Eastern Washington University
EWU Digital Commons

EWU Masters Thesis Collection

Student Research and Creative Works

2013

The game of Lights out

Rebecca S. Meyer
Eastern Washington University

Follow this and additional works at: http://dc.ewu.edu/theses

Part of the Physical Sciences and Mathematics Commons

Recommended Citation

Meyer, Rebecca S., "The game of Lights out" (2013). *EWU Masters Thesis Collection*. 167.
http://dc.ewu.edu/theses/167

This Thesis is brought to you for free and open access by the Student Research and Creative Works at EWU Digital Commons. It has been accepted for inclusion in EWU Masters Thesis Collection by an authorized administrator of EWU Digital Commons. For more information, please contact jotto@ewu.edu.

THE GAME OF LIGHTS OUT

A Thesis

Presented To

Eastern Washington University

Cheney, Washington

In Partial Fulfillment of the Requirements

for the Degree

Master of Science

By

Rebecca Meyer

Spring 2013

THESIS OF REBECCA MEYER APPROVED BY

____ DATE:________

RONALD GENTLE, GRADUATE STUDY COMMITTEE

____ DATE:________

W. DALE GARRAWAY, GRADUATE STUDY COMMITTEE

____ DATE:________

REBEKAH REMPE, GRADUATE STUDY COMMITTEE

Abstract

The game of Lights Out is a game with simple rules but an intriguing mathematical structure. The goal of this paper is to first introduce a basic overview of the classic game including game rules and methods for finding solutions, and then extend these concepts using many different mathematical tools. An analysis of special game moves called quiet patterns and a solution method called light chasing will be addressed. A deeper examination of the abstract structure of the game board and the implications of this structure will be started as well as suggested possibilities for further research.

Acknowledgements

First and foremost, I would like to thank Dr. Gentle for all of his insight and guidance throughout this process. I have truly enjoyed having a great adviser that has made this thesis so enjoyable to work on. I would like to thank my friends and family, specifically Robert Meyer, Korina Meyer, Kristina Meyer and Matthew Luttrell for all of their support and encouragement throughout this process. I would also like to thank Robert DeLorto for the numerous thesis work parties and the help motivating me to keep working. Lastly, I would like to thank The Service Station for the countless mocha's and chai tea's that made this thesis possible.

Contents

Abstract iii

Acknowledgements iv

1 Introduction **1**

1.1 History of the Game 1

1.2 Game Play Rules and Variations 2

2 The Game of Lights Out **5**

2.1 Game Board and Light Connections 5

2.2 Light States and Configurations 11

2.3 Classic Lights Out Game Play 13

2.4 The μ Mapping 21

2.5 The Solve Matrix 23

2.6 Invertibility 26

2.7 Pseudo-Inverse 28

2.8 Using the Pseudo-Inverse 35

3 Quiet Patterns **38**

3.1 Defining Quiet Patterns 38

3.2 Solvability Regarding Quiet Patterns 39

3.3 Finding the Spanning Set of the Null Space 42

3.4 Row Reducing A 47

3.5 Method for Generating Quiet Patterns 56

3.6 Symmetry and Rotations 59

3.7 Even Number of Ones 61

4 Light Chasing 65

4.1 General Algorithm 65

4.2 Solvability Regarding Light Chasing 67

4.3 Individual Game Board Rules 72

5 Structure Based on Game Board Dimensions 74

5.1 The Sylvester Equation Representation 74

5.2 Use of the Characteristic Polynomial and Eigenvectors 79

5.3 General Representation of Quiet Patterns 87

5.4 Quiet Pattern Null Space Correspondence 96

5.5 Light Chasing General Theory 100

5.6 Reduced Row Echelon Form of A 106

5.7 The Chase Matrix 112

5.8 Several G_n Chasing Rules 122

5.9 Further Study of $\{b\}$ 125

5.10 Conclusion and Further Research Ideas 132

Bibliography 134

Appendix 136

Matlab Programs 136

Chapter 1

Introduction

The game of Lights Out is a simple game with straightforward rules that is a stimulating puzzle for anyone up to the challenge of a logic game. The most intriguing aspects of this game are hidden within the mathematical structure describing the game that allows for the use of techniques from various fields of mathematics. Before diving in to the mathematics, this chapter provides some background on the game itself in its many forms, including the rules used in game play.

1.1 History of the Game

In 1978, Parker Brothers released one of the earliest models of an electronic hand-held game, called Merlin. It was comprised of a small grid of LED lights, a speaker, control buttons, and a memory of several games and puzzles. Vulcan Electronics produced a similar game in 1983 called XL-25, however the most popular game of its kind, Lights Out, was created as a hand-held toy by Tiger Electronics in 1995. Throughout the late 1990's several versions of this game were released, including Lights Out grids of various sizes, Orbix, Lights Out

2000, and Alien Tiles. Today, this game, and several of its variations can be found online or even downloaded to smart phones as apps.

Lights Out and its variations have not only been played by millions of people but have also been studied in detail by many. Several published contributors to the study of the mathematics behind the game include Jaap Scherphuis, Klaus Sutner, Don Pelletier, J. Goldwasser, W. Klostermeyer, G, Trapp and S. Kauffman. Their research extends the simplicity of the game to graph theory, linear algebra, polynomial sequences and many other topics. The Lights Out game was given the title σ Game by mathematicians, referencing the σ action of button presses to solve a game.

1.2 Game Play Rules and Variations

The traditional game of Lights Out is played on a grid of 5×5 lights. The game starts out with an initial condition, where some combination of lights are on and the rest are off. The goal of the player is to press the correct sequence of buttons in order to turn all of the lights out. Each light on the grid is a button, and pressing a light changes the state of the light pressed, as well as the lights that are adjacent in a + shape around it. Regardless of the version of the game you play, there are a few underlying components to the Lights Out game.

Example 1.1 If each of the boxes on this grid represent a light, consider what happens when light x is pressed. Under the rules of the classic game of Lights Out, the light and those connected in a + pattern around it will change state. If button x is pressed, then x as well as the lights labeled with $*$'s will change state. Here are a few possibilities.

	$*$			
$*$	x	$*$		
	$*$			

			$*$	x
				$*$

		$*$		
	$*$	x	$*$	

□

The first important component of the Lights Out Game is the graph it is played on. In the traditional versions of the game, it is played on a square grid of some size, commonly a 3×3 grid for Merlin or 5×5 for Lights Out. Newer versions have extended the playing board to grids of all sizes, including rectangular grids, and three dimensional figures such as a $3 \times 3 \times 3$ cube for the Lights Out Cube and a Dodecahedron for Orbix.

The next component is the possible states of each light. Most commonly, lights can be in one of two states, either on or off. Games such as Lights Out 2000 have colored lights, where each time a button is pressed, the light and its neighbors cycle between red, green or off. The version called Alien Tiles increases the difficulty even more by adding four different colors the lights can be, red, green, blue and yellow.

There are many possibilities for how a button press will affect neighboring lights. In the traditional Lights Out game, pressing a light changes the state of the light pressed, as well as any other lights in a + pattern around it. Some versions include a wrap-around effect, where the grid acts as if it were a torus. Sometimes the neighbors of the light will include all lights in a square pattern around the light pressed, all the lights in a × shape or even all the lights that could be reached by a knight chess piece from the button pressed. Alien Tiles changes all the lights in the same row and column as the button pressed. In any version, a button press could change its own state as well as its neighbors, or only change its neighbors. In increasingly difficult versions, the player is restricted to only pressing certain lights. There are versions where the player can only press lit buttons, or toggle between lit and unlit. There are endless possibilities if the grid is extended to a directed graph, where button presses between any given pair of lights are not necessarily mutually affected, rather only in a certain direction.

This list of variations is not limited to what is given here, but it does give a taste of the vast possibilities. The rest of this paper will focus mostly on the original Lights Out game for sake of simplicity, however many of the ideas introduced here can be extended easily to these other variations.

After spending some time playing the game of Lights Out in its many forms, several questions naturally begin to arise. These questions aim to push the game to the limit as an attempt is made by the player to search for simple solutions, to find methods that work under any circumstance, and to explore where and why impossible configurations occur. In order to develop concrete answers to these questions and many more, a mathematical structure must be put into place. This next chapter will build several important definitions to set up this structure for further exploration.

Chapter 2

The Game of Lights Out

Using the classic game of Lights Out as reference, the following definitions help describe what game play looks like from a mathematical point of view. This foundation can be applied to many abstract ideas that produce concrete results about the game of Lights Out.

2.1 Game Board and Light Connections

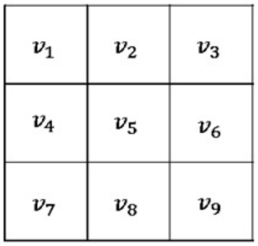
Instead of seeing the game board as a grid, it can be viewed it as a finite connected graph where each vertex represents a light.

Definition 2.1 A *game board*, $G = (V, E)$ is a finite connected graph with set of n vertices $V = \{v_1, v_2, ..., v_n\}$ and set of undirected edges E. An *edge* $(v_i, v_j) \in E$ for some $1 \leq i, j \leq n$ implies a connection between vertex v_i and vertex v_j on game board G.

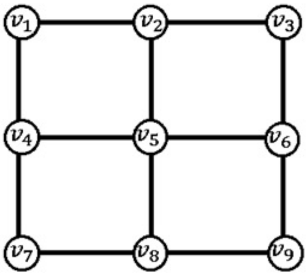
The classic Lights Out game uses a game board with dimensions 5×5, labeled G_5, which contains 25 vertices. The same game can be played on any $n \times n$ game board, labeled G_n. The game board G_3 will be used to explore introductory examples, however most beginning computations for the G_5 and any size G_n can be done identically. The way in which the lights are connected on any game board G is determined by the set of edges E. Each edge defined gives important information as to how a button press will affect its adjacent lights.

In the classic Lights Out game, each light is connected to the lights that are adjacent in a + pattern around them, creating a grid. For example, consider the vertex v_1 with v_2 directly to the right of it on graph G. Then edge (v_1, v_2) symbolizes that pressing either v_1 or v_2 will change the state of v_1 as well as v_2 since the edges are undirected in the classic Lights Out game. There is also an edge from each vertex to itself since pressing a button changes the state of the vertex pressed in the classic game of Lights Out.

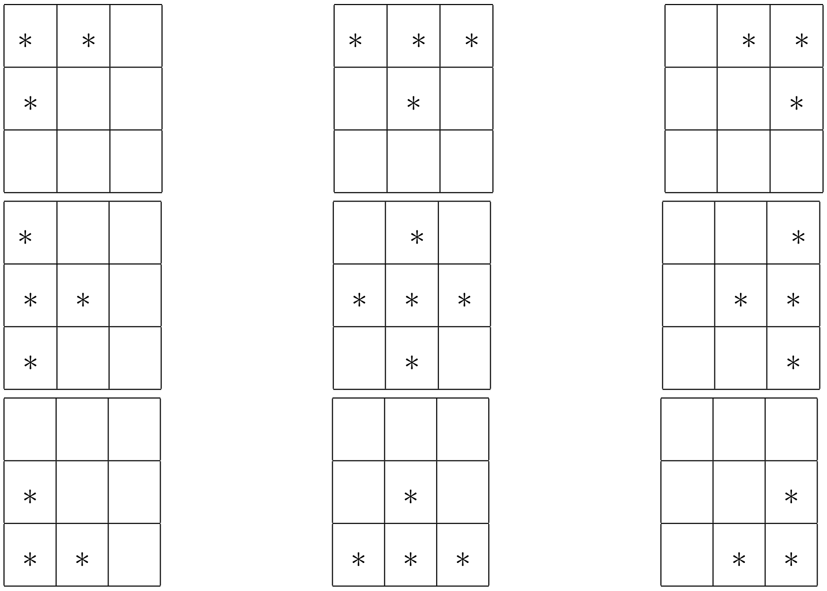
Example 2.2 The game board G_3 is set up with the following vertex labeling. This grid on the left shows what the game board looks like, while the graph on the right shows the vertices and edges of the game board.



v_1	v_2	v_3
v_4	v_5	v_6
v_7	v_8	v_9



The following grids represent the effect of pressing each of the 9 vertices. Pressing v_1 will change the state of the lights with stars as shown in the top left grid. Pressing the vertex v_2 will change the state of the lights directly to the left, right and below it, including v_2 itself. This pattern continues with each vertex of the game board.



□

Any graph G can be expressed in terms of an adjacency matrix in order to more easily examine its characteristics using methods from linear algebra. This is essentially a way to express all of the connections in the graph simultaneously.

Definition 2.3 A *game board adjacency matrix*, A_n for graph G_n is an n^2 x n^2 matrix with $1 \leq i, j \leq n$, where for all i and j, a 1 in the ith row and jth column indicates an edge between v_i and v_j. A_n is sometimes denoted as A to simplify notation.

Example 2.4 The adjacency matrix for G_3 is the following matrix. The graph of G_3 from the previous example can be used to determine where the 1 entries will occur in the matrix. Each row represents a vertex, and the entries in each row represent which lights will be affected from that particular button press. The second matrix has been broken up into $n \times n$ blocks to help illustrate the concepts in the remainder of this section.

$A_3 =$

$$\begin{bmatrix} 1&1&0&1&0&0&0&0&0\\ 1&1&1&0&1&0&0&0&0\\ 0&1&1&0&0&1&0&0&0\\ 1&0&0&1&1&0&1&0&0\\ 0&1&0&1&1&1&0&1&0\\ 0&0&1&0&1&1&0&0&1\\ 0&0&0&1&0&0&1&1&0\\ 0&0&0&0&1&0&1&1&1\\ 0&0&0&0&0&1&0&1&1 \end{bmatrix} = \left[\begin{array}{ccc|ccc|ccc} 1&1&0&1&0&0&0&0&0\\ 1&1&1&0&1&0&0&0&0\\ 0&1&1&0&0&1&0&0&0\\ \hline 1&0&0&1&1&0&1&0&0\\ 0&1&0&1&1&1&0&1&0\\ 0&0&1&0&1&1&0&0&1\\ \hline 0&0&0&1&0&0&1&1&0\\ 0&0&0&0&1&0&1&1&1\\ 0&0&0&0&0&1&0&1&1 \end{array}\right]$$

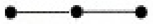
$\square$

Even for small game boards, it can be challenging to keep track of all the connections as this matrix is being assembled. Since these game boards take on a grid shape for the classic game, there is a way to efficiently build the adjacency matrix that will work for any size desired.

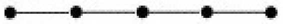
First examine a single row of the game board. It will be a line graph with n vertices since the game board is size $n \times n$, where each vertex is connected to the vertices directly to the left and right creating a line. There is no wrap around in the classic Lights Out game, so the first and last vertex will only have one edge instead of two. An adjacency matrix can be built for these line graphs that will reoccur in the game board adjacency matrix in a predictable way.

Definition 2.5 The *line graph adjacency matrix*, $\hat{A}_n$ is the $n \times n$ adjacency matrix for the line graph with n vertices.

Example 2.6 The line graph for game board G_3 is labeled as follows:



with corresponding $\hat{A}_3 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$



with corresponding $\hat{A}_5 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$

□

After working through a few different sizes of $\hat{A}_n$, and considering the structure of the graph, it becomes clear that the line graph adjacency matrix will always have all ones down the three center diagonals. This makes a predictable pattern for $\hat{A}_n$ for any size of line graph.

The line graph adjacency matrix quickly describes the connections that occur horizontally between vertices. To describe the vertical connections, all that is needed is an $n \times n$ identity matrix, labeled $\hat{I}_n$. This makes sense because when considering the button press of a specific row, only the vertices directly below (or above) will be changed.

There are also going to be some rows of the game board that will not be effected by button presses on a specific row. This calls for a matrix of $n \times n$ size containing all zeros. For simplicity, call this $\hat{0}_n$.

The process of building the game board adjacency matrix is the logical placement of these 3 kinds of matrices based on the structure of a grid graph. Although A is an $n^2 \times n^2$ matrix, for sake of construction consider A to be an $n \times n$ matrix with each entry being an $n \times n$ matrix. This will create the desired $n^2 \times n^2$ dimension of A. The entries along the diagonal will all be

$\hat{A}_n$, the entries directly above and below the diagonal will be $\hat{I}_n$ and all the remaining entries will be $\hat{0}_n$.

Here is a general method for constructing A_n:

$$A_n = \begin{bmatrix} \hat{A}_n & \hat{I}_n & \hat{0}_n & \dots & \hat{0}_n \\ \hat{I}_n & \hat{A}_n & \ddots & \ddots & \vdots \\ \hat{0}_n & \ddots & \ddots & \ddots & \hat{0}_n \\ \vdots & \ddots & \ddots & \hat{A}_n & \hat{I}_n \\ \hat{0}_n & \dots & \hat{0}_n & \hat{I}_n & \hat{A}_n \end{bmatrix}$$

The following picture is the same A_n matrix with each component highlighted in order to illustrate the pattern.

$$\begin{bmatrix} \boxed{\hat{A}_n} & \hat{I}_n & \hat{0}_n & \dots & \hat{0}_n \\ \hat{I}_n & \boxed{\hat{A}_n} & \ddots & \ddots & \vdots \\ \hat{0}_n & \ddots & \boxed{\ddots} & \ddots & \hat{0}_n \\ \vdots & \ddots & \ddots & \boxed{\hat{A}_n} & \hat{I}_n \\ \hat{0}_n & \dots & \hat{0}_n & \hat{I}_n & \boxed{\hat{A}_n} \end{bmatrix} \begin{bmatrix} \hat{A}_n & \boxed{\hat{I}_n} & \hat{0}_n & \dots & \hat{0}_n \\ \boxed{\hat{I}_n} & \hat{A}_n & \boxed{\ddots} & \ddots & \vdots \\ \hat{0}_n & \boxed{\ddots} & \ddots & \boxed{\ddots} & \hat{0}_n \\ \vdots & \ddots & \boxed{\ddots} & \hat{A}_n & \boxed{\hat{I}_n} \\ \hat{0}_n & \dots & \hat{0}_n & \boxed{\hat{I}_n} & \hat{A}_n \end{bmatrix} \begin{bmatrix} \hat{A}_n & \hat{I}_n & \boxed{\hat{0}_n} & \boxed{\dots} & \boxed{\hat{0}_n} \\ \hat{I}_n & \hat{A}_n & \ddots & \boxed{\ddots} & \boxed{\vdots} \\ \boxed{\hat{0}_n} & \ddots & \ddots & \ddots & \boxed{\hat{0}_n} \\ \boxed{\vdots} & \boxed{\ddots} & \ddots & \hat{A}_n & \hat{I}_n \\ \boxed{\hat{0}_n} & \boxed{\dots} & \boxed{\hat{0}_n} & \hat{I}_n & \hat{A}_n \end{bmatrix}$$

I developed a simple program called GenAdj that quickly generates an adjacency matrix for any square game board with the classic + connections. The code for this program can be found in the appendix.

This matrix can be checked for accuracy by looking at how an adjacency matrix is traditionally built. Each row represents a vertex, and the entries in each column represent where the connections are to that vertex. Therefore this method of construction produces the desired results of the adjacency matrix.

2.2 Light States and Configurations

This adjacency matrix represents the connections of the game board, which are unchanging during game play. However, there is an important component of the game that is constantly changing, which is the state of the lights. At each moment in the game, the current state of the game board can be explicitly represented in terms of symbols representing which lights are on and which ones are off. In order to do this simply, the values for on and off can be symbolized using numbers.

Definition 2.7 The possible states of the lights in the game are represented by the *light state set.* For the classic Lights Out game, $S = \{0, 1\}$ where 0 represents off and 1 represents on. For other variations with multiple colors, different numbers can be assigned to each color for a larger light state set.

Between each play of the game from start to finish, the game board has some combination of lights on and off. Each moment of the game is represented as a configuration which gives the player the exact information about the state of the entire game board.

Definition 2.8 A *configuration* C, is a matrix or vector describing which lights of the game board are on and which are off. Each of the vertices are assigned a number based on their current state. Configuration C can be represented as an $n \times n$ matrix where each position corresponds directly to the position of the vertex on the game board, or as a $1 \times n^2$ row (or column) vector with corresponding entries to the n^2 vertices of the graph.

Example 2.9 Given the game board G_3, suppose that lights v_1, v_5 and v_6 are on and the rest are off. Then the configuration of the board is written as the vector,

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

or as the matrix,

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

$\square$

There are two commonly used configurations in this game. The initial configuration of the game, also called the source configuration is represented as C_s and the end goal, or target configuration is represented as C_t. In the classic Lights Out game, C_s can be any combination of on and off vertices, and the target configuration is always to turn all lights off, or the configuration containing all zero entries.

Definition 2.10 The set of all possible source configurations C_s is called the *configuration space*, denoted C_G for game board G.

For the classic game of Lights Out, C_G is not restricted in any way. The game board can start out with any combination of lights on and off except for the trivial all off configuration. The size of C_G depends on the size of the game board. For any grid G_n, any configuration will have n^2 entries since that is the number of vertices on the graph. Each graph has a possibility of 2 states, therefore there will be 2^{n^2} configurations in C_G. Once the source configuration is determined, the configuration space can possibly become limited depending on the game variation.

Definition 2.11 The *game play space*, C_{C_s} is the set of all possible configurations that can be reached by some combination of button presses, given a specific starting configuration.

The size of C_{C_s} depends on concepts that will be built throughout the remainder of this chapter. C_G and C_{C_s} will be exactly the same if any configuration can be reached by a sequence of button presses, but usually the structure of the adjacency matrix of a game board will limit the size of C_{C_s}. This will be discussed in more detail in Section 3.2.

