row = []
            for j in range(self.cols):
                button = tk.Button(self.master, bg='white', width=2, height=1,
                                   command=lambda i=i, j=j: self.switch(i, j))
                button.grid(row=i, column=j)
                row.append(button)
            self.buttons.append(row)
        self.randomize_board()

    def randomize_board(self):
        for i in range(self.rows):
            for j in range(self.cols):
                if random.random() < 0.5:
                    self.switch(i, j)

    def switch(self, i, j):
        self.toggle_button(i, j)
        if i > 0:
            self.toggle_button(i-1, j)
        if i < self.rows-1:
            self.toggle_button(i+1, j)
```

```
        if j > 0:
            self.toggle_button(i, j-1)
        if j < self.cols-1:
            self.toggle_button(i, j+1)
        if self.check_win():
            tk.messagebox.showinfo('Light Out', 'Você Ganhou!')
            self.master.destroy()

    def toggle_button(self, i, j):
        if self.buttons[i][j]['bg'] == 'white':
            self.buttons[i][j]['bg'] = 'black'
        else:
            self.buttons[i][j]['bg'] = 'white'

    def check_win(self):
        for i in range(self.rows):
            for j in range(self.cols):
                if self.buttons[i][j]['bg'] == 'black':
                    return False
        return True

root = tk.Tk()
root.title('Light Out')

game = LightOut(root, cols=6) #Modificar a quantidade de colunas
root.mainloop()
```

ApêndiceC -O número de colunas e a configuração inicial são fornecidas pelo usuário

O código abaixo permite que o usuário insira o número de colunas e a configuração inicial, há no próprio código comentários que buscam explicar como ele funciona.

```
#Jogo em que a configuração inicial é fornecida pelo usuário, assim como o número de colu
#Primeiramente, na linha 63 faça a alteração no número de colunas;
#Em seguida na linha 62 "initial_board = [[1, 0], [1, 0]]" a primeira chaves represente a

import tkinter as tk

class LightOut:
    def __init__(self, master, rows=2, cols=5, initial_board=None):
        self.master = master
        self.rows = rows
        self.cols = cols
        self.buttons = []
        self.create_board(initial_board)

    def create_board(self, initial_board):
        for i in range(self.rows):
            row = []
            for j in range(self.cols):
                if initial_board and initial_board[i][j] == 1:
                    button = tk.Button(self.master, bg='black', width=2, height=1,
                                       command=lambda i=i, j=j: self.switch(i, j))
                else:
                    button = tk.Button(self.master, bg='white', width=2, height=1,
                                       command=lambda i=i, j=j: self.switch(i, j))
                button.grid(row=i, column=j)
                row.append(button)
            self.buttons.append(row)

    def switch(self, i, j):
        self.toggle_button(i, j)
        if i > 0:
            self.toggle_button(i-1, j)
        if i < self.rows-1:
            self.toggle_button(i+1, j)
        if j > 0:
            self.toggle_button(i, j-1)
        if j < self.cols-1:
            self.toggle_button(i, j+1)
```

```
        if self.check_win():
            tk.messagebox.showinfo('Light Out', 'Você Ganhou!')
            self.master.destroy()

    def toggle_button(self, i, j):
        if self.buttons[i][j]['bg'] == 'white':
            self.buttons[i][j]['bg'] = 'black'
        else:
            self.buttons[i][j]['bg'] = 'white'

    def check_win(self):
        for i in range(self.rows):
            for j in range(self.cols):
                if self.buttons[i][j]['bg'] == 'black':
                    return False
        return True

root = tk.Tk()
root.title('Light Out')

# Define a configuração inicial dos botões
initial_board = [[1, 1, 0, 1, 1, 0, 1, 1], [0, 1, 0, 0, 1, 0, 0, 1]]
game = LightOut(root, rows=2, cols=8, initial_board=initial_board)


root.mainloop()
```