VOL. 88, NO. 1, FEBRUARY 2015 1

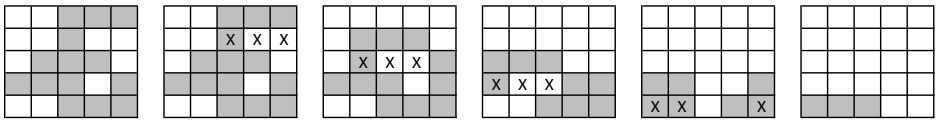
Chasing the Lights in Lights Out

C. David Leach
University of West Georgia
Carrollton, GA 30118
cleach@westga.edu

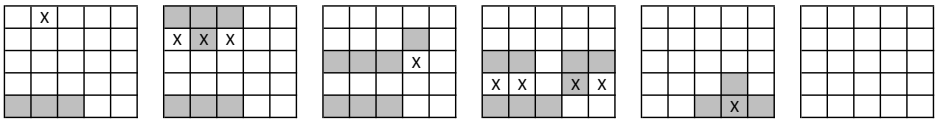
Lights Out is a game played on a 5×5 grid of light-up buttons. It was marketed in the 1990s as a handheld electronic game by Tiger Electronics®, and versions and variations of it are available online. Each button can be in one of two states: on or off. The neighbors of a button include the buttons directly above, below, right, and left of the button. When a button is pressed, its state and the state of its neighbors are toggled. At the start of the game, the lights are lit in a seemingly random configuration; the goal is to turn all of the lights off.

Determining the optimal solution–one requiring the fewest button presses–for a given configuration can be done using Gauss-Jordan elimination. However, the matrices involved are rather large and do not lend themselves well to repeated play of the game.

Let's play a game using a very simple strategy: starting with the second row, press every button in that row that is immediately beneath a light in row 1 that is on. After completing this task, all lights in row 1 are off. Now do the same with row 3, pressing those buttons beneath the lights in row 2 that are now lit. Continue row by row until you get to the bottom of the board. The diagram below shows our progress through a game, with the initial configuration shown on the leftmost board. White cells are off, shaded cells are on, and X represents a button being pressed.



At this point our simple strategy seems to break down, since it does not provide any means for turning out the lights in the bottom row. With no clear way to proceed, suppose that we "randomly" press the second button in the first row. This turns on a few lights, so we repeat the process of clearing the rows from top to bottom; however, this time when we finish, all the lights are off.



The strategy we employed here is popularly known as *light chasing*. Light chasing consists of three stages. Stage 1, shown in the first diagram above, is a what we term a *chase*. At the end of stage 1, the only lights that remain on are those in the bottom row. Stage 2 involves a lookup using Table 1, which can be found on the web at a number of places, including [**14**]. If we let 0 and 1 represent the off and on states respectively, the bottom row of our board gives the configuration 11100. The lookup table tells us to press only the second button in the top row. If the lights in the bottom row are not in one of the configurations in the lookup table, then the game is unsolvable. Stage 3 is another chase, only at the end of stage 3 all the lights are off.

Light chasing requires practically no computation, but it does require that we have the lookup table at our disposal. Generating the lookup table takes a bit of computational effort, but once it is done, it can be used repeatedly.

Lights Lit in Bottom Row	Buttons to Press in Top Row
00111	00010
01010	10010
01101	10000
10110	00001
10001	11000
11011	00100
11100	01000

TABLE 1: Lookup table for Lights Out.

History

The mathematics behind Lights Out has been studied by many different authors, and several results have been discovered and published more than once. This has led to confusion as to who made certain discoveries first.* Much of the reason for this confusion is because different authors approached the problem from different perspectives and used different terminology to describe the game. The name *Lights Out* was introduced by Tiger Electronics in 1995, and many articles, including [**1**] have used that name. Other names that have been used to describe the same problem include the *switch-setting problem* [**3, 4**] the *lamp-lighting problem* [**10**], and the *nine tails problem* (for a 3×3 board). Some authors approached the game as a 2-dimensional cellular automaton, and used the names σ*-game* and σ*-automata* [**2, 7, 13**]. The earliest reference to a Lights Out-type problem comes from [**11**], in which Lovász credits Tibor Gallai for solving the *all-ones problem*, a related problem in graph theory, in the early 1970s.

The concept of light chasing has been examined for the on-off game in [**2, 3, 4, 13**]. The multicolor case, in which the lights cycle through a sequence of colors before turing off, was examined in [**7**]. However, the term *light chasing*, while popular on internet discussions of Lights Out, does not appear in any of those articles.

The main purpose of this paper is to examine light chasing, with a particular emphasis on generating lookup tables. We will begin by generating Table 1, then generate lookup tables for variations of the game with grids of other sizes and lights that cycle through multiple colors. The methods we use can be implemented using the free software SageMath [**12**] or another computer algebra system.