2.3 Classic Lights Out Game Play

The game of Lights Out is played by using a sequence of moves, called button presses, that help get the player closer to the ending configuration. Each move will change the state of the button pressed and those directly connected to it by an edge. This change of state can be thought of as an increase in state by one. Since there are only two possible states in the classic game of Lights Out, all computations can be done modulo 2. Each vertex is a button that represents a possible different move that can be made by the player.

Definition 2.12 For a graph G_n, a *neighbor matrix* M_i is an $n \times n$ matrix where i corresponds to the v_i matrix that was pressed. A one is placed in each position that represents an increase of state from the v_i button press. Any move can also be represented as a $1 \times n^2$ vector.

Example 2.13 The 9 possible neighbor matrices for all nine vertices on G_3 are:

$$M_1 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} M_2 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} M_3 = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

$$M_4 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} M_5 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} M_6 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$M_7 = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} M_8 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} M_9 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

□

The easiest way to represent how a move of the game changes the state of the vertices is to use matrix addition. If the game board starts with a certain configuration, C with its matrix representation, and one move is made by pressing vertex v_i, the resulting configuration will be $C + M_i$. This will increase the state of all the lights that are connected by an edge with the vertex pressed by one.

This idea can also be extended to row (or column) vector representations of the neighbor matrices and configurations. It can be seen that the ith neighbor matrix, if expressed as a row (or column) vector with n^2 entries, corresponds to the ith row (or column) of the adjacency matrix since the adjacency matrix describes all edge connections of vertices. If e_i represents the row vector with zeros everywhere except for a 1 in the ith entry, then e_iA will represent the ith row of A, which is the effect of pressing vertex i. This means that e_iA is the neighbor matrix M_i now represented as a row vector. Then a new configuration after pressing vertex i can be calculated by $C_s + e_iA$. Similarly for e_i as a column vector with zeros everywhere except for a 1 in the

ith entry, the new configuration will be $C_s + Ae_i$.

Example 2.14 Suppose an initial configuration of the G_3 game board has lights v_2, v_3, v_4, and v_7 on. Assume that v_7 is pressed. Then, using matrix representations, the next configuration of the game board, C' can be determined as follows.

$C_s + M_7 = C'$

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

The same exact calculations can be done if the configurations are written as vectors instead of an $n \times n$ matrix. Here is the same example done using column vectors and the adjacency matrix.

$$C_s + Ae_7 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$\square$

In most cases, more than a single move is needed to solve a game completely. This requires a sequence of moves from the player.

Definition 2.15 A *neighbor sum matrix*, M is an $n \times n$ matrix that is the sum of all move matrices used throughout the game to get from C_s to C_t. If $i_1, i_2, ...i_k$ represents the $j = 1, ..., k$ buttons pressed during the play of a game, then $M = M_{i_1} + M_{i_2} + ... + M_{i_k} = \sum_{j=1}^{k} M_{ij}$.

Since a single move is equivalent to matrix addition, then a sequence of moves is simply a sum of move matrices played throughout the course of a game. This means that if M is result of all the buttons pressed throughout the game, then

$$C_t = C_s + M \tag{2.1}$$

Example 2.16 Suppose the game is being played on G_3, where the source configuration is:

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

It is fairly easy to see that the moves needed to solve this game are to press v_1 and v_9. Now,

$$M = M_1 + M_9 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

To ensure that Equation 2.1 holds true, then $C_s + M$ should equal C_t which is the 3×3 matrix of all 0's.

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

□

It is fairly easy to see now that the order in which the buttons are pressed does not matter for this version of the game. This is based on the fact that moves can be represented as matrix addition and the addition of matrices is commutative.

Next, examine how the number of times a button is pressed affects the game. The moves done by the player can be simplified significantly based on this fact that all calculations are done modulo 2. Consider what occurs when the same vertex is pressed more than one time. If v_i is pressed an odd number of times, this is the same as pressing the vertex a single time since any odd number modulo 2 is one. An example of this would be if v_i is pressed three times, then $M_i + M_i + M_i = M_i$. Similarly, if v_i is pressed an even number of times, this is the same as not pressing the vertex at all since any even number modulo 2 is zero. If v_i is pressed twice, then $M_i + M_i = 0$. When looking at versions of the game that are larger than two states, this result will not hold true, but similar ideas can be extended and computations can be modified for these variations.

The column vector representation of a sequence of M_i's will also produce a similar equation that can be used to solve the game. Since $M_i = Ae_i$ for each individual move, then the neighbor sum matrix $M = M_{i_1} + M_{i_2} + ... + M_{i_k}$ as can be expressed in column vector notation as $A \sum_{j=1}^{k} e_{ij}$. This means that equation 2.1 can be rewritten in terms of column vectors as,

$$C_t = C_s + A \sum_{j=1}^{k} e_{ij} \tag{2.2}$$

A nearly identical equation holds for the row vector representation of configurations.

$$C_t = C_s + \left(\sum_{j=1}^{k} e_{ij} \right) A \tag{2.3}$$

Definition 2.17 A *game move* X, is a row (or column) vector with each entry representing the number of times a button is pressed during game play. This is exactly $\sum_{j=1}^{k} e_{ij}$. The *game move space*, X_G is the collection of all possible game moves for a graph G.

This means that multiplying the game move vector and the adjacency matrix is the same thing as adding each M_i from each v_i played to get M. Therefore, when X is represented as a row vector,

$$XA = M \tag{2.4}$$

So Equation 2.3, with X represented as a row vector, can be rewritten as,

$$C_t = C_s + XA \tag{2.5}$$

This form of the equation is more useful when trying to determine exactly which vertices need to be pressed. Before looking at how to solve for the winning game moves, here is an example of this equation at work.

Example 2.18 Suppose the G_3 game board starts with the source configuration given in Example 2.16. Then,

$$C_s = \left[\begin{array}{ccccccccc} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{array} \right]$$

If vertices v_1 and v_9 are pressed, then the game move vector will be,

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

In order to determine the new configuration of the board after pressing these vectors using Equation 2.5, first calculate XA:

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

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

For sake of comparison, XA will be exactly same as M calculated in Example 2.16 written as a matrix instead of a vector.

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

Then to determine the ending configuration from this given initial condition and game moves,

$$C_s + XA = C_t$$

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

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

□

Here is another example where the vertices pressed do not turn all the lights out but the two representations are compared.

Example 2.19 Suppose the G_3 game board starts with lights v_2 and v_5 on. Then,

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

If vertices v_1, v_6 and v_8 are pressed, then the game move vector will be,

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

Then C_t, the new configuration of the board after these buttons are pressed, can be calculated first by finding XA,

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

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

So using Equation 2.5,

$$C_t = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

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

This means that given the starting configuration, pressing v_1, v_6 and v_8 will leave all lights on except for v_2 and v_9.

To solve this same problem using Equation 2.1, C_s would be written in its matrix form, so

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

Now calculate M as follows,

$$M = M_1 + M_6 + M_8 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Then using Equation 2.1

$$C_t = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Which is identical to the C_t solved for using the other equation. $\square$

The ability to describe the game using matrix operations, and the skill of switching between matrix and vector form become extremely helpful in the development of ideas surrounding this game.

2.4 The μ Mapping

Not all versions of the game can be thought of using matrix multiplication and addition. For the purpose of extending these concepts to other versions, consider thinking of moves in a more broad sense using algebraic mappings.

When it is desired to determine the state of a single vector given a configuration without looking at the entire row vector, C can be thought of as a function, $C : V \to S$ where $C(v)$ represents the state of the vertex v. In the classic game, $C(v) = 1$ if v is on and $C(v) = 0$ if v is off for any $v \in V$.

Now, the game can be expressed as a mapping, $\mu : V \times C_G \to C_G$ where

$$\mu(v, C)(u) = \begin{cases} C(u) + 1 & \text{if } (u, v) \text{ is an edge in } G \\ C(u) & \text{otherwise} \end{cases}$$

Note that all calculations for any representation of the μ mapping are done mod $|S|$, where $|S|$ is the order of set S. This is because the codomain C_G consists of all elements of this form.

So $\mu(v, C)$ represents the configuration that results after the effect of pressing vertex v in configuration C. Since most games require more than one vertex to be pressed, the definition of μ can be slightly altered for an entire game move vector. The game move vector X can be thought of as a mapping where $X : V \to \mathbb{N}$ where $X(v)$ is the number of times vertex v is pressed. Then an alternate way of defining μ is, $\mu : X_G \times C_G \to C_G$ where

$$\mu(X, C)(v) = C(v) + X(v)$$

So $\mu(X, C)$ would represent the configuration that results after the effect of game moves X after starting in configuration C.

While the purpose of defining μ is to be able to extend these ideas into other versions of the game, consider what μ looks like for the classic game of Lights Out. The μ mapping will be defined using Equation 2.5. Therefore, for the classic game of Lights Out,

$$\mu(X, C) = C + XA \qquad (2.6)$$

All of the critical definitions developed in this chapter so far have helped to unfold a mathematical structure to the classic Lights Out game that will serve as a launching point for further understanding of the game.

2.5 The Solve Matrix

For any given initial configuration C, the effect of a sequence of button presses X can be computed with Equation 2.6, $\mu(X, C) = C + XA$. The matrix A is symmetric, meaning $A = A^T$, and X can be represented equivalently as a row or column vector. So if X is a column vector, then $XA = AX^T$ therefore Equations 2.6 can also be written as $\mu(X, C) = C + AX^T$ and the only thing that changes is the representation of X from a row matrix to a column matrix, both of which are equivalent in theory. This definition of μ is useful when the initial configuration and the sequence of button presses are known. In the game of Lights Out, only the initial configuration and the goal configuration are immediately known to the player. With a little work, the equation for $\mu(X, C)$ can be adjusted accordingly.

The most commonly used case of this μ mapping would be where C_s is the starting configuration, and $C_t = \mu(X, C_s)$. As stated earlier in Chapter 2, this is exactly Equation 2.5, $C_t = C_s + XA$.

Now X becomes the desired configuration to solve for in order to know which buttons to press to solve the game. Then X can be found using basic algebra modulo 2.

$$\begin{aligned} C_s + XA &= C_t \\ XA &= C_t - C_s \\ X &= (C_t - C_s)A^{-1} \end{aligned}$$

This process of solving for X depends entirely on the assumption that

A is an invertible matrix. Section 2.6 will discuss how to calculate this inverse of A whenever it is possible, however that is not always the case. It turns out that not all of the adjacency matrices are guaranteed to be invertible, which suggests the need for an alternate approach. This concept will be explored in Section 2.7.

The equation for X can be simplified even further. Since the target configuration for the classic game of Lights Out is to turn all lights off, $C_t = 0$. Also, since all calculations are done mod 2, addition is equivalent to subtraction. Therefore,

$$X = C_s A^{-1} \tag{2.7}$$

For an example of this equation at work, consider the invertible game board of G_3. The methods used to determine A^{-1} are from linear algebra and discussed in more detail in the next section along with the method for non-invertible matrices.

Example 2.20 Let C_s be the starting configuration on game board G_3 where

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

In order to determine what vertices need to be pressed, first calculate the inverse of A with entries mod 2. Then using Equation 2.7, the button presses needed to solve this game can be found as follows.

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

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

Therefore the vertices that need to be pressed to turn all the lights out would be $v_1,\ v_4,\ v_6$ and v_7. □

This method is effective, and is the most direct conclusion from Equation 2.5 . There is a way to simplify this process using the fact that the configuration matrix consists of entries that are either 0 or 1. This is especially helpful as the matricies increase in size.

For any C_s consisting of entries either 0 or 1, the multiplication C_sA^{-1} can be expressed as $\sum\limits_{i \in V} a_i$ where V is the set of indices corresponding to the nonzero entries in C_s and a_i is the ith row of A^{-1}. This works because of the way that matrix multiplication is defined. Each entry of C_s is multiplied by each entry of the corresponding column in A^{-1}. This summation removes all the entries from C that end up being multiplied by 0 and do not affect the outcome of the result. A way to turn this idea into an organized procedure is to use a submatrix of A^{-1} to solve.

Definition 2.21 A *solve matrix* A_X^{-1} is a submatrix of A^{-1} consisting of only rows from A^{-1} that correspond to the lights that are on in the initial configuration.

Adding down along the columns of the solve matrix produces the exact game move vector that is desired to solve the game and turn all lights out. This is exactly the process of multiplying, and gives structure to the process of $\sum_{i \in V} a_i$.

Example 2.22 Let C_s be the starting configuration on game board G_3 where

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

Then

$$A_X^{-1} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Adding down along the columns of A_X^{-1} produces the game move vector.

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

□

Any game board with an invertible adjacency matrix can be solved in this way. This means that given any initial configuration, there is a game move vector that will solve the game.

2.6 Invertibility

There are several ways to determine if a matrix is invertible using techniques from linear algebra. To simply answer the question of invertibliity, calculating the determinant modulo 2 can be used. Most computer algebra systems (CAS) such as Matlab or Mathematica are equipped with the ability to calculate the rank of a matrix. The most useful approach for determining invertibility for these games would be to use elementary matrix operations to get the matrix into reduced row echelon form. This may require more computations than the determinant or rank however the results from doing these calculations can be used regardless of the outcome of invertibility.

The inverse of a matrix is calculated by augmenting the matrix with an identity matrix of the same size, and row reducing until the identity is in place of the original matrix. The number of nonzero rows left in the original matrix after it is in reduced row echelon form represents the rank of the matrix. If the rank is the same as n for an $n \times n$ matrix, then the matrix is invertible.

Out of the first 20 $n \times n$ game boards, the following twelve out of the twenty have invertible adjacency matrices. They are $G_1, G_2, G_3, G_6, G_7, G_8, G_{10}$, $G_{12}, G_{13}, G_{15}, G_{18}$ and G_{20}. As of my research right now, there is no clear pattern for anticipating which values of n produce invertible game boards, but ideas as to how this pattern could be found will be discussed in the next chapter.

Example 2.23 The game board G_2 is a unique example where $A = A^{-1}$.

$$G_2 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \qquad G_2^{-1} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

□

2.7 Pseudo-Inverse

Having an invertible game board matrix is the ideal situation for solving what game moves are needed to win a game. However, when the matrix turns out to be non-invertible, all is not lost. It turns out that a pseudo-inverse can be used to determine solutions to any game, which will be explored in Section 2.8. First, a general process of determining a pseudo-inverse is needed.

Definition 2.24 A *pseudo-inverse* P, of matrix A is an $n \times n$ matrix such that $PA = A'$ where A' is the reduced row echelon form of A. When A is invertible, P is simply A^{-1}.

When a matrix is not invertible, a pseudo-inverse can be calculated by row reducing the augmented matrix $[\ A \mid I\ ]$ to get a matrix $[\ A' \mid P\ ]$ where A' is the reduced row echelon matrix of A and P is a pseudo-inverse for A. Each step in the process of reducing a matrix into reduced row echelon form corresponds to a multiplication by an elementary matrix E. Therefore to get A into reduced row echelon form, A will be multiplied by a sequence of elementary matrices $E_nE_{n-1}...E_2E_1$ such that $E_nE_{n-1}...E_2E_1A = A'$. If the matrix is invertible the inverse will be the result of multiplying all the elementary matrices, so $A^{-1} = E_nE_{n-1}...E_2E_1$. If a matrix is singular, then there are multiple ways to row reduce and get a row of zeros. The pseudo inverse will also be the result of multiplying all the elementary matrices together, so $P = E_nE_{n-1}...E_2E_1$, but there are several sequences of elementary matrices that will reduce A into echelon form.

Calculating a pseudo inverse of these matrices can be somewhat of a challenge for several reasons. Their large size makes it difficult to do these computations by hand, however the aid of a computer provides its challenges as well. Most Computer Algebra Systems (CAS) are able to calculate the pseudo-inverse with a series of commands. I primarily used Matlab to explore these matrices and had to try several approaches.

The first method of determining a pseudo inverse would be to augment the adjacency matrix (using GenAdj) and identity of proper size and then use the reduced row echelon form command (rref) making sure the format of outputs is rational numbers. Then take the common denominator of all entries, multiply the matrix by this number, and finally simplify all entries modulo 2.

This method is problematic whenever the common denominator of all entries is an even number. After multiplying all numbers by an even number, then taken modulo 2, the resulting pseudo-inverse will be entirely zeros. Therefore an alternate method is needed in order to ensure the pseudo inverse is properly acquired.

Example 2.25 Here is a matrix where the first method of calculating the pseudo-inverse fails.

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

The process of computing the inverse begins with augmenting with the identity and row reducing to this point.

$$\left[\begin{array}{ccc|ccc} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right] \backsim \left[\begin{array}{ccc|rrr} 1 & 0 & 0 & -1/2 & 1/2 & 1/2 \\ 0 & 1 & 0 & 1/2 & -1/2 & 1/2 \\ 0 & 0 & 1 & 1/2 & 1/2 & -1/2 \end{array}\right]$$

Clear the denominator of the right matrix, to get,

$$\begin{bmatrix} -1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{bmatrix}$$

Taken modulo 2, the proposed pseudo matrix is

$$P = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Now to check, multiplying A and P should produce A', which in this case is the identity matrix. However,

$$AP = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Therefore, this process may not be effective in determining the pseudo-inverse of a matrix, so another method is needed. □

The second, and truly accurate method of determining the pseudo-inverse requires a different approach when trying to compute. I developed a program, InvSol, whose code can be found in the appendix, that will solve for the inverse of a matrix doing each step modulo 2 instead of waiting until the end. Following this process ensures that the pseudo-inverse calculated will actually satisfy $PA = A'$.

It is important to note that the pseudo inverse will additionally be an invertible matrix. Since the process of finding the pseudo inverse begins with the identity and a series of row operations are preformed to generate the pseudo inverse, these steps can be undone in the reverse order, returning P to the identity. Therefore P is invertible.

A pseudo inverse of a game board holds several important pieces of information. In order to illustrate these findings, examine the adjacency matrix of the traditional Lights Out game board, G_5, its reduced row echelon form, and its pseudo-inverse. The three matrices on the next three pages are all important to reference throughout the remainder of this thesis.

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

Figure 2.1: A: Adjacency matrix for the game board G_5

$$\left[\begin{array}{ccccccccccccccccccccccccc}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{array}\right]$$

Figure 2.2: A': Reduced row echelon form for the game board G_5

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

Figure 2.3: P: Pseudo inverse matrix for the game board G_5 (computed using InvSol program in MatLab)

2.8 Using the Pseudo-Inverse

The inverse or pseudo-inverse of an adjacency matrix are an important tool to use for determining the solutions to a game of lights out. Equation 2.7 showed that the solutions of buttons that need to be pressed can be determined by multiplying the starting configuration of the board by the inverse of the game board. Luckily, when the game board is not invertible, a pseudo-inverse works in the exact same way as a true inverse does to determine these solutions.

Finding game moves that will solve the game amounts to finding an X such that $XA = C_t - C_s$, as discussed earlier. Since X can be written as a row or column vector, A is symmetric, $C_t = 0$ and $-C_s = C_s$ modulo 2, this equation is equivalent to finding X for $AX = C_s$. When A does not have an inverse, X cannot be found by multiplying both sides by A^{-1} as it was before. Instead, multiply both sides of the equation by the pseudo-inverse P. Then to find X, solve $PAX = PC_s$.

Since $PA = A'$ by definition of pseudo-inverse, the system becomes $A'X = PC_s$, which can be solved in the way systems are traditionally solved. When A is not invertible, A' will have rows of all zeros at the bottom, so there will be multiple solutions to the system depending on the free variables. A further look into this will come in Chapter 3. For now, it is only important to notice that the particular solution will in fact be a possible solution to the system.

Example 2.26 For the game board G_5, suppose the starting configuration is first written in matrix form to more easily visualize, and then in vector form to do computations with. In order to save space, C_s is written as a row vector, but in computations is used as a column vector.

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

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

Since A' is already in reduced row echelon form, the particular solution to the system $A'X = PC_s$ will be the vector PC_s taken modulo 2. In order to be seen more clearly, X has been written in matrix form.

$A'X = PC_s$

$$\begin{bmatrix} \\ \quad A' \quad \\ \\ \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_{25} \end{bmatrix} = \begin{bmatrix} \\ \quad P \quad \\ \\ \end{bmatrix} \begin{bmatrix} \\ C_s \\ \\ \end{bmatrix}$$

So solving the system can be done by augmenting the A' matrix, and the vector found by multplying PC_s.

$$\left[\begin{array}{c|c} & \\ \quad A' \quad & PC_s \\ & \end{array}\right]$$

All solutions to this system can be found by parametrizing the free variables, as found in A'. This will be done in more detail later, but for now, the vector resulting in multiplying P and C_s and inserting zeros in the components corresponding to the free variables, will be a particular solution. For G_5 the free variables are the last two components. So PC_s is a solution provided its last two components are zero since these correspond to the free variables of A. One possibility for a game move X would be,

$$X = PC_s = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

As observed, the last two components are both zero so this is a solution. Therefore these 15 button presses will solve this starting configuration.

□

This method will always generate a solution of game moves, but unfortunately, not all starting configurations of a non invertible game board are actually solvable. In Section 3.2 the number of solvable configurations will be discussed, including how to determine if a configuration is solvable.

Chapter 3

Quiet Patterns

When an adjacency matrix is invertible, the only way to leave the state of all the lights unchanged is to not press any lights at all. When an adjacency matrix is not invertible, things become significantly more interesting. This chapter will explore the the many different aspects of these special game moves and how they affect the game.

3.1 Defining Quiet Patterns

Solving a system such as $AX = C_s$ for a matrix A that is not invertible means that there will be multiple solutions for any solvable initial configuration. The reason for this is because of how the general solution of a system is calculated. In terms of the game, this means there will exist at least one sequence of buttons that can be pressed which will result in no change in the state of the lights overall. These sequences of moves are given the name quiet patterns.

Definition 3.1 A *quiet pattern* q is a sequence of game moves that leave the state of all lights on the board unchanged when pressed.

In terms of linear algebra, quiet patterns are exactly the elements of the null space of a matrix.

Definition 3.2 The *null space* of a matrix A, $null(A)$, is all vectors X such that $AX = 0$. When a matrix is not invertible, the null space is non-trivial.

These game moves are important in understanding the inner workings of the game and have several interesting properties of their own that will be examined throughout this chapter.

3.2 Solvability Regarding Quiet Patterns

When a game board does not have an invertible matrix, there is a possibility of having an initial condition that is unsolvable. In fact this occurs more frequently than one would anticipate. When developing the game, it was important to program only solvable starting configurations, otherwise the game would have been immensely more frustrating and probably not as popular. The set of all quiet patterns of a game board are directly linked to the starting configurations that will be solveable.

If the goal is to produce a solvable starting configuration, it is simple to work backwards. There are only a finite number of game moves that are possible for any G_n, each one constructed by pressing a unique combination of lights. In order to determine a possible starting configuration, consider the game going backwards. Start with all lights on the game board being off and press some game move X. This will result in a configuration $C_X = AX$. Each unique game move X will produce an initial condition that is solvable by that game move.

The solvable configurations can also be discussed in a more general sense. The game of Lights Out is essentially a linear transformation of configurations of the game board, where multiplication of the adjacency matrix and some game move AX, transforms the state of the lights. Then $C_s \mapsto C_t$ under this linear transformation, call it Ω. The vector C_X represents the configuration of the board after a game move vector X has been pressed. A Lights Out game is solvable when the equation $C_s + C_X = C_t$ has a solution, or when manipulated modulo 2, when $C_X = C_s + C_t$. Define the linear transformation to be a mapping

$$\Omega : \mathbb{Z}_2^{n^2} \to \mathbb{Z}_2^{n^2} \text{ where } \Omega(AX) \mapsto C_s + C_t$$

Then only elements in the range of this mapping will have corresponding game move vectors that solve the game. Using concepts from linear algebra, these solvable configurations can be related to the quiet patterns in the following way.

Theorem 3.3 Let $Y = C_s + C_t$. The configuration Y is in the range of Ω ($AX = Y$ has a solution X) if and only if Y is orthogonal to the basis of all quiet patterns.

Proof: Let $Y = C_s + C_t$.
$AX = Y$ has a solution, meaning Y is in the range of Ω

$\Leftrightarrow$ $Y \in Col(A)$ where $Col(A)$ is in the column space of A, meaning Y is a linear combination of the columns of A. If each of the columns of A are labeled C_i for $1 \leq i \leq n^2$, and each entry of X represented as x_i, then $AX = x_1C_1 + x_2C_2 + ... + x_{n^2-1}C_{n^2-1} + x_{n^2}C_{n^2}$ by how matrix multiplication is defined.

$\Leftrightarrow Y \in Row(A)$ where $Row(A)$ is in the row space of A since A is symmetric, meaning $A = A^T$.

$\Leftrightarrow Y \in row(A)^{\perp\perp}$ since for any finite dimensional vector space, taking the orthogonal complement twice returns you to the original subspace, $V^{\perp\perp} = V$

$\Leftrightarrow Y \in null(A)^{\perp}$. This is because $null(A) = Row(A)^{\perp}$ since $AX = 0$ if and only if each row of A is orthogonal to X.

$\Leftrightarrow Y$ is orthogonal to the basis of all quiet patterns ($null(A)$).

∎

This means that a solvable configuration is one that is orthogonal to the basis of the null space. More generally stated,

$$AX = C_s \text{ has a solution } X \Leftrightarrow \; C_s \cdot q = 0 \;\; \forall q \in null(A)$$

Therefore, once the basis of the null space of a game board is found, solvable configurations can be determined by checking orthogonality. This check can help determine if using the pseudo-inverse to find a game move solution will work.

Provided that a solution exists to a chosen configuration, game boards that have quiet patterns will have multiple ways to solve any given configuration. Adding a quiet pattern to a solution produces a new solution, therefore the number of unique quiet patterns will determine exactly the number of ways a specific configuration can be solved.

The number of quiet patterns a board has will be exactly the size of the null space of A. This value can be found once more is known about how to calculate the basis of the null space, and from there, how many elements this basis produces.

3.3 Finding the Spanning Set of the Null Space

In order to determine which sequences of button presses are quiet patterns, the spanning set of the null space must be calculated. This is the best way to determine all possible quiet patterns for any size game board. Traditionally this is done by looking at the reduced row echelon form of the adjacency matrix, and parametrizing the non-pivot columns of A'.

In general, for any consistent system $AX = B$ the solution X can be found by augmenting A and B to get the matrix $[\ A \mid B\ ]$ and row reduce A until it is in reduced echelon form. When A is invertible, row reducing the left will result in the identity matrix leaving the new augmented matrix $[\ I \mid B'\ ]$ where B' is the only solution to the system.

When A is not invertible, the reduced row echelon form of A becomes A' on the left, yielding PB on the right where P is a pseudo-inverse. This means that after row reducing, the augmented matrix becomes $[\ A' \mid PB\ ]$. A particular solution of the system can be found by creating a vector with zeros in the entries corresponding to free variable and filling in the rest with the entries of PB that do not correspond to zero rows.

When there are multiple solutions to the system, a generic solution is assembled by parameterizing the free variables. This is done by looking at the non-pivot columns of A' which are the columns that do not contain any leading entries. The number of such columns describes the number of parameters needed to write a general solution to the system.

A general solution is found by adding a particular solution to the generic solution.

Example 3.4 In order to find the general solution for a given system, here is a system that will help illustrate the process. The following matrix has already been put into reduced row echelon form.

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

Notice how column 3, 5 and 6 are non pivot columns because they do not contain any leading entries (which are bold). This means three parameters are needed for the three free variables in order to define the general solution, call them $t_1 = x_3$, $t_2 = x_5$, and $t_3 = x_6$.

First, a particular solution can be found by looking at the right column, corresponding to PB. The top three entries are the ones that correspond to nonzero rows of the matrix and are 1, 0, 1. Since 3, 5 and 6 are the free variables, first put zeros in these entries. Then fill in the entries labeled with stars with the entries in PB that correspond to nonzero rows.

$$\begin{bmatrix} * \\ * \\ 0 \\ * \\ 0 \\ 0 \end{bmatrix} \qquad\qquad \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Now, the generic solution comes from taking each of the non pivot columns and replacing the free variable terms with 0 or 1 depending on which column is being looked at. For example, column 3 should have a 1 in the third entry and zeros in the fifth and sixth entries since it corresponds to the free variable x_3. Column 5 will have a 1 in the fifth entry but zeros in the third and sixth entry. And column 6 will have a 1 in the sixth entry and zeros in the third and fifth entry. Each column should be multiplied by it's corresponding parameter, and then added together to produce the generic solution,

$$t_1 \begin{bmatrix} 1 \\ 1 \\ \mathbf{1} \\ 0 \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} + t_2 \begin{bmatrix} 1 \\ 0 \\ \mathbf{0} \\ 1 \\ \mathbf{1} \\ \mathbf{0} \end{bmatrix} + t_3 \begin{bmatrix} 1 \\ 1 \\ \mathbf{0} \\ 0 \\ \mathbf{0} \\ \mathbf{1} \end{bmatrix}$$

The general solution is found by adding a particular solution to the generic solution. A particular solution to this matrix is the augmented right column. So a general solution to this system will be,

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + t_1 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + t_2 \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} + t_3 \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

□

Now, the system needed to determine the basis of the null space is $AX = 0$ so the particular solution will be the zero vector, meaning that the generic solution will be the general solution for the null space.

Example 3.5 To calculate the spanning set of the null space for the G_5 game board using the reduced row echelon form of the matrix, examine the matrix shown in figure 2.2. Consider the system $AX = 0$ where X is the vector with entries $x_1, x_2, ..., x_{25}$. Looking at the matrix of A', the reduced row echelon form of A, it can be seen that the non-pivot columns of A' are the last two columns, which are columns 24 and 25.

Define a vector v to be column 24, rows $1 - 23$ of A' and vector w to be column 25 rows $1 - 23$ of matrix A'. Then, the free variables in this example are $t_1 = x_{24}$ and $t_2 = x_{25}$ since columns 1-23 are all columns that contain pivot positions. Then the general solution X can be written as,

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{24} \\ x_{25} \end{bmatrix} = t_1 \begin{bmatrix} v \\ 1 \\ 0 \end{bmatrix} + t_2 \begin{bmatrix} w \\ 0 \\ 1 \end{bmatrix}$$

Label $q_1 = \begin{bmatrix} v \\ 1 \\ 0 \end{bmatrix}$ and $q_2 = \begin{bmatrix} w \\ 0 \\ 1 \end{bmatrix}$

Therefore $\{q_1, q_2\}$ form the basis of the null space of G_5.

In order to conserve space and to examine this basis in more detail, the quiet patterns q_1 and q_2 can be written in matrix form instead of vector form,

$$q_1 = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix} \quad q_2 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

$\square$

This spanning set of the null space is exactly the basis of the space of all quiet patterns for the game. This means that not all the quiet patterns have been found, but they can all now be determined.

Every linear combination of the elements that define the basis of the null space will be an element of the null space of A. What this means in terms of the game is that any linear combination of quiet patterns will result in a new quiet pattern. Luckily the number of linear combinations can be narrowed down significantly, since pressing a single buttons an odd number of times is the same effect as pressing the button once, and an even number of times is the same as not pressing a button at all. The same results hold for any sequence of buttons, pressing all the lights from q_1 one time is the same as pressing these buttons any odd number of times. This means that all possible quiet patterns from the game board will be linear combinations of the basis with coefficients either 0 or 1.

Example 3.6 From the previous examples, it was shown that the basis of the null set for G_5 is $\{q_1, q_2\}$. Since the only weights that will possibly yield a new quiet pattern are 0 and 1, the following list will be all possible quiet patterns of G_5.

$0q_1 + 0q_2 = 0$, the trivial quiet pattern of not pressing any lights

$1q_1 + 0q_2 = q_1$

$0q_1 + 1q_2 = q_2$

$1q_1 + 1q_2 = q_3$, a new quiet pattern

$$q_1+q_2 = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Therefore G_5 has four quiet patterns, $0, q_1, q_2$ and q_3.

□

The method of parametrizing to find the spanning set of the null space of A may be simplified due to some important properties of A' for Lights Out game boards.

3.4 Row Reducing A

One of the most critical pieces of anticipating patterns in the game of Lights Out for any size n is fully understanding the adjacency matrix. As shown previously, the adjacency matrix for an $n^2 \times n^2$ game board can be built with $n \times n$ matrix entries with the following structure.

$$A_n = \begin{bmatrix} \hat{A}_n & \hat{I}_n & \hat{0}_n & \dots & \hat{0}_n \\ \hat{I}_n & \hat{A}_n & \ddots & \ddots & \vdots \\ \hat{0}_n & \ddots & \ddots & \ddots & \hat{0}_n \\ \vdots & \ddots & \ddots & \hat{A}_n & \hat{I}_n \\ \hat{0}_n & \dots & \hat{0}_n & \hat{I}_n & \hat{A}_n \end{bmatrix}$$

Row reducing this A_n produces a similar structure for any size n. First, subdivide the adjacency matrix of A_n into a 2×2 as follows,

$$A_n = \left[\begin{array}{cccc|c} \hat{A}_n & \hat{I}_n & \hat{0}_n & \dots & \hat{0}_n \\ \hline \hat{I}_n & \hat{A}_n & \ddots & \ddots & \vdots \\ \hat{0}_n & \ddots & \ddots & \ddots & \hat{0}_n \\ \vdots & \ddots & \ddots & \hat{A}_n & \hat{I}_n \\ \hat{0}_n & \dots & \hat{0}_n & \hat{I}_n & \hat{A}_n \end{array}\right] = \left[\begin{array}{c|c} v & 0 \\ \hline M & w \end{array}\right]$$

The submatrix M is invertible since it is upper triangular with ones down the diagonal, therefore M^{-1} exists.

Then working with this 2×2 version of A_n, a matrix can be found that will reduce A_n to,

$$\left[\begin{array}{c|c} 0 & M^{-1} \\ \hline 1 & -vM^{-1} \end{array}\right] \left[\begin{array}{c|c} v & 0 \\ \hline M & w \end{array}\right] = \left[\begin{array}{c|c} I & M^{-1}w \\ \hline 0 & -vM^{-1}w \end{array}\right]$$

Since $-1 = 1$ when taken modulo 2, A_n can always be row reduced into the form,

$$\left[\begin{array}{c|c} I & M^{-1}w \\ \hline 0 & vM^{-1}w \end{array}\right]$$

This means that the structure of A after row reducing will have the last n rows as all zeros except for the farthest right $n \times n$ matrix $vM^{-1}w$. The important observation to make here is that there will only be at most n non-pivot columns of A once it is put into reduced row echelon form and they will only exist in the lower $n \times n$ corner of the A' matrix.

Non-pivot columns of A' are directly linked to finding the basis of the null space. After working with several adjacency matrices of various sizes, and experimenting with n as large as 70, an interesting pattern arose.

Conjecture 3.1 Non-Pivot Columns of A'
For r equal to the rank of A, then the last $n^2 - r$ columns of A' will be the only non-pivot columns of A'.

Every matrix that I have tried using my InvSol Matlab program shares the same structure for A', where the last $n^2 - r$ rows are all zeros, and the only non-pivot columns are the furthest right $n^2 - r$ columns.

Proving that this structure holds true for any A' will be a difficult task. It depends entirely on what $vM^{-1}w$ looks like for any size n. Exploring the structure of this important $n \times n$ matrix will be covered in more detail in Chapter 5.

Even though this structure can not be proven rigorously at this time, there are some important conclusions that can be made if it does hold true. Consider a general non-invertible A where row reducing A leads to a submatrix A'' in row echelon form and $n^2 - r$ rows of zeros at the bottom.

$$A' = \begin{bmatrix} & & \\ & A'' & \\ & & \\ & & \\ 0 & \cdots & 0 \end{bmatrix}$$

When the non-pivot columns are the furthest right columns only, a basis of the nullspace of A can be obtained by augmenting the non pivot columns of A'', call this submatrix A''', and creating the following matrix with the appropriate size identity matrix I. The identity matrix will have the same number of rows and columns as the number of non-pivot columns in A'.

$$\left[\begin{array}{c} A''' \\ \hline I \end{array}\right]$$

Each column of this augmented matrix will be an element of the basis of the null space of A.

Example 3.7 For the game board G_4, the reduced row echelon form of A is the following matrix. Notice that the last four columns are the non-pivot columns.

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

Since the last four columns are the only non-pivot columns, these four columns, disregarding their zero rows at the bottom, will create the following sub-matrix. A 4×4 identity is augmented on the bottom of the sub-matrix.

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

The columns of this sub-matrix will be a basis for the null space of G_4, and will generate all the quiet patterns for this game board.

□

If it can be proven that A'_n shares this unique structure for any n, this leads to an important observation. Since the basis for the quiet patterns can be found using the last $n^2 - r$ columns of A', and A is symmetric (meaning $A = A^T$), there is another way to determine a basis of quiet patterns for A.

Proposition 3.1 Quiet Patterns for G_n Found in P

If A_n is a symmetric, non-invertible $n^2 \times n^2$ matrix with rank r and the non-pivots occur in the last $n^2 - r$ columns, then the rows R_i of the pseudo-inverse P, with $n^2 - r < i \leq n^2$ will form a basis for the quiet patterns.

Proof:

Based on the general form of A' (refer to the 2×2 representation from the beginning of this section) and assuming that Conjecture 3.1 holds true, the last $n^2 - r$ rows of P will be orthogonal to the columns of A (Since $PA = A'$). The matrix A is symmetric, so the last $n^2 - r$ rows of P are also orthogonal to the rows of A. Since P is invertible these last $n^2 - r$ rows of P form an independent set. Therefore, the last $n^2 - r$ rows of P give a basis of the null space of A_n which is equivalent to the basis of the set of quiet patterns of A_n.

▮

In order to see how this works, consider the G_5 example again.

Example 3.8 Another way to determine the basis of the null space of G_5 is by using Proposition 3.1. The rank of G_5 is 23, therefore the last two rows of A' are zero. This means that rows 24 and 25 form quiet patterns. Here is a sub-matrix of P found in figure 2.3 of the last two rows.

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

Each row represents a quiet pattern. These quiet patterns, rewritten in their matrix form are as follows and are identical to the quiet patterns found previously.

$$R_{24} = q_1 = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix} \qquad R_{25} = q_2 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Coincidentally, this produces the exact same basis as the other methods did. This will not always be the case however, but both methods will produce a valid basis of the null space. □

The 5×5 case is the most widely studied game board, since it is the size of the most popular toy that was distributed to play the game on. An interesting example to look at that has not been studied in as much detail is the 4×4 game board.

Example 3.9 The game board G_4 has rank 12, meaning there will be 4 quiet patterns that define the basis of the null space.

The last four rows of a pseudo inverse are as follows, and due to Proposition 3.1 form a basis of the quiet patterns for G_4.

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

Written in their matrix form, they are:

$$q_1 = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \qquad q_2 = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$q_3 = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad q_4 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

There are $15 = 2^4 - 1$ non trivial quiet patterns for G_4 overall which can all be generated by adding linear combinations of these four quiet patterns.

Other bases for the null space of the G_4 game board will be explored in more detail using properties of recursion and the $\hat{A}_4$ matrix discussed in Chapter 5.

□

It is now possible to determine exactly how many quiet patterns each game board will have. As mentioned earlier, this will explain exactly how many different ways the solution to a single solveable configuration can be found.

Let d be the nullity of A (meaning the dimension of the null space which equals $n^2 - r$). There are only two possible coefficients (0 or 1) for each element of the basis, and there will be d elements in the basis since the last d rows correspond to quiet patterns. Then the total number of vectors in the null space of A will be,

$$|null(A)| = 2^d$$

This means that for any solvable configuration, there are 2^d different game moves that will solve the game. For example, the 5×5 game has dimension 25 and rank 23, so it's nullity is 2. There are $2^2 = 4$ quiet patterns, and therefore 4 different ways to solve any solveable configuration.

So far, all of the methods of finding quiet patterns depend on information from either A' or P. There is yet another way to create quiet patterns without using matrices.

3.5 Method for Generating Quiet Patterns

After looking at several examples of quiet patterns, it becomes fairly obvious that they have a few properties in common. These similarities provide an alternative way to build quiet patterns rather than finding them in the reduced row echelon form of the adjacenty matrix. These properties are largely based on the idea that pressing a light an even number of times will return the light to its original state.

Since a quiet pattern returns all the lights to their original state, the following results will always hold true for individual vertices on a game board.

Theorem 3.10 If a vertex is not pressed in a given quiet pattern, then 0, 2 or 4 of its connected vertices must be pressed to ensure a quiet pattern. If a vertex is pressed in a given quiet pattern, then 1 or 3 of its connected vertices must be pressed to ensure a quiet pattern.

Proof:

Let v be a vertex of game board G_n and let q be a quiet pattern of G_n.

The state of v after q is pressed depends only on its connected vertices. Since v is on an $n \times n$ board, it will have 2, 3 or 4 possible connecting vertices depending on it's placement on the board (a corner, an edge or somewhere in the center of the board). Pressing a neighboring vertex of v will result in a change of state for v by the rules of the game.

First, assume v is not pressed. For v to be unchanged after pressing all of q, an even number of connected vertices to v must be pressed. Since there are at most 4 vertices connected to v, the only possibilities are 0, 2 or 4 connecting vertices pressed to leave the state of v unchanged.

Next, assume v is pressed. Since v is pressed, the state of v will change. In order to return v to its original state, an odd number of connected vertices must be pressed. Since v has at most 4 connecting vertices, either 1 or 3 connecting vertices must be pressed to leave the state of v unchanged. ∎

This theorem creates a method for generating quiet patterns. If any vertex is designated to be pressed in a quiet pattern, then there are only a limited number of possibilities of what to do with the neighboring vertices in order to create a quiet pattern following the rules outlined in the theorem.

Example 3.11 In order to build a quiet pattern of G_4, start by placing a 1 in the upper left corner of the 4×4 matrix, representing that the first vertex will be pressed in this quiet pattern.

$$q = \begin{bmatrix} 1 & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}$$

Then since there is a one in this position, there can only be 1 or 3 other ones connected to it. Since three is not possible in the corner position, choose the following (the other choice will also create a greedy quiet pattern):

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

Again, examine the adjacent vertices to the entries that were just put in starting at the upper right and working down to the lower left.

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

This process can be continued until the lower right corner is reached and checking each adjacent vertex at each step.

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

This is one example of a quiet pattern for the G_4 game board and is some linear combination of the basis that was found in Example 3.9

□

This process of generating quiet patterns can be followed for any size game board, and gives a procedure to find quiet patterns without needing information from the pseudo inverse. Some game boards do not have quiet patterns however, so it will be impossible to fill in the blank quiet pattern matrix with anything other than all zeros and still follows the rules in the theorem.

3.6 Symmetry and Rotations

All the G_n game boards with quiet patterns share an important property. Since they are square in size, the orientation of the game board does not play a significant role. The square shape of the game board in the game of Lights Out can be connected to the dihedral group of order 8 which describes all the rigid motions of a square.

This means that starting with any quiet pattern and applying any rigid motion of the square will produce a quiet pattern. Rigid motions are rotations of $0°, 90°, 180°$, and $270°$ or reflections across the horizontal, vertical or either diagonal axis of symmetry.

Once a quiet pattern is found, either by generating it or finding the basis of the null space, new quiet patterns can be found by performing symmetries and rotations on that original quiet pattern. Also, adding two quiet patterns together will still leave the state of the lights unchanged overall, giving yet another way to generate new quiet patterns.

Example 3.12 Using the quiet pattern for G_4 found in example 3.11, new quiet patterns can be found.

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

First rotate 90° to produce, $q = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$

Notice that any other rotation will yield one of these two possibilities. Also, this quiet pattern is symmetrical across the vertical, horizontal and diagonal axis so the reflections do not produce any new quiet patterns for this specific example.

Adding these two together will produce another quiet pattern for G_4.

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

This shows that from one quiet pattern, it is possible to produce several new quiet patterns using rotations, symmetries and addition.

□

The rotations and reflections of the square game board can be applied to more than just quiet patterns. If an initial configuration is given, with a known game move, then any other configuration of the initial configuration can be solved with the same motion applied to the game move.

3.7 Even Number of Ones

Quiet patterns share an interesting similarity regarding the number of button presses needed. This is due to the fact that quiet patterns leave the state of the lights unchanged so there are certain restrictions about how they can be structured, as described in the previous section.

Theorem 3.13 Every quiet pattern will require an even number of button presses.

Proof: Let v be a vector in $\mathbb{Z}_2^{n^2}$. Then consider $v^T Av$ where A is the adjacency matrix of the game board G_n. Since A is a symmetric $n^2 \times n^2$ matrix with entries a_{ij}, it has an associated quadratic form in n variables given by,

$$q(x_1, x_2, ..., x_n) = v^T Av = \sum_{i,j=1}^{n^2} a_{ij} x_i x_j$$

Since A is symmetric, $a_{ij} = a_{ji}$, so $a_{ij} x_i x_j + a_{ji} x_i x_j = 0$ because the same term added twice will produce 0. This means there is no longer a need for two indicies because all terms not on the diagonal will cancel each other out. Now the sum can be rewritten as follows.

$$q(x_1, x_2, ..., x_n) = \sum_{i=1}^{n^2} a_{ii} x_i x_i$$

Also, along the diagonal of A all entries will be 1, so $a_{ii} = 1$ for all i. Also $x_i x_i = x_i$ in $\mathbb{Z}_2$ since $0 \cdot 0 = 0$ and $1 \cdot 1 = 1$. Therefore the associated quadratic form of A can be simplified to,

$$q_A(x_1, ..., x_n) = v^T Av = \sum_{i=1}^{n^2} x_i^2 = v^T v = \sum_{i=1}^{n^2} x_i$$

This summation, before it is reduced modulo 2, is called the weight of the vector v represented as $wt(v)$. It represents the number of ones in the vector.

A game move vector v is a quiet pattern if and only if $Av = 0$. Multiply by v^T on the left and this implies that $v^TAv = 0$ will only happen when the weight of v, $wt(v)$ is even, making the summation equal to zero when taken modulo 2.

$$wt(v) = v^TAv = \sum_{i=1}^{n^2} x_i = 0 \Rightarrow \text{ Even number of } x_i = 1$$

∎

Another way to see why all quiet patterns have even weight is to view the interaction of a button press and it's neighbors. If a vertex is pressed in a quiet pattern, then an odd number of it's neighbors (either one or 3) must also be pressed in order to keep the state of that vertex unchanged.

Definition 3.14 A *handshake of vertex* v, is defined when a vertex and one of it's neighbors, connected either vertically or horizontally, are pressed in a quiet pattern. Then $hs(v)$ is the number of handshakes for vertex v.

Definition 3.15 The number of *mutual handshakes of a quiet pattern* q, $mhs(q)$ is defined to be the total number of handshakes exchanged for a quiet pattern.

$$mhs(q) = \frac{\displaystyle\sum_{v \in q} hs(v)}{2}$$

If v_1 and v_2 have a handshake, then v_2 and v_1 have a handshake as well so two individual handshakes make one mutual handshake.

Since each v will have an odd number of handshakes in order to leave it's state unchanged, the sum in the numerator will be the sum of odd numbers. It is impossible to have a fractional amount of handshakes on any game board so there has to be an integer amount of them.

$$\frac{\sum \text{Odd numbers}}{2} = \# \text{ of mutual handshakes (integer value)}$$

This means that the summation of the handshakes must be an even number in order to guarantee that the numerator is divisible by 2. Adding an odd number of odd's will be odd, which is not divisible by 2. Therefore this means the summation in the numerator must be an even number of odd's added together. This means there must be an even number of vertices in any quiet pattern.

Example 3.16 Given the following quiet pattern in G_4, we can examine the number of handshakes and mutual handshakes.

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

Vertex $\in q$	# of Handshakes
v_2	1
v_3	3
v_4	1
v_5	1
v_7	1
v_9	3
v_{10}	1
v_{13}	1

Then the number of mutual handshakes will be,

$$mhs(q) = \frac{\displaystyle\sum_{v \in q} hs(v)}{2} = \frac{1+3+1+1+1+3+1+1}{2} = \frac{12}{2} = 6$$

So this quiet pattern has a total of 6 mutual handshakes. □

One application of the property that quiet patterns have an even number of ones is to solve the All Ones Problem. Instead of the goal being to turn all of the lights out, the game board now starts completely off and the goal is to turn all the lights on. Define a new matrix J_n, and $n \times n$ matrix where each entry is one. If the game board starts with all lights off, then the equation $AX = J$ describes the game move X needed in order to solve the All Ones Problem.

The J_n matrix of all ones corresponds to the all 1 vector in $\mathbb{Z}_2^{n^2}$. Taking the dot product of J as a vector and q as a vector,

$$J \cdot q = (1,1,1,1...1) \cdot (q_1, q_2, ..., q_n) = q_1 + q_2 + ... + q_n = 0$$

The dot product equals 0 because of the theorem above, that any quiet pattern will have an even number of entries equal to one, therefore making their sum 0 when taken modulo 2.

This means that $J \in null(A))^{\perp}$, therefore by Theorem 3.3, $AX = J$ will have a solution. Therefore any game board with quiet patterns will have a solution to the all ones problem.

Chapter 4

Light Chasing

There is an alternative way of solving the game of Lights Out that relies on an algorithm that can be applied to any game board for a given n. Light chasing is the most practical way to solve Lights Out without needing to do large calculations by hand or on a computer. Its algorithm is simple enough to memorize once it is developed, and will always work even though it is not necessarily the fastest way to solve the game.

4.1 General Algorithm

The process of light chasing applies to any game board with specific rules that vary depending on the dimension. This general algorithm is what will turn all the lights out for any initial configuration that is solveable.

Algorithm 4.1 Light Chasing (Any Size Game Board)

1. Given a solvable configuration of G_n, express the configuration of the game board in matrix form. Label the rows $r_1, r_2, ..., r_n$.

2. Starting with $i = 1$, press the r_i pattern in the r_{i+1} row, for $1 \leq i \leq n-1$.

More generally, press the vertices directly below the lights that are on in the row above, starting with the top row and repeating the process with the next row until the bottom row is reached.

3. Depending on the configuration of the bottom row, refer to the step 3 rule for the particular n size game board and press the corresponding combination of vertices in r_1. These rules can be found in section 4.3 and 4.4.

4. Repeat step 2, chasing the lights to the bottom row, which will solve the game. ∎

Using this algorithm is an intuitive way to solve the game and can provide a faster way to determine if an initial configuration is solvable. In order to understand the process of light chasing, here is an example of the first 2 steps for a particular initial configuration.

Example 4.1 Suppose the initial configuration for the G_5 game board is as follows,

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

The following matrices represent the light chasing process applied to each row of this matrix. For each new configuration, the vertices that were pressed are in bold and the state of each light in the game is represented as a 0 or 1.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ \mathbf{1} & 0 & \mathbf{1} & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \mathbf{0} & 0 & \mathbf{1} & 0 & \mathbf{0} \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & \mathbf{1} & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} \end{bmatrix}$$

That completes step 2 of the light chasing algorithm and shows the bottom row configuration for this C_s. In order to further solve the game, the step 3 rules for G_5, developed later in this chapter, would need to be referenced.

$\square$

4.2 Solvability Regarding Light Chasing

The step 3 rule of the algorithm varies for each size n game board, and depends entirely on what the bottom row configuration is after the lights are chased the first time through. Before getting into the specific rules, a few observations must be made about these bottom row configurations.

As stated before, when a game board has an invertible matrix, this means that all possible initial configurations have a solution. In terms of light chasing, this means that all possible configurations of the bottom row will be solvable. The last row will have n vertices, each with 2 possible states, therefore there will be 2^n possible bottom row configurations, each with a corresponding step 3 rule for light chasing.

For example, the G_3 game board has an invertible adjacency matrix. This means that each initial configuration can be chased to one of the $2^3 = 8$ bottom row configurations and each bottom row configuration will have a step 3 rule that will solve the game.

When the game board does not have an invertible matrix, there will be some initial configurations that do not have solutions, and once chased, correspond to bottom row configurations that are not solveable. This reduces the number of step 3 rules needed for the game board, because there will be less bottom row configurations, and actually provides a test for checking solvability of an initial configuration.

Theorem 4.2 Light Chasing Solvability
Let c be a game board configuration of G_n where the lights have been chased to the bottom row. Let Q the matrix of quiet patterns for G_n where each column represents a quiet pattern in the basis. If $c \cdot Q = 0$ then c is a solveable bottom row configuration. Therefore any C_s that is transformed into c after light chasing will be a solvable configuration.

Proof: From previous work, a configuration C_s can be solved if and only if $C_s \cdot q = 0$ for all $q \in null(A)$. A direct application of this result is that if $c \cdot q = 0$ for each $q \in null(A)$ then, $c \cdot Q = 0$ will hold true. Since the process of light chasing can be reversed to the initial configuration that c came from, C_s will be solvable if c is solvable.

∎

The configuration c in vector form will be all zeros except for some combination of zeros and ones in the last n entries since the only lights on will be in the bottom row. Due to this set up of the bottom row configurations, the only entries of c that could be non-zero are the last n entries, it is sufficient to simplify this dot product to the last n entries of both c and each q.

This means that the last n entries of the basis of the quiet pattern help determine if an initial configuration is solvable after it is chased to the bottom row. Define a subvector of c called $\bar{c}$ with entries $n^2 - n < i \leq n^2$, and a submatrix of Q, called $\bar{Q}$ with only the i rows of each column. If $\bar{c} \cdot \bar{Q} = 0$ then c is a solvable bottom row configuration.

If Proposition 3.1 holds true for any size n where the basis for the quiet patterns can be found in P, then these last n entries of the basis of quiet patterns can be taken directly from P.

Example 4.3 In G_5, the last five entries of the two quiet patterns can be found in P as the last 5 entries of the bottom two rows of P. Turning these rows into columns for Q and then creating the submatrix of the last n entries of the quiet patterns gives,

$$\bar{Q} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}$$

If an initial configuration is chased to the following bottom row configuration,

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

So taking the last 5 entries and writing the subvector, $\bar{c} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \end{bmatrix}$

In order to check if this bottom row configuration is solvable, then $\bar{c} \cdot \bar{Q}$ would have to equal zero when taken modulo 2.

$$\bar{c} \cdot \bar{Q} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 3 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

The result was not a vector of all zeros, therefore any initial configuration becomes c after light chasing will not be a solvable configuration. No game move exists that will solve that configuration.

□

While it is convenient to be able to check if a particular c configuration will have a solution, it is also possible to produce the set of all bottom row configurations that will have solutions. This can be done by finding the solution set to the system $\bar{x} \cdot \bar{Q} = 0$, or equivalently when the bottom row vector is written as a column vector, and $\bar{Q}^T \cdot \bar{x} = 0$.

Example 4.4 In order to find all possible bottom row configurations that will be solvable, find all $\bar{x}$ satisfying $\bar{x} \cdot \bar{Q} = 0$. Since systems traditionally are set up in the form $AX = 0$, use the transpose of $\bar{Q}$ which represents the same quiet patterns, just listed as columns instead of rows.

$$\bar{Q}^T = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

This can be parametrized to find a generic solution for this matrix multiplied by some solution equal to zero. If the last five vertices are labeled a, b, c, d, e, then for parameters $r = c, s = d$, and $t = e$,

$$r\begin{bmatrix}1\\1\\1\\0\\0\end{bmatrix}+s\begin{bmatrix}0\\1\\0\\1\\0\end{bmatrix}+t\begin{bmatrix}1\\0\\0\\0\\1\end{bmatrix}\text{ for } r,s,t\in\{0,1\}$$

This results in a total of $2^3 = 8$ possible solutions to the system. This means there are eight bottom row configurations that will be solvable. That reduces the $2^5 = 32$ possible bottom row configurations down to $2^3 = 8$ solvable bottom row configurations.

The solvable bottom row configurations for G_5, listed as row vectors are,

$$\begin{array}{ll}
c_1 = \begin{bmatrix}1 & 1 & 1 & 0 & 0\end{bmatrix} & r = 1,\ s,t = 0\\
c_2 = \begin{bmatrix}0 & 1 & 0 & 1 & 0\end{bmatrix} & s = 1,\ r,t = 0\\
c_3 = \begin{bmatrix}1 & 0 & 0 & 0 & 1\end{bmatrix} & t = 1,\ r,s = 0\\
c_4 = \begin{bmatrix}1 & 0 & 1 & 1 & 0\end{bmatrix} & r,s = 1,\ t = 0\\
c_5 = \begin{bmatrix}0 & 1 & 1 & 0 & 1\end{bmatrix} & r,t = 1,\ s = 0\\
c_6 = \begin{bmatrix}1 & 1 & 0 & 1 & 1\end{bmatrix} & t,s = 1,\ r = 0\\
c_7 = \begin{bmatrix}0 & 0 & 1 & 1 & 1\end{bmatrix} & r,s,t = 1\\
c_8 = \begin{bmatrix}0 & 0 & 0 & 0 & 0\end{bmatrix} & r,s,t = 0
\end{array}$$

□

This process can be done for any game board that has quiet patterns and will help reduce the number of possible bottom row configurations. The reason why finding all possible bottom row configurations is important is because the bottom row configurations are what determine the rules for step 3 of the light chasing algorithm. Narrowing down the bottom row configurations to only the solvable ones makes the algorithm cleaner and easier to memorize. This list of vectors can also be used as a check to determine if a configuration is solvable, instead of having to multiply PC_s and seeing if that game move solves the game.

4.3 Individual Game Board Rules

There are several ways to determine the step three rules for light chasing on any game board. Once all the solvable bottom row configurations are found, the correct first row presses can be calculated by working backwards.

Start with a game board that is entirely off. Press some combination of top row buttons, then chase to the bottom row. If the lights that are left on in the bottom row match a solvable bottom row configuration, then that top row button press pattern will solve that configuration.

Example 4.5 Starting with all the lights off on a 5×5 game board, press the top row configuration,

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

Then after pressing v_1 and v_2, the configuration of the board is,

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

Then chase the lights to the bottom row,

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

This matches the bottom row configuration c_3 from example 4.4. This means that if the bottom row configuration only has v_{21} and v_{25} on, then pressing v_1 and v_2 in the top row, and chasing will turn all of the lights on the game board off.

□

On game boards such as G_5 that have non-invertible adjacency matrices, there will be multiple top row button press patterns that work to solve the game for a given bottom row configuration. This is directly linked with the idea of quiet patterns, and how any solvable starting configuration will have multiple game moves that can solve it if there are quiet patterns.

This process of matching up bottom row configurations and top row button presses can be somewhat tedious since it requires a guess and check method, especially as the size of the game board increases. Instead of blindly testing all the solvable bottom row configurations with top row button presses, a more intentional approach can be used. The theory supporting this approach is developed in detail in Section 5.5 and Section 5.7 which rely on concepts that are derived eariler in that chapter.

Chapter 5

Structure Based on Game Board Dimensions

Predicting the structure and properties for games of any size is a difficult task. In order to get a glimpse into things such as the structure of A' and the number of quiet patterns for any size n, tools like polynomial representations and recursion can begin to answer some of these difficult questions.

5.1 The Sylvester Equation Representation

While the effect of a button press can be represented by a single matrix or vector, it is beneficial to consider what happens when broken down into several steps. For example, with the classic game of lights out, a button press represents a + pattern, so if 1's represent the press of a light, the following matrix represents a button press at any localized area of any game board. By matrix addition it can be broken down into anywhere from 2 to 5 steps. The most beneficial breakdown to study will be the two step addition example. A basic example is pressing the central button on G_3, but this can be extended to any press on any size game board.

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

Previously, the adjacency matrix A was used to describe the effect of connections on a game board. So given any game move X, AX represents the new configuration of the board after the game move is pressed. This idea can be extended by breaking up the moves made into horizontal connections and vertical connections.

Define two matrices, $\hat{A}_n$ and $\hat{B}_n$ that will help recreate the multiplication of the adjacency process in two steps.

The matrix $\hat{A}_n$ is the line graph adjacency matrix as defined in chapter 2 and will be a matrix of size $n \times n$ with 1's only in the three center diagonals. Then the multiplication of $X\hat{A}_n$ will represents the vertical connections including the button press itself.

$$\hat{A}_n = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 1 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 1 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 & 1 & 1 \\ 0 & \cdots & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

The matrix $\hat{B}_n$ will be the matrix describing the horizontal connections, not including the button press itself, and will consist of 1's only in the diagonals above and below the center diagonal. The multiplication of $\hat{B}_n X$ will therefore represent the connections directly above and below the vertex pressed.

$$\hat{B}_n = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 & 0 & 1 \\ 0 & \cdots & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

When a game move is represented as an n^2 length vector, $X \mapsto AX$ describes the effect of button presses on the game board. Consider what occurs when X is represented as an $n \times n$ matrix instead of it's typical n^2 length vector. This requires a different mapping, where $X \mapsto \hat{B}_n X + X\hat{A}_n$. Here is an example to illustrate this new mapping.

Example 5.1 Let X be the following game move on the board G_4.

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

Then to determine the effect this button press will have on the game board, calculate $\hat{B}_4 X + X\hat{A}_4$

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

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

So pressing the button sequence in X will change the state of all the lights except for v_1, v_2, v_3 and v_6.

This is exactly the same outcome that is produced by multiplying AX where X is in vector form instead of matrix form. □

Having the game move vector in matrix form makes it easier to visualize what is going on with the game board. This new mapping provides a way to express the equation $AX = Y$ where X was in vector form, into $\hat{B}_n X + X\hat{A}_n = Y$ where X is in matrix form. Quiet patterns can be found by solving the equation $AX = 0$ for X, so similarly, quiet patterns can be discovered by finding all X such that $\hat{B}_n X + X\hat{A}_n = 0$.

This matrix equation is in a form that is called the Sylvester Equation, which in general form is $MX + XN = 0$ where X, N, M are $n \times n$ matrices. This equation has applications to other areas of mathematics and engineering specifically problems regarding dynamical systems.

There is a sufficient condition describing when a solution to the Sylvester Equation will exist, which implies the existence of a quiet pattern in terms of the game. It is commonly referred to as the Eigenvalue Condition.

Theorem 5.2 The Eigenvalue Condition

If λ is an eigenvalue of M with eigenvector v such that $-\lambda$ is an eigenvalue of N^T with eigenvector w then $X = vw^T$ is a solution to the equation $MX + XN = 0$ and therefore a quiet pattern.

Proof: A quick check will establish this fact. If $X = vw^T$, then

$$\begin{aligned}
&MX + XN \\
&= Mvw^T + vw^T N && \text{since } X = vw^T \\
&= Mvw^T + (N^T wv^T)^T && \text{since } L^{TT} = L \text{ and } K^T L^T = (LK)^T \text{ for any matrices } L, K \\
&= \lambda vw^T + (-\lambda wv^T)^T && \text{since } Mv = \lambda v \text{ and } N^T w = -\lambda w \\
&= \lambda vw^T - \lambda vw^T && \text{since } K^T L^T = (LK)^T \\
&= 0
\end{aligned}$$

∎

This means there is a new way of describing quiet patterns, as the product of two vectors vw^T. An example at work can be found later in this chapter in example 5.10. The dimension of vw^T will be $n \times n$ since v is an n length column vector and w^T is an n length row vector. This matrix multiplication, vw^T is actually called the outer product of two vectors. While the inner product produces a scalar from two vectors, the outer product produces a matrix from two vectors. This method is limited however, because it doesn't yield all possible quiet patterns.

In terms of the game, this means that quiet patterns can be created as outer products of the eigenvectors of $\hat{A}_n$ and $\hat{B}_n$. In order to see if there is any predictable structure to these eigenvectors, a few mathematical tools can be used.

5.2 Use of the Characteristic Polynomial and Eigenvectors

As seen in the previous section, the eigenvectors v and w play an important role in determining game moves and specifically quiet patterns for a game board. The associated polynomials of the matrices that these eigenvectors come from can help anticipate the properties of these important eigenvectors.

The two matrices defined in the previous section, $\hat{A}_n$ and $\hat{B}_n$ are useful in finding solutions to the game of Lights Out. They are very similar and can be related by the simple fact that $\hat{A}_n = I + \hat{B}_n$. This is because $\hat{B}_n$ is exactly the adjacency matrix of the line graph without the assumption that each vertex is connected to itself, as it is in $\hat{A}_n$. Some interesting properties of the these two matrices arise after examining them in more detail. In order to study these matrices in a more general sense, consider their associated characteristic polynomials.

Definition 5.3 The *characteristic polynomial* of a $n \times n$ matrix M is the polynomial defined by $p_n(\lambda) = det(\lambda I - M)$ where I is the $n \times n$ identity matrix.

First, examine the characteristic polynomial of $\hat{B}_n$. Working through a few steps of the cofactor expansion process used to calculate the determinant shows that there is another way to define the characteristic polynomial of $\hat{B}_n$ recursively. In order to illustrate this process, consider the G_5 example.

Example 5.4 Consider the game board G_5. Then,

$$\hat{B}_5 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Then the characteristic polynomial requires computing the determinant of the matrix $\lambda I - \hat{B}_n$. Note the form that this matrix takes for any n is λ as elements on the diagonal, and -1 as entries in the upper and lower diagonal. Also notice, a matrix is defined with brackets $[M]$ and the notation for the determinant of the matrix will be defined with bars, $|M| = det(M)$.

$$p_5(\lambda) = \begin{vmatrix} \lambda & -1 & 0 & 0 & 0 \\ -1 & \lambda & -1 & 0 & 0 \\ 0 & -1 & \lambda & -1 & 0 \\ 0 & 0 & -1 & \lambda & -1 \\ 0 & 0 & 0 & -1 & \lambda \end{vmatrix}$$

Using cofactor expansion on the bottom row of the matrix produces the first step in expanding this polynomial.

$$= -(-1) \begin{vmatrix} \lambda & -1 & 0 & 0 \\ -1 & \lambda & -1 & 0 \\ 0 & -1 & \lambda & 0 \\ 0 & 0 & -1 & -1 \end{vmatrix} + \lambda \begin{vmatrix} \lambda & -1 & 0 & 0 \\ -1 & \lambda & -1 & 0 \\ 0 & -1 & \lambda & -1 \\ 0 & 0 & -1 & \lambda \end{vmatrix}$$

$$= -(-1)\begin{vmatrix} \lambda & -1 & 0 & 0 \\ -1 & \lambda & -1 & 0 \\ 0 & -1 & \lambda & 0 \\ 0 & 0 & -1 & -1 \end{vmatrix} +\lambda\ p_4(\lambda)$$

Again, using cofactor expansion now on the right column of the matrix, produces the following result.

$$= (-1)\begin{vmatrix} \lambda & -1 & 0 \\ -1 & \lambda & -1 \\ 0 & -1 & \lambda \end{vmatrix} +\lambda\ p_4(\lambda)$$

$$= (-1)\ p_3(\lambda) + \lambda\ p_4(\lambda)$$

Picking this order of cofactor expansion has produced a recursive way to express they characteristic polynomial of $\hat{B}_5$.

$p_5(\lambda) = \lambda\ p_4(\lambda) - p_3(\lambda)$

□

This same idea can be extended for any n larger than 2 because of the way the $\hat{B}_n$ is constructed. Cofactor expansion of the bottom row, and then the right column will always produce the following result,

$$p_n(\lambda) = \lambda\ p_{n-1}(\lambda) - p_{n-2}(\lambda) \text{ for } n > 2 \tag{5.1}$$

Equation 5.1, taken modulo 2, holds a significant pattern that is a reoccurring concept when studying the game of Lights Out.

$$p_n(\lambda) = \lambda p_{n-1}(\lambda) + p_{n-2}(\lambda)$$

Expand the first few polynomials, all reduced modulo 2.

$$p_1 = det(\lambda I - \hat{B}_1) = det(\lambda I + \hat{B}_1) = |\lambda| = \lambda$$

$$p_2 = \begin{vmatrix} \lambda & 1 \\ 1 & \lambda \end{vmatrix} = \lambda^2 - 1 = \lambda^2 + 1$$

$$\begin{array}{llll} p_3 & = \lambda p_2 + p_1 & = \lambda(\lambda^2+1) + \lambda & = \lambda^3 \\ p_4 & = \lambda p_3 + p_2 & = \lambda(\lambda^3) + \lambda^2 + 1 & = \lambda^4 + \lambda^2 + 1 \\ p_5 & = \lambda p_4 + p_3 & = \lambda(\lambda^4 + \lambda^2 + 1) + \lambda^3 & = \lambda^5 + \lambda \\ p_6 & = \lambda p_5 + p_4 & = \lambda(\lambda^5 + \lambda) + \lambda^4 + \lambda^2 + 1 & = \lambda^6 + \lambda^4 + 1 \\ \vdots & & & \end{array}$$

This sequence can continue for any n, and will be studied in more detail throughout this chapter as it arises in several situations.

Using this equation of recursively defining the characteristic polynomial, an important relationship between the determinants of $\hat{B}_n$ can be explained. When $\lambda = 0$, then $p_n(0) = det(-\hat{B}_n)$ which is equivalent to $(-1)^n det(\hat{B}_n)$. This produces a pattern for the determinants that can be described recursively.

$$\begin{array}{lcl} p_n(0) & = & 0\ p_{n-1}(0) - p_{n-2}(0) \\ det(-\hat{B}_n) & = & 0 - det(-\hat{B}_{n-2}) \\ (-1)^n det(\hat{B}_n) & = & -(-1)^{n-2} det(\hat{B}_{n-2}) \\ (-1)^n det(\hat{B}_n) & = & (-1)^{n-1} det(\hat{B}_{n-2}) \\ det(\hat{B}_n) & = & -det(\hat{B}_{n-2}) \end{array}$$

Therefore, the determinant of $\hat{B}_n$ can be defined recursively as

$$det(\hat{B}_n) \quad = \quad -det(\hat{B}_{n-2}) \tag{5.2}$$

Now in order to link $\hat{B}_n$ with $\hat{A}_n$, consider what happens when $\lambda = -1$. To get a feel for this, first look at an example.

Example 5.5 For the game G_4,

$$p_4(-1) = det(-1I - \hat{B}_4) = \begin{vmatrix} -1 & -1 & 0 & 0 \\ -1 & -1 & -1 & 0 \\ 0 & -1 & -1 & -1 \\ 0 & 0 & -1 & -1 \end{vmatrix} = det(-\hat{A}_n) = (-1)^n det(\hat{A}_n)$$

□

This is true for any n because of how $\hat{B}_n$ and $\hat{A}_n$ are defined, therefore,

$$p_n(-1) = (-1)^n det(\hat{A}_n)$$

Using this observation and equation 5.1,

$$\begin{aligned} p_n(-1) &= (-1)p_{n-1}(-1) - p_{n-2}(-1) \\ (-1)^n det(\hat{A}_n) &= (-1)(-1)^{n-1} det(\hat{A}_{n-1}) - (-1)^{n-2} det(\hat{A}_{n-2}) \\ (-1)^n det(\hat{A}_n) &= (-1)^n det(\hat{A}_{n-1}) - (-1)^n (-1)^{-2} det(\hat{A}_{n-2}) \\ det(\hat{A}_n) &= det(\hat{A}_{n-1}) - det(\hat{A}_{n-2}) \end{aligned}$$

Therefore, the determinants of $\hat{A}_n$ have the following recursive property.

$$det(\hat{A}_n) = det(\hat{A}_{n-1}) - det(\hat{A}_{n-2}) \text{ for } n > 2 \qquad (5.3)$$

This equation is similar to a familiar recursive formula. If you increase the indices and label $det(\hat{A}_n) = d_n$, then equation 5.3 is the same as,

$$d_{n+1} = d_n - d_{n-1}$$

When taken modulo 2, this equation is the Fibonacci recursion formula $a_{n+1} = a_n + a_{n-1}$.

Since these recursive formulas are defined only for $n > 2$, it is necessary to find these initial values when $n = 1, 2$. An important generalization can be made about the eigenvalues of each matrix by looking at the recursive properties of their determinants.

For Equation 5.2, $det(\hat{B}_n) = -det(\hat{B}_{n-2})$,

When $n = 1$, $det(\hat{B}_1) = \ | \ 0 \ | \ = \ 0$.

When $n = 2$, $det(\hat{B}_2) = \begin{vmatrix} 0 & 1 \\ 1 & 0 \end{vmatrix} = -1$

Using these as the initial values, this means, $det(\hat{B}_n) = \begin{cases} 0 & \text{for } n \text{ odd} \\ 1 & \text{for } n \text{ even} \end{cases}$

For Equation 5.3, $det(\hat{A}_n) = det(\hat{A}_{n-1}) - det(\hat{A}_{n-2})$,

When $n = 1$, $det(\hat{A}_1) = |1| = 1$.

When $n = 2$, $det(\hat{A}_2) = \begin{vmatrix} 1 & 1 \\ 1 & 1 \end{vmatrix} = 0$

The type of recursion in equation 5.3 with the given initial values can be solved by examining its corresponding characteristic quadratic. Moving the terms of equation 5.3 produces,

$$d_{n+1} - d_n + d_{n-2} = 0$$

Therefore equation 5.3 can be associated with the quadratic,

$$q(x) = x^2 - x + 1$$

Looking at the pattern of d_n values helps to determine what d_0 will equal. Given initial values of $d_1 = 1$ and $d_2 = 0$, the next several values can be calculated using equation 5.3.

d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}
1	0	$0-1$ -1	$-1-0$ -1	$-1-(-1)$ 0	$0-(-1)$ 1	$1-0$ 1	$1-1$ 0	$0-1$ -1	$-1-0$ -1

The pattern of $1, 0, -1, -1, 0, 1$ emerges which means that this sequence is periodic and will repeat itself every 6 values. This means, that going backward a term, the initial term d_0 is forced to equal 1 in order to follow the pattern.

The underlying reason as to why the roots are periodic is because the roots of $q(x) = x^2 - x + 1$ which are $e^{\frac{\pi i}{3}}$ and $e^{-\frac{\pi i}{3}}$, are the primitive 6th roots of unity. The nth primitive roots of unity are all the n roots of unity that are not also a kth root of unity for some smaller k. Primitive roots of unity have a nice property of periodicity. If z is an nth root of unity then the sequence of powers $..., z^{-1}, z^0, z^1, ...$ is n periodic because $z^{i+n} = z^i \cdot z^n = z^i \cdot 1 = z^i$. This means that the associated quadratic to the recursive polynomial has roots with period 6.

Since the computations in this game are done modulo 2, the values of the determinant of each matrix can be reduced modulo 2. This means the sequence, $1, 0, -1, -1, 0, 1$ actually equals $1, 0, 1, 1, 1, 0, 1$ which now has period three, repeating $1, 1, 0$.

This means $det(\hat{A}_n) = \begin{cases} 0 & n \equiv -1 \bmod 3 \\ 1 & n \equiv 0, 1 \bmod 3 \end{cases}$

Now that the pattern for the determinants of both matrices is known, consider again equation 5.1,

$$p_n(\lambda) = \lambda \; p_{n-1}(\lambda) - p_{n-2}(\lambda)$$

When computations are done modulo 2, $det(\lambda I - \hat{B}_n) = det(\lambda I + \hat{B}_n)$, so the equation can be rewritten as

$$p_n(\lambda) = \lambda \; p_{n-1}(\lambda) + p_{n-2}(\lambda)$$

Eigenvalues must satisfy the equation $(\lambda I - A)v = 0$ for nonzero vectors v by definition. For any matrix, $Mv = 0$ requires $det(M) = 0$ since v is non-zero. This means that eigenvalues must satisfy the property $det(\lambda I - A) = 0$. So in order to find the eigenvalues of these matrices, the important determinants to examine are those n values that would make the determinant equal zero. This leads to the following two theorems,

Theorem 5.6 Eigenvalue of $\hat{B}_n$
When n is odd, $det(\hat{B}_n) = 0$ and hence 0 is an eigenvalue of $\hat{B}_n$.

Theorem 5.7 Eigenvalue of $\hat{A}_n$
When $n = -1 \bmod 3$, $det(\hat{A}_n) = 0$ and hence 0 is an eigenvalue of $\hat{A}_n$.

Since both $\hat{A}_n$ and $\hat{B}_n$ are involved in a solving a particular game board configuration, it is most important to look at what values of n will satisfy both of these requirements for eigenvalues. So if n is odd and $n \equiv -1 \bmod 3$ then 0 is an eigenvalue for both $\hat{A}_n$ and $\hat{B}_n$. This leads to an additional finding that eigenvalues of these matrices can also be related in the following way.

Theorem 5.8 Relationship Between Eigenvalues of $\hat{A}_n$ and $\hat{B}_n$
λ is an eigenvalue of $\hat{B}_n$ if and only if $\lambda + 1$ is an eigenvalue of $\hat{A}_n$.

Proof: The two matrices are related by the fact that modulo 2,

$$\hat{B}_n = \hat{A}_n - I = I + \hat{A}_n$$

This means that,

$$\begin{aligned} p_n(\lambda) &= det(\lambda I - \hat{B}_n) \\ &= det(\lambda I + \hat{B}_n) \\ &= det(\lambda I + I + \hat{A}_n) \\ &= det((\lambda + 1)I + \hat{A}_n) \end{aligned}$$

Eigenvalues of $\hat{B}_n$ are all λ such that $p_n(\lambda) = 0$ which is equivalent to $det((\lambda + 1)I + \hat{A}_n) = 0$. Therefore $\lambda + 1$ will be an eigenvalue of $\hat{A}_n$

∎

So combining all these theorems we gives the following results that will hold true for the n described.

Theorem 5.9 Eigenvalues of $\hat{A}_n$ and $\hat{B}_n$
0 is an eigenvalue for $\hat{B}_n \Leftrightarrow$ 1 is an eigenvalue for $\hat{A}_n \Leftrightarrow n$ is odd, also
0 is an eigenvalue for $\hat{A}_n \Leftrightarrow$ 1 is an eigenvalue for $\hat{B}_n \Leftrightarrow n \equiv -1 \bmod 3$

This important generalization can now be used to produce quiet patterns for n satisfying the restrictions in the theorem.

5.3 General Representation of Quiet Patterns

As discussed before, quiet patterns are solutions to the Sylvester equation and some satisfy the eigenvalue condition. This means that the solutions to the equation $\hat{B}_n X + X\hat{A}_n = 0$ are all quiet patterns, and if λ is an eigenvalue of $\hat{B}_n$ with eigenvector v and taken modulo 2, $-\lambda = \lambda$ is an eigenvalue of $\hat{A}_n$ with eigenvector w, then $X = vw^T$ is a quiet pattern.

Using this knowledge, another procedure can be defined to produce quiet patterns of a game board. First it needs to be determined which values of n will produce eigenvalues that are the same for $\hat{A}_n$ and $\hat{B}_n$.

According to Theorem 5.9, this will occur when both the conditions for n are satisfied, so when n is both odd and $n \equiv -1 \bmod 3$.

$$\begin{array}{lll} n \equiv -1 \bmod 3 & \Rightarrow & n = 3k - 1 \text{ for } k \in \mathbb{Z} \\ n \text{ odd} & \Rightarrow & k \text{ is even, so } k = 2l \text{ for } l \in \mathbb{Z} \\ k = 2l & \Rightarrow & n = 6l - 1 \end{array}$$

Therefore, in order for n to satisfy both of the conditions in Theorem 5.9,

$$n \equiv -1 \bmod 6 \tag{5.4}$$

So game boards of size n satisfying this condition will have a quiet patterns that can be found by calculating their eigenvectors v and w and multiplying vw^T.

Example 5.10 Let $n = 5$, because $5 \equiv -1 \bmod 6$. According to Theorem 5.9, then $\lambda = 0, 1$. Eigenvectors of any matrix A are all nonzero x vectors that satisfy the equation $Ax = \lambda x$.

Then solving $\hat{B}_5 v = 0v = 0$ requires row reducing $\hat{B}_n$.

$$\hat{B}_5 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \backsim \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \text{ So, } v = t \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

Then the only solutions to $\hat{B}_5 v = 0$ are vectors parametrized by $t = v_5$ and of the form above. In terms of the game, the only eigenvector that is necessary to look at is when $t = 1$. Any other value of t taken modulo 2 will either produce the trivial quiet pattern, or the same one when $t = 1$.

An identical process can be followed for $\hat{A}_5$. This time the solutions for $\hat{A}_5 w = 0$ will be parametrized by $t = w_5$ which again is most useful to look at the case $t = 1$.

$$\hat{A}_5 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \backsim \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \text{So, } w = t \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Then according to the eigenvalue condition, the multiplication of vw^T produces a quiet pattern.

$$vw^T = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} [1 \quad 1 \quad 0 \quad 1 \quad 1] = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

□

This is not the only quiet pattern that can be produced this way. Since $\hat{A}_n = \hat{B}_n + I$, it can be shown that the roles of $\hat{A}_n$ and $\hat{B}_n$ can be switched.

$$\begin{aligned} & \hat{B}_n X + X\hat{A}_n = 0 \\ \Leftrightarrow \quad & \hat{B}_n X + X(\hat{B}_n + I) = 0 \\ \Leftrightarrow \quad & \hat{B}_n X + X + X\hat{B}_n = 0 \\ \Leftrightarrow \quad & (\hat{B}_n + I)X + X\hat{B}_n = 0 \\ \Leftrightarrow \quad & \hat{A}_n X + X\hat{B}_n = 0 \end{aligned}$$

This means that when Sylvester's equation is taken modulo 2 and when n satisfies the condition in equation 5.4, another quiet pattern can be produced by multiplying wv^T.

Example 5.11 Using the v and w calculated in the previous example, another quiet pattern can be produced.

$$wv^T = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} [1 \quad 0 \quad 1 \quad 0 \quad 1] = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

□

In the previous two examples, the eigenvectors v and w were calculated by using the eigenvalue $\lambda = 0$ in the equation $Av = \lambda v$. It is clear from Theorem 5.9 that whenever the condition in equation 5.4 is satisfied, both 0 and 1 will be eigenvalues for $\hat{A}_n$ and $\hat{B}_n$. Using the shared eigenvalue of 1 will not produce any new quiet patterns. This is because if $Av = 1v$, $Bw = 1w$ then $Bv = 0v$, $Aw = 0w$ which produces nothing new.

After $n = 5$, the next n value this works for is $n = 11$. While these matrices get large quickly as n increases, there is a way to find a pattern that makes this computation easy to do.

The equations $\hat{A}_n w = 0$ and $\hat{B}_n v = 0$ can be solved in general for any n satisfying Equation 5.4. First consider what happens as each row of $\hat{A}_n$ is multiplied by w where the entries of w are labeled as x_i.

$$\hat{A}_n w = 0 \begin{bmatrix} 1 & 1 & 0 & 0 & \cdots & & \\ 1 & 1 & 1 & 0 & \cdots & & \\ 0 & 1 & 1 & 1 & 0 & \cdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots \\ & & & & & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ \end{bmatrix}$$

Label the ith row of $\hat{A}_n$ as Ri. Then the multiplication done row by row can be simplified modulo 2 using the results from each previous row. Here are the first several rows, each line corresponding to the entries of w expressed with entries x_i.

$$\begin{array}{llll}
R1 \cdot w = 0 & \Rightarrow x_1 + x_2 = 0 & \Rightarrow x_1 = x_2 & \text{so let } x_1 = x_2 = a \\
R2 \cdot w = 0 & \Rightarrow x_1 + x_2 + x_3 = 0 & \Rightarrow 0 + x_3 = 0 & \Rightarrow x_3 = 0 \\
R3 \cdot w = 0 & \Rightarrow x_2 + x_3 + x_4 = 0 & \Rightarrow a + 0 + x_4 = 0 & \Rightarrow x_4 = a \\
R4 \cdot w = 0 & \Rightarrow x_3 + x_4 + x_5 = 0 & \Rightarrow 0 + a + x_5 = 0 & \Rightarrow x_5 = a \\
R5 \cdot w = 0 & \Rightarrow x_4 + x_5 + x_6 = 0 & \Rightarrow a + a + x_6 = 0 & \Rightarrow x_6 = 0 \\
R6 \cdot w = 0 & \Rightarrow x_5 + x_5 + x_7 = 0 & \Rightarrow a + 0 + x_7 = 0 & \Rightarrow x_7 = a \\
R7 \cdot w = 0 & \Rightarrow x_6 + x_7 + x_8 = 0 & \Rightarrow 0 + a + x_8 = 0 & \Rightarrow x_8 = a \\
\vdots & & &
\end{array}$$

A pattern begins to emerge between the entries of w. Starting at the first entry of w and working down, the sequence $a\ a\ 0$ repeats.

In order to figure out what the end of w looks like, consider the last row of $\hat{A}_n$, written as $Rn = [0 \quad \cdots \quad 0 \quad 1 \quad 1]$. The structure of this row of A forces the last two entries to equal each other, therefore they must be a. This means $x_{n-1} = x_n = a$. Therefore, w will have the following form, following the pattern and ending in $a\ a$.

$$w = \begin{bmatrix} a \\ a \\ 0 \\ \vdots \\ a \\ a \\ 0 \\ a \\ a \end{bmatrix}$$

This brings up an interesting connection to what was seen earlier. The length of this eigenvector is the pattern of size three repeated k times, and then two additional entries, a a. This is $3k+2 \equiv -1 \bmod 3$.

This same process can be followed for $\hat{B}_n$ to find a pattern for the eigenvector v.

$$\hat{B}_n v = 0 \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 & 0 & 1 \\ 0 & \cdots & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Following the same idea of multiplying each row yields a new pattern.

$$\begin{array}{lllll}
R1 \cdot v = 0 & \Rightarrow x_2 = 0 & & & \\
R2 \cdot v = 0 & \Rightarrow x_1 + x_3 = 0 & \Rightarrow x_1 = x_3 & & \text{so let } x_1 = x_3 = a \\
R3 \cdot v = 0 & \Rightarrow x_2 + x_4 = 0 & \Rightarrow 0 + x_4 = 0 & \Rightarrow x_4 = 0 & \\
R4 \cdot v = 0 & \Rightarrow x_3 + x_5 = 0 & \Rightarrow a + x_5 = 0 & \Rightarrow x_5 = a & \\
R5 \cdot v = 0 & \Rightarrow x_4 + x_6 = 0 & \Rightarrow 0 + x_6 = 0 & \Rightarrow x_6 = 0 & \\
R6 \cdot v = 0 & \Rightarrow x_5 + x_7 = 0 & \Rightarrow a + x_7 = 0 & \Rightarrow x_7 = a & \\
R7 \cdot v = 0 & \Rightarrow x_6 + x_8 = 0 & \Rightarrow 0 + x_8 = 0 & \Rightarrow x_8 = 0 & \\
\vdots & & & &
\end{array}$$

The pattern for v is beginning with a and alternating between a and 0.
Again, the last row of $\hat{A}_n$ will force the last entry of w. $Rn = [0 \quad \cdots \quad 0 \quad 1 \quad 0]$.

Therefore $x_{n-1} = 0$ so the last entry $x_n = a$. Therefore, v will have the following form, following the alternating pattern and ending in a, which consequently forces n to be odd.

$$v = \begin{bmatrix} a \\ 0 \\ a \\ 0 \\ a \\ \vdots \\ a \\ 0 \\ a \end{bmatrix}$$

Example 5.12 Now building a quiet pattern for the $n = 11$ can be done without having to row reduce an 11×11 matrices. Following the patterns described above, the eigenvectors will be,

$$v = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \qquad w = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Then a quiet pattern can be found by multiplying vw^T.

$$vw^T = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

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

Switching the roles of v and w will produce yet another quiet pattern. Then rotations, reflections, and addition can create even more quiet patterns for this game board.

□

This way to develop quiet patterns began with the fact that both $\hat{A}_n$ and $\hat{B}_n$ satisfy the eigenvalue condition in a specific case where $\lambda = 0, 1$ are shared eigenvalues of both matrices. In order to utilize the eigenvalue condition to its full extent, all λ with λ and $\lambda + 1$ eigenvalues of $\hat{B}_n$ would need to be determined.

The eigenvalues of $\hat{B}_n$ will generally lie in some field extension of $\mathbb{Z}_2$. However, even when these eigenvalues are found, the solutions vw^T will have entries other than 0 and 1 giving this outer product no application to the game whatsoever. It would need to be determined when λ's from a field extension of $\mathbb{Z}_2$ will have only entries of 0 and 1 to make this finding relevant to the quiet pattern problem.

5.4 Quiet Pattern Null Space Correspondence

Recursion can be used to determine an important property of quiet patterns for a given n. If a general configuration C of the game board is expressed in matrix form, then label the vector from each row with as r.

$$C = \begin{bmatrix} \cdots & r_1 & \cdots \\ \cdots & r_2 & \cdots \\ \cdots & r_3 & \cdots \\ & \vdots & \\ \cdots & r_n & \cdots \end{bmatrix}$$

Then to create the n^2 length column vector used in the equation $A_nC = 0$ for configurations C, make each v a column vector, and express C as,

$$C = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_n \end{bmatrix}$$

Quiet patterns will satisfy the equation $A_nC = 0$, so multiplying gives,

$$A_nC = \begin{bmatrix} \hat{A}_n & I & 0 & \dots & 0 \\ I & \hat{A}_n & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \hat{A}_n & I \\ 0 & \dots & 0 & I & \hat{A}_n \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

This multiplication produces a predictable pattern based on the way A_n is constructed.

$$\begin{bmatrix} r_1\hat{A} + r_2 \\ r_1 + r_2\hat{A} + r_3 \\ \vdots \\ r_{i-1} + x_i\hat{A} + r_{i+1} \\ \vdots \\ r_{n-1} + r_n\hat{A} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

This produces a system of equations that can be solved inductively, starting with the first entry and working down to the nth entry.

$$\begin{array}{lll}
r_2 = r_1\hat{A} & & \\
r_3 = r_1 + r_2\hat{A} & \Rightarrow r_1 + (r_1\hat{A})\hat{A} & \Rightarrow r_1(I + \hat{A}^2) \\
r_4 = r_2 + r_3\hat{A} & \Rightarrow r_1\hat{A} + r_1(I + \hat{A}^2)\hat{A} & \Rightarrow r_1\hat{A}^3 \\
r_5 = r_3 + r_4\hat{A} & \Rightarrow r_1(I + \hat{A}^2) + (r_1\hat{A}^3)\hat{A} & \Rightarrow r_1(I + \hat{A}^2 + \hat{A}^4) \\
r_6 = r_4 + r_5\hat{A} & \Rightarrow r_1\hat{A}^3 + (r_1(I + \hat{A}^2 + \hat{A}^4))\hat{A} & \Rightarrow (\hat{A} + r_1\hat{A}^5) \\
r_7 = r_5 + r_6\hat{A} & \Rightarrow r_1(I + \hat{A}^2 + \hat{A}^4) + r_1(\hat{A} + \hat{A}^5)\hat{A} & \Rightarrow r_1(1 + \hat{A}^4 + \hat{A}^6) \\
\vdots & &
\end{array}$$

So every entry of this vector that represents a general quiet pattern for a size n game board can be expressed in terms of $\hat{A}$ and r_1. This sequence of coefficents of r_1 are familiar from the study of the characteristic polynomials of $\hat{B}_n$.

Due to the fact that this sequence reoccurs several times in this exploration, give this sequence the notation, $\{b\}$. To simplify notation, let 1 represent the $n \times n$ identity matrix, and $a = \hat{A}$. Then the first ten terms will be ,

$$\begin{aligned}
b_0 &= 1\\
b_1 &= a\\
b_2 &= 1 + a^2\\
b_3 &= a^3\\
b_4 &= 1 + a^2 + a^4\\
b_5 &= a + a^5\\
b_6 &= 1 + a^4 + a^6\\
b_7 &= a^7\\
b_8 &= 1 + a^4 + a^6 + a^8\\
b_9 &= a + a^5 + a^9\\
b_{10} &= 1 + a^2 + a^4 + a^8 + a^{10}\\
&\vdots
\end{aligned}$$

Now that $\{b\}$ represents the coefficients of r_1 in each entry, a general quiet pattern can be expressed in the form,

$$\begin{bmatrix} r_1 b_0 \\ r_1 b_1 \\ r_1 b_2 \\ \vdots \\ r_1 b_{n-1} \end{bmatrix}$$

Now that typical form of quiet patterns is known, the only unknown is r_1. In order to determine what restrictions are necessary on r_1 to make sure this vector is a quiet pattern, consider the last entry of the general quiet pattern. The last entry (row) of the general quiet pattern corresponds to the equation,

$r_{n-1} + r_n\hat{A} = 0$ which would mean that $r_1 b_{n-2} + r_1 a b_{n-1} = 0$. Since the

sequence b_i has the form $b_{k+1} = ab_k + b_{k-1}$, this means that $r_1 b_n = 0$. So r_1 must be a vector in the null space of the matrix b_n.
Therefore $null(A_n) = \{\begin{bmatrix} rb_0 & rb_1 & \cdots & rb_{n-1} \end{bmatrix}^T \mid rb_n = 0\}$

This same inductive process can be started at the end instead of the beginning. This means that instead of rewriting every entry of the vector in terms of r_1, they can be written in terms of r_n. This will be helpful when developing the theory behind light chasing.

$$\begin{aligned}
&r_{n-1} + r_n\hat{A} = 0 && \Rightarrow r_{n-1} = r_n\hat{A} \\
&r_{n-2} + r_{n-1}\hat{A} + r_n = 0 && \Rightarrow r_{n-2} = (r_n\hat{A})\hat{A} + r_n \quad \Rightarrow r_{n-2} = r_n(I + \hat{A}^2) \\
&\vdots \\
&r_1\hat{A} + r_2 = 0 && \Rightarrow r_n\hat{A}b_{n-1} + r_nb_{n-2} \qquad \Rightarrow r_nb_n = 0
\end{aligned}$$

Therefore, an alternative way to represent $null(A_n)$ is,

$$null(A_n) = \{\begin{bmatrix} rb_{n-1} & rb_{n-2} & \cdots & rb_1 & r \end{bmatrix}^T \mid rb_n = 0\}$$

This means that the null space of A_n, which is the basis of all quiet patterns for the game, is completely determined by the null space of b_n, the $n \times n$ matrix corresponding to the nth term of $\{b\}$. This means that for any game, the number of quiet patterns can be found by finding the size of the null space of b_n.

This concept of determining the size of the null space and quiet patterns is directly tied with both the row reduction process of the adjacency matrix and the theory behind the light chasing method.

5.5 Light Chasing General Theory

The sequence of b_i shows up in the process of solving games using the method of light chasing. Consider any starting configuration of a game board. To demonstrate, G_5 will be used, but this can easily be extended to any size.

A starting configuration can be represented as an $n \times n$ matrix, or a n^2 length vector, where the rows of C_s in matrix form, correspond to the n entries in the vector form.

$$C_s = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} \leftarrow r_1 \\ \leftarrow r_2 \\ \leftarrow r_3 \\ \leftarrow r_4 \\ \leftarrow r_5 \end{matrix}$$

$C_s = [r_1\ r_2\ r_3\ r_4\ r_5]^T$

Using the method of light chasing, the first step would be to press r_1 in the second row of the game board. The effect this has on the lights in the game board is found by the following multiplication,

$$\begin{bmatrix} \hat{A} & I & 0 & 0 & 0 \\ I & \hat{A} & I & 0 & 0 \\ 0 & I & \hat{A} & I & 0 \\ 0 & 0 & I & \hat{A} & I \\ 0 & 0 & 0 & I & \hat{A} \end{bmatrix} \begin{bmatrix} 0 \\ r_1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} r_1 \\ r_1\hat{A} \\ r_1 \\ 0 \\ 0 \end{bmatrix}$$

This means that the second row will now be $r_2 + \hat{A}r_1$, since initially r_2 was the configuration, and $\hat{A}r_1$ is the effect on the board after the fist step of light chasing. This is what needs to be pressed in row 3 so step two of light chasing will then be the following,

$$\begin{bmatrix} \hat{A} & I & 0 & 0 & 0 \\ I & \hat{A} & I & 0 & 0 \\ 0 & I & \hat{A} & I & 0 \\ 0 & 0 & I & \hat{A} & I \\ 0 & 0 & 0 & I & \hat{A} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ r_2 + \hat{A}r_1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ r_2 + r_1\hat{A} \\ r_2\hat{A} + r_1\hat{A}^2 \\ r_2 + r_1\hat{A} \\ 0 \end{bmatrix}$$

Now that the first and second rows of lights are off, the third row will be $r_3 + r_2\hat{A} + r_1\hat{A}^2$. Continuing the method of light chasing, the effect that the third row of light chasing will have on the game board is,

$$\begin{bmatrix} \hat{A} & I & 0 & 0 & 0 \\ I & \hat{A} & I & 0 & 0 \\ 0 & I & \hat{A} & I & 0 \\ 0 & 0 & I & \hat{A} & I \\ 0 & 0 & 0 & I & \hat{A} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ r_3 + r_2\hat{A} + r_1\hat{A}^2 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ r_3 + r_2\hat{A} + r_1\hat{A}^2 \\ r_3\hat{A} + r_2\hat{A}^2 + r_1(\hat{A} + \hat{A}^3) \\ r_3 + r_2\hat{A} + r_2\hat{A}^2 \end{bmatrix}$$

This means that now the fourth row of the game board will be $r_4 + r_3\hat{A} + r_2\hat{A}^2 + r_1(\hat{A} + \hat{A}^3)$, and the next step of the light chasing method will be to press this combination in the bottom row of G_5.

$$\begin{bmatrix} \hat{A} & I & 0 & 0 & 0 \\ I & \hat{A} & I & 0 & 0 \\ 0 & I & \hat{A} & I & 0 \\ 0 & 0 & I & \hat{A} & I \\ 0 & 0 & 0 & I & \hat{A} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ r_4 + r_3\hat{A} + r_2\hat{A}^2 + r_1(\hat{A} + \hat{A}^3) \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 0 \\ 0 \\ r_4 + r_3\hat{A} + r_2\hat{A}^2 + r_1(\hat{A} + \hat{A}^3) \\ r_4\hat{A} + r_3\hat{A}^2 + r_2(\hat{A} + \hat{A}^3) + r_1\hat{A}^4 \end{bmatrix}$$

Now the lights have been chased to the bottom row, so the final configuration of the game board will be C_s all the effects after each step of the chase.

$$\begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \end{bmatrix} + \begin{bmatrix} r_1 \\ r_1\hat{A} \\ r_1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ r_2 + r_1\hat{A} \\ r_2\hat{A} + r_1\hat{A}^2 \\ r_2 + r_1\hat{A} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ r_3 + r_2\hat{A} + v_1\hat{A}^2 \\ r_3\hat{A} + r_2\hat{A}^2 + r_1(\hat{A} + \hat{A}^3) \\ r_3 + r_2\hat{A} + r_2\hat{A}^2 \end{bmatrix}$$

$$+ \begin{bmatrix} 0 \\ 0 \\ 0 \\ r_4 + r_3\hat{A} + r + 2\hat{A}^2 + r_1(\hat{A} + \hat{A}^3) \\ r_4\hat{A} + r_3\hat{A}^2 + r_2(\hat{A} + \hat{A}^3) + r_1\hat{A}^4 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ r_5 + r_4\hat{A} + r_3\hat{A}^2 + r_2\hat{A}^3 + r_1(\hat{A}^2 + \hat{A}^4) \end{bmatrix}$$

This bottom row configuration can be expressed equivalently as,

$$r_5 + r_4\hat{A} + r_3(I + \hat{A}^2) + r_2\hat{A}^3 + r_1(I + \hat{A}^2 + \hat{A}^4)$$

This can be written using $\{b\}$,

$$r_5 + r_4b_1 + r_3b_2 + r_2b_3 + r_1b_4$$

More generally, if an initial configuration of a game board is

$$C_s = [r_1\ r_2\ \cdots\ r_n]^T$$

Then after chasing the lights, the bottom row configuration, denoted as c, will be

$c = [0\ \cdots\ 0\ \circledast]^T$ where $\circledast = r_nb_0 + r_{n-1}b_1 + r_{n-2}b_2 + \cdots + r_1b_{n-1}$

Any initial configuration C_s can be turned off if and only if the bottom row configuration c can be turned off. This is true since the button presses that were used to transform C_s into c by chasing can be reversed. Then the following is known about any configuration C_s, using concepts that were developed in Theorem 3.3.

A configuration C_s can be turned off

$\Leftrightarrow C_s \in Col(A) = Row(A) = null(A)^\perp$

$\Leftrightarrow c \in null(A)^\perp$

$\Leftrightarrow c \cdot q = 0$ for all $q \in null(A)$

In Section 5.4, it was shown that,

$$null(A_n) = \{\begin{bmatrix} rb_{n-1} & rb_{n-2} & \cdots & rb_1 & r \end{bmatrix}^T \mid rb_n = 0\}.$$

So, $q \in null(A)$ if and only if $q = \begin{bmatrix} rb_{n-1} & rb_{n-2} & \cdots & rb_1 & r \end{bmatrix}^T$ for some $r \in null(b_n)$.

Now, since c has all zero entries except for the last entry, $\circledast$, the dot product $c \cdot q$ will equal $\circledast \cdot r$ for all $r \in null(b_n)$.

This means the first step of light chasing will be to chase the lights to the bottom row, and then verify that $\circledast \cdot r = 0$ for all $r \in null(b_n)$, otherwise it is not possible to turn the lights off. So, $\circledast \cdot r = 0$ for all $r \in null(b_n)$ if and only if $\circledast \in (null(b_n))^{\perp} = Row(b_n) = Col(b_n)$ since b_n is symmetric.

Once it is verified that a solution will exist, it is now necessary to find a combination of vertices to press in the top row that once chased will result in the same $\circledast$ bottom row configuration, call it z which would turn all the lights out. Since $\circledast \in Row(b_n)$, this row vector z will exist and $zb_n = \circledast$.

In general, the idea is to start with an initial configuration labeled with rows r_i, chase them to the bottom leaving a bottom row configuration of $\circledast$, then press a combinations of lights z that once pressed and chased, will also produce $\circledast$ as a bottom row configuration, therefore changing the state of all the lights that were on in $\circledast$ and turning all lights out.

$$\begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_{n-1} \\ r_n \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \circledast \end{bmatrix} \rightarrow \begin{bmatrix} zA \\ z \\ 0 \\ \vdots \\ 0 \\ \circledast \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \circledast + \circledast \end{bmatrix} \rightarrow \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

It is now necessary to find this row vector z in $zb_n = \circledast$, which is an $n \times n$ system. Since b_n is symmetric, this is the same as solving $b_n z = \circledast$ with z represented as a column vector instead of a row vector. Since this notation is more common in solving systems, assume from this point on that z is a column vector even though it represents the button presses needed in the first row of the matrix. When b_n is invertible, $z = (b_n)^{-1}\circledast$ and any initial configuration will be solvable.

When b_n is not invertible solving this system can be done using the idea of a pseudo inverse of b_n. Previously, a pseudo inverse P was defined for an adjacency matrix A where $A' = PA$ for A' being the reduced row echelon form of A. Extend this definition to b_n now by labeling the pseudo inverse of b_n as R such that $b'_n = Rb_n$.

Finding a solution to the system $b_n z = \circledast$ then involves creating the augmented matrix $[\ b_n \mid \circledast\ ]$ and using row operations to reduce the matrix to $[\ b'_n \mid R\circledast\ ]$.

Theorem 5.13 The system, $b_n z = \circledast$ has a solution if and only if $\circledast \cdot R_j = 0$ for $n - r < j \leq n$.

Proof: Since b'_n is in reduced row echelon from, $b'_n = RA$ will have the last $n - r$ rows as entirely zeros where r is the rank of b_n. This means that the last $n - r$ rows of R will be in $Col(b_n)^{\perp} = Row(b_n)^{\perp}$. The rows of R are independent and $Row(b_n)^{\perp}$ will have a basis of size $n - r$. This means the last $n - r$ rows of R will be a basis of $Row(b_n)^{\perp} = null(b_n)$. Then

$b_n z = \circledast$ has a solution

$\Leftrightarrow \circledast \in Col(b_n) = Row(b_n) = null(b_n)^{\perp}$

$\Leftrightarrow \circledast \cdot Rj = 0$ for Rj the jth row of R for $n - r < j \leq n$

∎

Introducing parameters, the solution can be obtained in the same way shown in Section 3.3.

A special case of this is when b'_n has the last $n - r$ columns as its only pivot columns. This is exactly what was conjectured for A. The parameter positions will then be the last $n - r$ entries therefore making the vector $R\circledast$ a solution to the system.

Now the next step is to describe the structure of R, a pseudo inverse of b_n. This can be done with a closer examination of the reduced row echelon form of A.

5.6 Reduced Row Echelon Form of A

Since all the adjacency matrices of these game boards have a similar structure, some computations can be done that will hold for any size game board. When given the task to determine what the reduced row echelon form of A will be, this general process can be followed for any size n. As shown before, the adjacency matrix A can be row reduced into the following form through matrix multiplication.

$$A_n = \left[\begin{array}{cccc|c} \hat{A}_n & \hat{I}_n & \hat{0}_n & \dots & \hat{0}_n \\ \hline \hat{I}_n & \hat{A}_n & \ddots & \ddots & \vdots \\ \hat{0}_n & \ddots & \ddots & \ddots & \hat{0}_n \\ \vdots & \ddots & \ddots & \hat{A}_n & \hat{I}_n \\ \hat{0}_n & \dots & \hat{0}_n & \hat{I}_n & \hat{A}_n \end{array}\right] = \left[\begin{array}{c|c} v & 0 \\ \hline M & w \end{array}\right]$$

The matrix M is upper triangular with ones on it's diagonal. This means that M is invertible, and therefore the initial row reduction of A_n can be expressed as,

$$\left[\begin{array}{c|c} 0 & M^{-1} \\ \hline I & -vM^{-1} \end{array}\right] \left[\begin{array}{c|c} v & 0 \\ \hline M & w \end{array}\right] = \left[\begin{array}{c|c} I & M^{-1}w \\ \hline 0 & -vM^{-1}w \end{array}\right]$$

So taken modulo 2, A_n can be row reduced. For the rest of this section, the A will be used to symbolize A_n in order to simplify notation.

$$A = \left[\begin{array}{c|c} I & M^{-1}w \\ \hline 0 & vM^{-1}w \end{array}\right]$$

The key to making predictions about any size n game board lies in anticipating what $vM^{-1}w$ will look like for any size n. This will help determine if the last $n^2 - r$ columns always provide information about the quiet patterns, and even what sizes of n will have quiet patterns.

First, consider the matrix M^{-1}. As said before, M is invertible and actually contains entries that are the b_i sequence as developed previously. To see this, again for simplicity, let $a = \hat{A}$ and $1 = I$.

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

Hence, M^{-1} will be the following matrix due to the recursion formula for b in the sequence b_i.

$$M^{-1} = \begin{bmatrix} 1 & b_1 & b_2 & b_3 & b_4 & \cdots & b_{n-2} \\ 0 & 1 & b_1 & b_2 & b_3 & \cdots & b_{n-3} \\ 0 & 0 & 1 & b_1 & b_2 & \cdots & b_{n-4} \\ \vdots & & & & & & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 & b_1 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix}$$

Because of the structure of $\{b\}$, the multiplication of $vM^{-1}w$ can actually be calculated. Keeping in mind that $b_{k+1} = ab_k + b_{k-1}$.

$vM^{-1}w$=

$$v \begin{bmatrix} 1 & b_1 & b_2 & b_3 & b_4 & \cdots & b_{n-2} \\ 0 & 1 & b_1 & b_2 & b_3 & \cdots & b_{n-3} \\ 0 & 0 & 1 & b_1 & b_2 & \cdots & b_{n-4} \\ \vdots & & & & & & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 & b_1 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ a \end{bmatrix} = v \begin{bmatrix} b_{n-3} + ab_{n-2} \\ b_{n-4} + ab_{n-3} \\ b_{n-5} + ab_{n-4} \\ \vdots \\ 1 + ab_1 \\ a \end{bmatrix}$$

$$= v \begin{bmatrix} b_{n-1} \\ b_{n-2} \\ b_{n-3} \\ \vdots \\ b_2 \\ b_1 \end{bmatrix} = \begin{bmatrix} a & 1 & 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} b_{n-1} \\ b_{n-2} \\ b_{n-3} \\ \vdots \\ b_2 \\ b_1 \end{bmatrix}$$

$$= \left[\begin{array}{c} ab_{n-1} + b_{n-2} \end{array} \right] = \left[\begin{array}{c} b_n \end{array} \right]$$

Therefore $vM^{-1}w = b_n$. This means that the lower right corner of the adjacency matrix, will be the nth term of $\{b\}$ which is a sequence of $n \times n$ matrices.

$$A_n = \left[\begin{array}{c|c} I & M^{-1}w \\ \hline 0 & b_n \end{array} \right]$$

Also, notice that the $M^{-1}w$ entry of the A' was calculated above and will be the matrix with entries following the sequence $\{b\}$.

$$M^{-1}w = \left[\begin{array}{c} b_{n-1} \\ b_{n-2} \\ b_{n-3} \\ \vdots \\ b_2 \\ b_1 \end{array} \right]$$

Using this structure of A as it is put into row echelon form, it becomes clear that the largest number of 0 rows possible in the final reduced row echelon form of A is n, therefore proving that the nullity of $A \leq n$. This means there can not be more than n vectors in the basis of the quiet patterns for a game board, and all information about these quiet patterns is hidden in this lower $n \times n$ matrix.

Now, the Row Echelon form of A has been reduced to,

$$A_n = \left[\begin{array}{c|c} & b_{n-1} \\ I & \vdots \\ & b_1 \\ \hline 0 & b_n \end{array}\right]$$

Now, A can be put into reduced row echelon form in two main steps. First, the row reduce the lower right corner, b_n, and then second, the final n columns above b_n.

First, the lower right $n \times n$ corner of A can be put into reduced row echelon form using the pseudo inverse of b_n described in section 5.5. Then,

$$\left[\begin{array}{c|c} I & 0 \\ \hline 0 & R \end{array}\right] \left[\begin{array}{c|c} I & M^{-1}w \\ \hline 0 & b_n \end{array}\right] = \left[\begin{array}{c|c} I & M^{-1}w \\ \hline 0 & Rb_n \end{array}\right]$$

So in order to row reduce A even further, A must be multiplied by,

$$\left[\begin{array}{c|c} I & 0 \\ \hline 0 & R \end{array}\right]$$

Now that the bottom right corner is in reduced row echelon form, all that is left is to row reduce $M^{-1}w$. This can be done by using the pivot columns of Rb_n and eliminating the entries above the pivots. This process is essentially is a multiplication of elementary matrices. These elementary matrices will have the form,

$\left[\begin{array}{c|c} I & E_i \\ \hline 0 & I \end{array}\right]$ where E_i is a matrix with all zeros except for a single entry of 1.

Example 5.14 To demonstrate these elementary matrices, consider a 6×6

matrix. If there is a pivot position in the 5th row and 5th column, then to eliminate an entry in the third row, 5th column, the elementary matrix needed would be,

$$E_1 = \left[\begin{array}{cccc|cc} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

Multiplying E_1A produces the same result as adding row three and row five when taken modulo 2. $\square$

Several elementary matrices are needed in order to get A into reduced row echelon form. Multiplying several of these elementary matrices together, say j are needed, then produces,

$$\left[\begin{array}{c|c} I & E_j \\ \hline 0 & I \end{array}\right] \cdots \left[\begin{array}{c|c} I & E_3 \\ \hline 0 & I \end{array}\right] \left[\begin{array}{c|c} I & E_2 \\ \hline 0 & I \end{array}\right] \left[\begin{array}{c|c} I & E_1 \\ \hline 0 & I \end{array}\right] = \left[\begin{array}{c|c} I & E \\ \hline 0 & I \end{array}\right]$$

Then the final step of row reducing A is to multiply by this matrix that is the combination of all the elementary matrices. Therefore, in order to get A into reduced row echelon form, A must be multiplied by these two matrices,

$$\left[\begin{array}{c|c} I & E \\ \hline 0 & I \end{array}\right] \left[\begin{array}{c|c} I & 0 \\ \hline 0 & R \end{array}\right] = \left[\begin{array}{c|c} I & ER \\ \hline 0 & R \end{array}\right]$$

So, finally, A can be put into reduced row echelon form through the following multiplication,

$$A' = \left[\begin{array}{c|c} I & ER \\ \hline 0 & R \end{array}\right] \left[\begin{array}{c|c} 0 & M^{-1} \\ \hline I & vM^{-1} \end{array}\right] A$$

As shown earlier when row reducing the first column of A when expressed as a 2×2 matrix ,

$$\left[\begin{array}{c|c} 0 & M^{-1} \\ \hline I & vM^{-1} \end{array}\right] A = \left[\begin{array}{c|c} I & M^{-1}w \\ \hline 0 & b_n \end{array}\right]$$

Therefore A' can be expressed as a single matrix using this substitution,

$$A' = \left[\begin{array}{c|c} I & ER \\ \hline 0 & R \end{array}\right] \left[\begin{array}{c|c} I & M^{-1}w \\ \hline 0 & b_n \end{array}\right] = \left[\begin{array}{c|c} I & M^{-1}w + ERb_n \\ \hline 0 & Rb_n \end{array}\right]$$

The key piece here is that the lower right $n \times n$ block of A' will be b_n multiplied by a pseudo-inverse of b_n. This can be used to further understand the method of light chasing.

5.7 The Chase Matrix

In the previous section, it was shown that,

$$A' = \left[\begin{array}{c|c} I & ER \\ \hline 0 & R \end{array}\right] \left[\begin{array}{c|c} 0 & M^{-1} \\ \hline I & vM^{-1} \end{array}\right] A$$

From the definition of pseudo-inverse, $A' = PA$, so the result of multiplying the two matrices in the original equation together will produce P.

$$P = \left[\begin{array}{c|c} I & ER \\ \hline 0 & R \end{array}\right] \left[\begin{array}{c|c} 0 & M^{-1} \\ \hline I & vM^{-1} \end{array}\right] = \left[\begin{array}{c|c} ER & I + ERvM^{-1} \\ \hline R & RvM^{-1} \end{array}\right]$$

The lower left $n \times n$ corner of P will be R, a pseudo inverse of b_n.

This method shows one particular way to produce a pseudo-inverse of A, however there are several possible P such that $A' = PA$. This calls for the following theorem.

Theorem 5.15 The bottom left $n \times n$ sub-matrix of any pseudo-inverse of A_n will be a pseudo-inverse of b_n.

Proof: Suppose P is a and want to show that the lower left corner is always a pseudo inverse of b_n. Take a general P and divide it into a 2×2 matrix in the same way as before,

$$P = \left[\begin{array}{c|c} X & Y \\ \hline Q & Z \end{array}\right]$$

The goal now is to show that Q is a pseudo-inverse of b_n. This can be done using the single matrix representation of A' that was just developed and this general pseudo inverse P.

$$A' = \left[\begin{array}{c|c} I & M^{-1}w + ERb_n \\ \hline 0 & Rb_n \end{array}\right] = PA = \left[\begin{array}{c|c} X & Y \\ \hline Q & Z \end{array}\right] A \tag{5.5}$$

The A in this equation can be substituted using a few ideas that have been derived in this section. At the beginning of this section, A was first simplified using this multiplication,

$$\left[\begin{array}{c|c} 0 & M^{-1} \\ \hline I & vM^{-1} \end{array}\right] A = \left[\begin{array}{c|c} I & M^{-1}w \\ \hline 0 & b_n \end{array}\right]$$

Solving for A,

$$
\begin{aligned}
A &= \left[\begin{array}{c|c} 0 & M^{-1} \\ \hline I & vM^{-1} \end{array}\right]^{-1} \left[\begin{array}{c|c} I & M^{-1}w \\ \hline 0 & b_n \end{array}\right] \\
&= \left[\begin{array}{c|c} v & I \\ \hline M & 0 \end{array}\right] \left[\begin{array}{c|c} I & M^{-1}w \\ \hline 0 & b_n \end{array}\right]
\end{aligned}
$$

Using this way of expressing A and the general P matrix, equation 5.5 can now be written as,

$$
A' = \left[\begin{array}{c|c} I & M^{-1}w + ERb_n \\ \hline 0 & Rb_n \end{array}\right] = PA = \left[\begin{array}{c|c} X & Y \\ \hline Q & Z \end{array}\right] \left[\begin{array}{c|c} v & I \\ \hline M & 0 \end{array}\right] \left[\begin{array}{c|c} I & M^{-1}w \\ \hline 0 & b_n \end{array}\right]
$$

Simplifying this produces the equation,

$$
A' = \left[\begin{array}{c|c} I & M^{-1}w + ERb_n \\ \hline 0 & Rb_n \end{array}\right] = \left[\begin{array}{c|c} Xv + YM & X \\ \hline Qv + ZM & Q \end{array}\right] \left[\begin{array}{c|c} I & M^{-1} \\ \hline 0 & b_n \end{array}\right]
$$

Multiplying the two matrices on the right gives,

$$
A' = \left[\begin{array}{c|c} I & M^{-1}w + ERb_n \\ \hline 0 & Rb_n \end{array}\right] = \left[\begin{array}{c|c} Xv + YM & M^{-1}w(Xv + YM) + Xb_n \\ \hline Qv + ZM & M^{-1}w(Qv + ZM) + Qb_n \end{array}\right]
$$

Since these two matrices are equal, comparing the lower left corner of both matrices shows that $0 = Qv + ZM$. This can be used to simplify the lower right corner. Setting the entries in the lower right corner equal and using the fact that $0 = Qv + ZM$, $Rb_n = Qb_n$.

Since A is in reduced row echelon form, and $Rb_n = Qb_n$, Q is also a pseudo-inverse of b_n. Therefore, for any pseudo-inverse of A, the lower left $n \times n$ block of P will be a pseudo-inverse of b_n.

▮

This means that there are multiple ways to find pseudo-inverses of b_n which helps to determine the top row button presses needed to solve a game. Two methods of finding these pseudo-inverse's are illustrated in the next two examples, including how they are used to solve a specific bottom row configuration.

Example 5.16 Solving a game using light chasing for G_5 can be found by directly calculating a pseudo-inverse of the fifth term in $\{b\}$, which is $b_5 = \hat{A}+\hat{A}^5$.

$$b_5 = \hat{A} + \hat{A}^5 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}^5$$

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

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

Augmenting b_5 with the identity matrix and row reducing until b_5 is in reduced row echelon form gives the following matrix,

$$\left[\begin{array}{ccccc|ccccc} 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \rightarrow \left[\begin{array}{ccccc|ccccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right]$$

So a pseudo-inverse of b_5 is,

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

The null space for b_5 can be found by looking at the non-pivot columns of b_5' using parameters s and t.

$$s \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} + t \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} = sv_1 + tv_2$$

A basis for the row space of b_5 are the three non-zero rows of b_5', label them $r_1 = [\ 1\ 0\ 0\ 0\ 1\ ]$, $r_2 = [\ 0\ 1\ 0\ 1\ 0\ ]$ and $r_3 = [\ 0\ 0\ 1\ 1\ 1\ ]$. An example of one element in the row space of b_5 would be $0r_1 + 1r_2 + 1r_3 = [\ 0\ 1\ 1\ 0\ 1]$, call this c.

It was previously shown that for a bottom row configuration c, $b_n z = c$ has a solution of top row button presses z if and only if c is in the column space of b_n. This will be true if and only if $c \cdot v_1 = 0$ and $c \cdot v_2 = 0$. Checking this requirement for our chosen c shows this is true, and a solution z will exist.

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} = 0 \text{ and } \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} = 0$$

Then finding a solution to $b_n z = c$ for $c = [\ 0\ 1\ 1\ 0\ 1\ ]$ amounts to multiplying c by R. Since the non-pivot columns of b_5 are the last 2 columns of b_5, cR will be a solution to the system provided that the last 2 entries are zero.

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

Therefore, pressing the first vertex on the top row will solve this bottom row configuration. This solution can be double checked by verifying $b_5 z = c$ with the z that was calculated which will work.

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

$\square$

The second method of finding a pseudo-inverse of b_n is by taking the lower left $n \times n$ block of a pseudo-inverse of A and using this as the chase matrix.

Example 5.17 The lower left 5 block of the pseudo-inverse of A_5 is,

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

Let $c = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \end{bmatrix}$ be a bottom row configuration. This is the same c that was chosen in the previous example.

Then a top row button press solution can be found by multiplying this R taken from P with c. It was already determined that the size of the null space was two dimensional for G_5 therefore, this multiplication will produce a solution, provided that the last 2 entries of are zero, as shown in Theorem 4.2.

$$\left[\begin{array}{ccccc} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{array}\right] \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \left[\begin{array}{c} 1 \\ 0 \\ 0 \\ \hline 0 \\ 0 \end{array}\right]$$

Therefore this will be a solution, and pressing vertex one in the top row and chasing will solve this bottom row configuration.

□

Because of Theorem 5.13, the last d entries need to be zero in order for a configuration to be solvable. For that reason, a sub-matrix of a pseudo-inverse of b_n can be defined, called it the chase matrix, that can be used to find top row button presses for solvable configurations only.

Definition 5.18 The *chase matrix*, C_n, is a sub-matrix of R, a pseudo-inverse of b_n disregarding the last d rows where d is the dimension of the null-space of A_n. When b_n is invertible, $C = b_n^{-1}$. The chase matrix can only be used for solvable configurations of G_n.

Example 5.19 The chase matrix for G_5 corresponds to the sub-matrix of R using only the top three rows since G_5 has a 2 dimensional null space. Using the pseudo-inverse of b_5 from the previous example,

$$R = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix} \qquad C_5 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

□

This button presses needed in order to solve each of the eight solvable configurations for G_5 can now be found using this chase matrix as a starting point. Label each of the rows of C as Ci, then the following top row presses can be found.

Bottom Row Configuration	Calculations using C	Top Row Button Press
$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \end{bmatrix}$	$C1 + C2 + C3$	$\begin{bmatrix} 0 & 0 & 1 & 1 & 0 \end{bmatrix}$
$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 \end{bmatrix}$	$C2$	$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 \end{bmatrix}$
$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \end{bmatrix}$	$C1$	$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 \end{bmatrix}$
$\begin{bmatrix} 1 & 0 & 1 & 1 & 0 \end{bmatrix}$	$C1 + C3$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix}$
$\begin{bmatrix} 0 & 1 & 1 & 0 & 1 \end{bmatrix}$	$C2 + C3$	$\begin{bmatrix} 0 & 0 & 1 & 0 & 1 \end{bmatrix}$
$\begin{bmatrix} 1 & 1 & 0 & 1 & 1 \end{bmatrix}$	$C1 + C2$	$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \end{bmatrix}$
$\begin{bmatrix} 0 & 0 & 1 & 1 & 1 \end{bmatrix}$	$C3$	$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix}$
$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \end{bmatrix}$		solved

Now, instead of memorizing all eight possible bottom row configurations for G_5, only these three rows of C_5 are necessary to remember. As long as the game board has been chased to a solvable bottom row configuration, using the rows of the chase matrix seems to be an efficient way to determine which top row button presses will solve the configuration.

The following rules can be added to the algorithm of light chasing, specifically for the game board G_5:

Step 3 rules for G_5:

Light On	Press
21	4,5
22	3,4,5
23	4

$$C_5 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Finding the rules for step three for any size game board amount to a two step process. First it must be verified that c is a solvable bottom row configuration. This can be done by checking if $c \in Col(b_n) = Row(b_n) =$

$null(b_n)^{\perp}$. This can be done by multiplying $c \cdot r_i = 0$ for the last d rows of C where d is the dimension of the null-space of A. If this is true, then the second step is to find the top row button presses z by multiplying $C_5 c$.

This process works provided that the reduced row echelon form of b_n has all non-pivot columns in the last columns. This is equivalent to the conjecture made about the non-pivot columns of A being the last $n^2 - r$ columns since b_n is the lower right block of A. If there exists a b_n that does not have this form, then an extra step must be taken in order to obtain the top row button presses needed. First $C_n c$ must be computed only the first $n - d$ entries are used along with zeros inserted into the parametrized positions of $Rb_n = b'_n$.

When A_n is invertible, then so is b_n, so b_n will have a unique inverse. This also means that any configuration will be solvable, so in order to find the bottom top row button presses needed after a bottom row has been chased, all that must be done is $z = b_n^{-1} c$.

An interesting example to consider is the G_4 game board since it has nullity 4=n.

Example 5.20 Consider the G_4 game board. Finding the A'_4 matrix involves row reducing the bottom 4×4 square, which will be the fourth term in $\{b\}$.

$$b_4 = 1{+}a^2{+}a^4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}^2 + \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}^4$$

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

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

This means that the bottom right corner of A'_4 is all zeros. This means that $Row(b_n) = 0$ so the only solvable bottom row configuration is the zero vector. Therefore for any configuration of G_4 to be solved, all the lights must be turned off after the initial round of light chasing.

□

5.8 Several G_n Chasing Rules

The step three rules given in this section only apply to initial configurations that are solvable since they have been determined using the chase matrix.

Light chasing can be applied to any game board, but first examine a few examples of game boards that have invertible matrices. As defined before, the chase matrix will come from b_n^{-1}, and there will be exactly n rules to remember since these matrices are of full rank.

Example 5.21 An example of an invertible matrix is the G_3 game board.

Here are the step three rules for this game. The chase matrix C_3 comes from the first three columns and last three rows of A_3^{-1}.

Step 3 rules for G_3:

Light On	Press
7	1,2
8	1,2,3
9	2,3

$$C_3 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

□

Example 5.22 Here are several other examples of an invertible matrix game boards and their corresponding step 3 rules. Each of these three game boards have full rank, so their chase matrix is exactly R as found in the lower left corner of A^{-1}.

Step 3 rules for G_6:

Light On	Press
31	1,3
32	4
33	1, 5
34	2, 6
35	3
36	4,6

$$C_6 = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Step 3 rules for G_7:

Light On	Press
43	1,2
44	1,2,3
45	2,3,4
46	3,4,5
47	4,5,6
48	5,6,7
49	6,7

$$C_7 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Step 3 rules for G_8:

Light On	Press
57	3,5
58	2,6
59	1, 7
60	8
61	1
62	2,8
63	3, 7
64	4,6

$$C_8 = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

□

A couple more examples of non-invertible matrix light chasing rules may be helpful to examine. Here are the chase matrices and corresponding rules for two more games.

Example 5.23 The G_9 game has nullity 8, meaning there is only one row in the chase matrix. This means after chasing there is really only one option about what to do. From the chase matrix, it can be seen that if vertex 73 is on, then the rule is as follows.

Step 3 rules for G_9:

Light On	Press
73	1,3,5,7,9

$$C_9 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

The G_{11} game has nullity 6, so there are 5 rows in the chase matrix. Here are the corresponding rules for this game.

Step 3 rules for G_{11}:

Light On	Press
111	3,4,5,6,8,9,10,11
112	3,4,6,8,10,11
113	3,4,5,7,9,10,11
114	5,6,8,9
115	3,6,7,8,11

$$C_{11} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

□

A unique example is the G_4 game board. It's size is fairly small, however it has a unique property that requires some exploration.

These are only a limited number of examples but the same ideas can be used for any size matrix. As mentioned earlier, the method of light chasing is not the fastest way to solve a game, but it does provide an alternative that makes any solvable configuration simple to solve without needing to do many calculations.

5.9 Further Study of $\{b\}$

Now that a sequence has been developed that holds the information needed to anticipate quiet patterns and study light chasing, it would be nice to be able to generate the nth term of this sequence without needing the two previous terms.

The previously defined matricies M and M^{-1} can help produce a new way of defining $\{b\}$ since the entries of M^{-1} are all terms of the sequence b_n. For reference, here are the two matrices again.

$$M = \begin{bmatrix} 1 & a & 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & a & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & a & 0 & 0 & \cdots & 0 \\ \vdots & & & & & & & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & a \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \quad M^{-1} = \begin{bmatrix} 1 & b_1 & b_2 & b_3 & b_4 & \cdots & b_{n-2} \\ 0 & 1 & b_1 & b_2 & b_3 & \cdots & b_{n-3} \\ 0 & 0 & 1 & b_1 & b_2 & \cdots & b_{n-4} \\ \vdots & & & & & & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 & b_1 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix}$$

In order to demonstrate this process, I will use a 5×5 matrix M, but the same idea can be extended to any size M. The matrix M can be broken up into two matrices as follows,

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

Define a new matrix N that has ones only on the upper sub diagonal. For the 5×5 example N will be,

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

The matrix N has interesting properties. As it is raised to a power, the following pattern occurs.

$$N^2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad N^3 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$N^4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad N^5 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

This means that N is nilpotent of degree 5. Using this property of N raised to powers, M can be rewritten as two matrices,

$$M = I + (aN + N^2) = I + (aI + N)N$$

Now, in general for matrices S and K, with dimension $n \times n$, if $S = I + K$ and $K^n = 0$ then $S^{-1} = I - K + K^2 - K^3 + \cdots + (-1)^{n-1}K^{n-1}$. This comes from expanding $S^{-1} = (I + K)^{-1}$ as a power series, until reaching $K^n = 0$.

Applying this fact to the matrix M with $K = (aI + N)N$ and working modulo 2 will produce the following result.

$$M^{-1} = I + (aI + N)N + (aI + N)^2N^2 + (aI + N)^3N^3 + \cdots + (aI + N)^{n-1}N^{n-1}$$

This can be rewritten using the terms from $\{b\}$ with $b_0 = I$.

$$M^{-1} = \sum_{n=0}^{\infty} b_n N^n$$

This means that b_n is the coefficient of N^n. In order to define b_n, it is necessary to determine how to find the coefficient of N^n. This amounts to finding the term involving N^n in $(aI + N)^m N^m$ and summing over m.
Now $(aI+N)^m = \sum_{k=0}^{m} \binom{m}{k}(aI)^{m-k}N^k$, so $(aI+N)^m N^m = \sum_{k=0}^{m} \binom{m}{k}(aI)^{m-k}N^{m+k}$.

Then, in order to examine the coefficients of N^n, the values of n that are needed are $n = m + k$. Then the coeffieient of N^n is $\sum_{n=m+k} \binom{m}{k}(aI)^{m-k}$, which will therefore equal b_n where both m and k vary with $m + k = n$.

Then an explicit definition of b_n can therefore be written as follows, for $a = \hat{A}_n$, I the $n \times n$ identity and $m \geq k$,

$$\sum_{n=m+k} \binom{m}{k}(aI)^{m-k}$$

This produces the following sequence that is desired when taken modulo 2 where I is represented as 1.

$$\begin{array}{lllll}
b_1 & = \binom{1}{0}a^1 & & = a & \\
b_2 & = \binom{2}{0}a^2 + \binom{1}{1}a^0 & & = a^2 + 1 & \\
b_3 & = \binom{3}{0}a^3 + \binom{2}{1}a & & = a^3 + 2a & = a^3 \\
b_4 & = \binom{4}{0}a^4 + \binom{3}{1}a^2 + \binom{2}{2}a^0 & & = a^4 + 3a^2 + a & = a^4 + a^2 + 1 \\
b_5 & = \binom{5}{0}a^5 + \binom{4}{1}a^3 + \binom{3}{2}a^1 & & = a^5 + 4a^3 + 3a & = a^5 + a \\
\vdots & & & &
\end{array}$$

There are several ways to develop the sequence b_n. Another approach is through 2×2 matrices of a certain form.

Starting with the matrix $M = \begin{bmatrix} 0 & 1 \\ 1 & a \end{bmatrix}$, and raising M to different powers produces a pattern. Here are the first few examples,

$$M^2 = \begin{bmatrix} 0 & 1 \\ 1 & a \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & a \end{bmatrix} = \begin{bmatrix} 1 & a \\ a & 1+a^2 \end{bmatrix}$$

$$M^3 = \begin{bmatrix} 0 & 1 \\ 1 & a \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & a \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & a \end{bmatrix} = \begin{bmatrix} a & 1+a^2 \\ 1+a^2 & 2a+a^3 \end{bmatrix}$$

The sequence b_n shows up in the entries of these matrices. In general,

$$M^n = \begin{bmatrix} b_{n-1} & b_{n-1} \\ b_{n-1} & b_n \end{bmatrix}$$

Then to find the next term M^{n+1},

$$M^{n+1} = \begin{bmatrix} b_{n-1} & b_{n-1} \\ b_{n-1} & b_n \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & a \end{bmatrix} = \begin{bmatrix} b_{n-1} & b_{n-2} + ab_{n-1} \\ b_n & b_{n-1} + ab_n \end{bmatrix}$$

This produces the exact same recursive equation that was seen in Section 5.2, so $b_{n+1} = b_{n-1} + ab_n$. This means that the terms of b_n can be computed using M^n and then taken modulo 2 which is needed for the game of Lights Out.

The characteristic polynomial of M is the quadratic $x^2 - ax - 1$, so its roots are the eigenvalues $\lambda_1 = \dfrac{a + \sqrt{a^2+4}}{2}$ and $\lambda_2 = \dfrac{a - \sqrt{a^2+4}}{2}$.

Consider the matrix $M - \lambda_i I$ and its reduced row echelon form.

$$M - \lambda_i I = \begin{bmatrix} -\lambda_i & 1 \\ 1 & a - \lambda_i \end{bmatrix} \to \begin{bmatrix} 1 & a - \lambda_i \\ 0 & 0 \end{bmatrix}$$

The solutions to this will have the form $t \begin{bmatrix} \lambda_i - a \\ 1 \end{bmatrix} = t \begin{bmatrix} \dfrac{-a \pm \sqrt{a^2+4}}{2} \\ 1 \end{bmatrix}$

Set P equal to the basis for all solutions to $M - \lambda_i I$ where each element is a column. There will be only two columns, λ_1 and λ_2.

$$P = \begin{bmatrix} \dfrac{-a + \sqrt{a^2+4}}{2} & \dfrac{-a - \sqrt{a^2+4}}{2} \\ 0 & 0 \end{bmatrix}$$

Then $det(P) = \sqrt{a^2+4}$, and $P^{-1} = \dfrac{1}{\sqrt{a^2+4}} \begin{bmatrix} 1 & \dfrac{a + \sqrt{a^2+4}}{2} \\ -1 & \dfrac{-a + \sqrt{a^2+4}}{2} \end{bmatrix}$.

This means that $P^{-1}MP = \begin{bmatrix} \lambda_1 & 0 \\ \lambda_2 & 0 \end{bmatrix}$ so,

$$M = P \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} P^{-1}$$

$$M^n = P \begin{bmatrix} \lambda_1^n & 0 \\ 0 & \lambda_2^n \end{bmatrix} P^{-1}$$

It was shown before that for powers of M^n, the bottom right entry will be b_n.

Then, computing the lower right entry of the above multiplication will produce,

$$
\begin{aligned}
b_n &= \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1^n & 0 \\ 0 & \lambda_2^n \end{bmatrix} \left(\frac{1}{\sqrt{a^2+4}} \begin{bmatrix} \dfrac{a+\sqrt{a^2+4}}{2} \\ \dfrac{-a+\sqrt{a^2+4}}{2} \end{bmatrix} \right) \\
&= \frac{1}{\sqrt{a^2+4}} \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1^n & 0 \\ 0 & \lambda_2^n \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} \\
&= \frac{1}{\sqrt{a^2+4}} \begin{bmatrix} \lambda_1^n & \lambda_2^n \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} \\
&= \frac{1}{\sqrt{a^2+4}} \left(\lambda_1^{n+1} - \lambda_2^{n+2} \right)
\end{aligned}
$$

Therefore, another way to define the sequence b_n is in terms of the eigenvalues of the matrix M.

Another approach is to look at a specific power series expansion. Using the recursive definition of b_n as described before, $b_0 = 1, b_1 = a, ..., b_{n+1} = ab_n + b_{n-1}$, then define,

$$f(x) = \sum_{i=0}^{\infty} b_i x^i$$

Then the three following equations are true,

$$-f(x) \quad = -(b_0 + b_1 x + b_2 x^2 + \cdots + b_{n+1} x^{n+1}) \cdots \tag{5.6}$$

$$axf(x) \quad = ab_0 x + ab_1 x^2 + ab_2 x^3 + \cdots + ab_n x^{n+1} \cdots \tag{5.7}$$

$$x^2 f(x) \qquad = b_0 x^2 + b_1 x^3 + \cdots + b_{n-1} x^{n+1} \tag{5.8}$$

Adding these three equations together yields,

$$(x^2 + ax - 1) f(x) = -b_0 + (-b_1 + ab_0)x = -1 + 0x$$

Therefore,

$$f(x) = \frac{-1}{x^2 + ax - 1}$$

The coeffcients of the taylor expansion of this function $f(x)$ will be the sequence b_n.

These definitions of b_n makes it easy to quickly determine the terms of this sequence for any size n. Having this explicit definition of $\{b\}$ makes it possible to produce a computer program that could generate the critical matrix and its inverse that will help determine the critical lower right corner of the A' matrix for any size n which has been shown to be significant in the study of quiet patterns and light chasing. A further investigation of this sequence could reveal a predictable pattern that would make it possible to anticipate which game boards will have quiet patterns, as well as justify the claim that the non-pivot columns for any game board are the last $n^2 - r$ columns.

5.10 Conclusion and Further Research Ideas

The possibilities for further research on this topic are endless. Even limited to the classic game of Lights Out on square grids, there is still more work to be done to anticipate when game boards will have quiet patterns by studying the sequence of b_n in even more detail. While some of the introductory work on these topics has been addressed in this paper, there is plenty of room to extend these ideas even further.

The many variations of the game of Lights Out discussed in Chapter 1 are also all opportunities for more research. Changing the size of the game board, the rules of which lights are effected by a button press, and the possible state of the lights all change the game and the calculations done significantly. One idea that sparked my interest is how to approach games with directed edges, which would bring a whole new dimension to the game since the order of button presses would matter.

The most interesting extension of the research started in this paper is how it can be applied practically to scientific modeling. This game is directly linked with the theory of finite cellular automaton. Starting with an initial configurations of cells and a few simple rules about how they interact with their neighbors, a simulation of the change in the cells states can be run over time. A basic example of this theory is Conway's Game of Life, which is a simulation that can be used to predict the population of predators and their prey. It is predicted by Stephen Wolfram, that the use of cellular automata will some day produce scientific discoveries that have so far been left unexplained due to our limited view of mathematical modeling. While the exact applications of the game of Lights Out are currently unknown, this game is strongly tied with the ideas needed to understand this increasingly important scientific theory.

Bibliography

[1] Dodis, Yevgeniy and Peter Winkler. "Universal Configurations in Light-Flipping Games." *Society for Industrial and Applied Mathematics* (2001):n.pag.Print.

[2] Eriksson, H. "Note on the Lamp Lighting Problem." *Advances in Applied Mathematics* 27.2-3 (2001):357-66. Print.

[3] Joyner, David. *Adventures in Group Theory: Rubik's Cute, Merlin's Machine, and Other Mathematical Toys.* Baltimore: Johns Hopkins UP, 2002. Print.

[4] "Lights Out (game)." *Wikipedia.* N.p., 05 Mar.2013. Web. May 2013.

[5] Mulholland, Jamie. "Lecture 24: Lights Out Puzzle." *SFU Department of Mathematics.* N.p., Mar. 2011. Web. May 2013.

[6] Nowakowski, Richard J. *More Games of No Chance.* Cambridge: Cambridge UP, 2002. Print.

[7] Pelletier, Don. "Merlin's Magic Square," (1987):n. pag. Print.

[8] Scherphuis, Jaap. "Jaap's Puzzle Page." *Lights Out Mathematics.* N,p.,n.d. Web. May 2013.

[9] Sutner, Klaus. "The Sigma Game and Cellular Automata." *Stevens Institute of Technology* (n.d):n.pag. Print.

[10] Wolfram, Stephen. *A New Kind of Science*. Champaign, IL: Wolfram Media, 2002. Print.

[11] Wolfram, Stephen. "Geometry of Binomial Coefficients." *American Mathematical Monthly* 91.9 (1984): n pag. Print.

Appendix

Matlab Programs

Several programs were written to aid in the large calculations done throughout this paper and to ensure the modulo 2 calculations were performed at each step. The code for the following programs can be found on the next several pages,

- Ahat: generates the $\hat{A}_n$ matrix needed to create the adjacency matrix A and also used in calculating terms of $\{b\}$.
- GenAdj: generates the adjacency matrix A_n for any size n under the rules of the classic game of Lights Out.
- InvSol: Calculates the augmented matrix containing the reduced row echelon form of A in the first n^2 columns and the inverse or pseudo inverse calculated modulo 2 at each step in the last n^2 columns.
- Prref: Reduces any square matrix modulo 2 at each step. Useful in row reducing the terms of $\{b\}$.

```
function ahat =Ahat(n)
% Creates the adjacency linegraph/hat matrix of size n
A = eye(n);             % Start by putting one's on diagonal
for x= 1:n              % Place 1's in upper diagonal
   if x>1
       A(x,x-1)=1;
    end
   if x<n               % Place 1's in lower diagonal
       A(x,x+1)=1;
   end
end

ahat= A;
end
```

```
function adjmatrix  = GenAdj( n )
% Generates an adjacency matrix for square game boards with nxn vertices with + shape
edges

% Create identity matrix size n
I = eye(n);

% Create A (linegraph/hat matrix)
A = eye(n);             % Start by putting one's on diagonal
for x= 1:n              % Place 1's in upper diagonal
   if x>1
       A(x,x-1)=1;
    end
   if x<n               % Place 1's in lower diagonal
       A(x,x+1)=1;
   end
end

% Build M, the adjacency matrix, using I and A
m=n*n;                              % Dimension of A
M= zeros(m);                        % Start out with all zeros as placeholders
for x=1:n                           % essentially inputting matricies at each n entry
    for y=1:n
        if x==y                     %along the diagonal, copy the A matrix entries
            M= copy1(M,A,x,y,n);
        elseif abs(x-y)==1;         % along the upper and lower diag copy I
            M= copy1(M,I,x,y,n);
        end
    end
end

adjmatrix=M;
end

function q = copy1(src, hat, locx, locy, n)
%

%src is M that was set up initally as all 0's (changing at each step)
%hat is either A or I
%locx and locy correspond to the 1-n index



%changes nxn matrix to n^2xn^2 matrix
%copies hat(n,n) into src(n^2,n^2)

i=1; %n based row
for a=(locx-1)*n+1:locx*n %n*n based row
    j=1; %n based column
    for b=(locy-1)*n+1:locy*n %n*n based column
        src(a,b)= hat(i,j);
        j=j+1;
    end
    i=i+1;
end
q=src;
end
```

```
function invsol = InvSol(n)
% Finds the inverse or pseduo inverse of the adjacency matrix with the following
specifications:
% nxn dimension game boards with cross pattern edges with light state set size 2 (on
or off)
% Also outputs the number of quiet patterns in the game board
% Need GenAdj function code to run this function

M=GenAdj(n);    % GenAdj is the function written to generate the adjacency matrix for
an nxn game board
m=n*n;          % The size of the adjacency matrix is actually n^2 (this is the number
of vertices)

M= mod(M,2);    % Ensure inputs are mod 2, as they should already be

M= [M eye(m)];  % Augment adjacency matrix with identity matrix

%FORWARD ELIMINATION (mod 2 simplified):
% Moving through columns from left to right, ensures a 1 in the pivot position when
possible and eliminates all 1's below the pivot using row operations

startback=0;    % Used to determine when to start backward elimination
finalcol=0;     % Used to determine which was the last non-zero column (rank of
matrix)

for col=1:m                                     % Scan through columns left to right
    for row=col:m                               % Scan through rows from diagonal
entry down
        if row==col                             % In diagonal position: create pivots
                if M(row, col)==1               % If a 1 already in the diagonal,
pivot already there
                    finalcol=col;               % Update status of rank of the matrix
                    continue;
                else                            % If a one is not in the diagonal
                    foundit=0;                  % Used to advance on to backward
elimination if no swap is possible
                    for row2=row+1:m            % Scan down the rows below the
diagonal
                        if M(row2,col)==1       % If a one is found below the diagonal
                            temp= M(row,:);     % Swap this row and the pivot row
                            M(row,:)=M(row2,:);
                            M(row2,:)=temp;
                            foundit=1;          % Swapped, now move on to eliminate
ones below the pivot and continue forward elimination
                            break;
                        end
                    end
                    if foundit==0               % If there is not a 1 below the
diagnoal, no pivot possible
                                                % Start backward elimination
                        startback=1;
                        break;                  % Break out of forward elimination
                    end
                end
        else                                    % Eliminating ones below the pivot
            if M(row,col)==0                    % If already a zero entry, no change
necessary
                continue;
            else                                % For rows with one entries, replace
this row with the sum of current row and pivot row
                M(row,:)= mod(M(col,:)+ M(row,:),2);
            end
        end
    end

    if startback==1
        break;                                  % Break out of forward eliminaton
    end
end

%Displays the rank and number of quiet patterns
str=fprintf('Rank = %d\n', finalcol);
str=fprintf('Nullity = %d\n', m-finalcol);

%BACKWARD ELIMINATION (mod 2 simiplified):
% Moves along columns right to left eliminating ones above each pivot, creating a
identity matrix on the left (or as close as possible)

for col=finalcol:-1:1                           % Scan through columns right to left
    if M(col,col) ~= 1                          % Assert that the pivot is a 1 (should
be at this point)
        disp('Non-zero pivot at this column in back elimination')
        col                                     % Display column where this error
occurs
        invsol=M;
        return
    end
    for row=col-1:-1:1                          % Scan through rows from one above
diagonal, to the top of the matrix
        if M(row,col)~=0                        % When there is a 1, eliminate the 1
by replacing row with sum of current row and pivot row
            M(row,:)= mod(M(col,:)+ M(row,:),2);
        end
    end
end
invsol=M;
end
```

```
function prref = Prref(a,n)
%Row reduces any square matrix a (must input the dimension n) using mod 2
%at each step

m=n;          % The size of the adjacency matrix is actually n^2 (this is the number
of vertices)

%FORWARD ELIMINATION (mod 2 simplified):
% Moving through columns from left to right, ensures a 1 in the pivot position when
possible and eliminates all 1's below the pivot using row operations

startback=0;    % Used to determine when to start backward elimination
finalcol=0;     % Used to determine which was the last non-zero column (rank of
matrix)

for col=1:m                                     % Scan through columns left to right
    for row=col:m                               % Scan through rows from diagonal
entry down
        if row==col                             % In diagonal position: create pivots
                if a(row, col)==1               % If a 1 already in the diagonal,
pivot already there
                    finalcol=col;               % Update status of rank of the matrix
                    continue;
                else                            % If a one is not in the diagonal
                    foundit=0;                  % Used to advance on to backward
elimination if no swap is possible
                    for row2=row+1:m            % Scan down the rows below the
diagonal
                        if a(row2,col)==1       % If a one is found below the diagonal
                            temp= a(row,:);     % Swap this row and the pivot row
                            a(row,:)=a(row2,:);
                            a(row2,:)=temp;
                            foundit=1;          % Swapped, now move on to eliminate
ones below the pivot and continue forward elimination
                            break;
                        end
                    end
                    if foundit==0               % If there is not a 1 below the
diagnoal, no pivot possible
                                                % Start backward elimination
                        startback=1;
                        break;                  % Break out of forward elimination
                    end
                end
        else                                    % Eliminating ones below the pivot
            if a(row,col)==0                    % If already a zero entry, no change
necessary
                continue;
            else                                % For rows with one entries, replace
this row with the sum of current row and pivot row
                a(row,:)= mod(a(col,:)+ a(row,:),2);
            end
        end
    end

    if startback==1
        break;                                  % Break out of forward eliminaton
    end
end

%Displays the rank and number of quiet patterns
```

```
%str=fprintf('Rank = %d\n', finalcol);
%str=fprintf('Nullity = %d\n', m-finalcol);

%BACKWARD ELIMINATION (mod 2 simiplified):
% Moves along columns right to left eliminating ones above each pivot, creating a
identity matrix on the left (or as close as possible)

for col=finalcol:-1:1                           % Scan through columns right to left
    if a(col,col) ~= 1                          % Assert that the pivot is a 1 (should
be at this point)
        disp('Non-zero pivot at this column in back elimination')
        col                                     % Display column where this error
occurs
        prref=a;
        return
    end
    for row=col-1:-1:1                          % Scan through rows from one above
diagonal, to the top of the matrix
        if a(row,col)~=0                        % When there is a 1, eliminate the 1
by replacing row with sum of current row and pivot row
            a(row,:)= mod(a(col,:)+ a(row,:),2);
        end
    end
end
prref=a;
end
```

Vita

Author: Rebecca S. Meyer

Place of Birth: Bellevue, WA

Undergraduate Schools Attended:

Whitworth University

Degrees Awarded:

Bachelor of Science in Mathematics, 2011

Whitworth University