Modeling the game

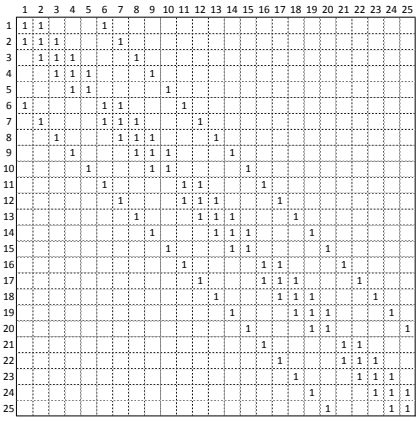
Since a light has two states–on and off–we can represent its state with an element from $\mathbb{Z}_2$. Pressing a button adds 1 to its state and its neighbors' states modulo 2. This

*The author would like to thank the anonymous referees for providing much of the historical information contained in this section.

simple observation has two useful consequences: since addition is commutative, the order in which the buttons are pressed is unimportant; since addition in $\mathbb{Z}_2$ is an order 2 operation, no button needs to be pressed more than once. At the beginning of a game, the lights have some initial configuration. There are 2^{25} possible configurations, but only 1/4 of them are solvable [**1**].

Before we analyze light-chasing, let us briefly outline the usual method for solving the game with linear algebra. For a more detailed description, see [**1**]. We label the 25 buttons from the upper left to the lower right as $b_1, \ldots, b_{25}$. A configuration of the game board is described by the vector $\vec{b} = [b_1, b_2, \ldots, b_{25}]$, where $b_i = 1$ if light i is on and 0 if off. We define the 25×25 matrix M by $m_{i,j} = 1$ if pressing button b_i affects light b_j and 0 otherwise. Solving $M\vec{x} = \vec{b}$ for $\vec{x}$ gives the set of buttons to press to turn all the lights off.

Let A be the the tridiagonal 5×5 submatrix in the upper left corner of M. The entries of A describe which buttons in row i affect which lights in row i. The matrix A is repeated along the diagonal–once for each row on the game board–and is used extensively in the analysis of light chasing.



$= M$

Light chasing

Before we attempt to solve the game using light chasing, let's examine the effect that a single chase would have on the rows of the game board if we were to begin with all the lights off and we press an arbitrary set of buttons in the top row. For each row i we define two vectors: let $\vec{p}_i = [p_{i1}\ p_{i2}\ p_{i3}\ p_{i4}\ p_{i5}]$, where p_{ij} is the number of times that the j^{th} button in row i will be pressed during the chase; let $\vec{s}_i = [s_{i1}\ s_{i2}\ s_{i3}\ s_{i4}\ s_{i5}]$, where s_{ij} is the state of the j^{th} light in row i after row i is pressed, but before row $i+1$ is pressed during the chase. Although $\vec{p}_i$ and $\vec{s}_i$ refer to rows of the game board, in all of our calculations, they will be used as column vectors. Once we choose an arbitrary $\vec{p}_1$, the remaining $\vec{p}_i$ and all the $\vec{s}_i$ vectors are completely determined. The value of $\vec{s}_i$ is determined only by the buttons pressed in rows i and $i-1$, and is given by the following equation from [**4**]:

$$\vec{s}_i = A\vec{p}_i + \vec{p}_{i-1}. \tag{1}$$

We adopt the convention that $\vec{p}_0 = \vec{0}$, since there are no buttons to be pressed above row 1. To turn off the lights in row $i-1$, we must have that

$$\vec{p}_i = -\vec{s}_{i-1} \tag{2}$$

for $i \geq 2$. Since our calculations are being done over $\mathbb{Z}_2$, we could dispense with negatives and rewrite equation (2) as $\vec{p}_i = \vec{s}_{i-1}$; however, we will retain the negative here and in subsequent equations in order to generalize to $\mathbb{Z}_p$ for $p > 2$ later. Starting with $\vec{p}_0 = \vec{0}$ and arbitrary $\vec{p}_1$, by alternately applying equations (1) and (2), we can write

$$\vec{s}_i = S_i \vec{p}_1, \tag{3}$$

where S_i is a 5×5 matrix. We can check that for $1 \leq i \leq 5$, the S_i are given by the polynomials $S_1 = A$, $S_2 = -A^2 + I_5$, $S_3 = A^3 - 2A$, $S_4 = 3A^2 - A^4 - I_5$, and

$$S_5 = A^5 - 4A^3 + 3A = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix}. \tag{4}$$

Since row 5 is the last row, the vector $S_5\vec{p}_1$ tells the overall effect that the chase has on the last row. More precisely, the chase has the overall effect of adding the vector $S_5\vec{p}_1$ to the last row. The other rows all get zeroed out by the end of the chase.

In our computation of S_5, we use a recursive process that requires computing S_i for all $i \leq 5$, but it's possible to compute any value of S_i directly. Goldwasser et al. [**3**] were the first to show that S_i is given by the $(k+1)^{st}$ Fibonacci polynomial, reduced modulo 2 and evaluated at A. Later we will examine this connection in more detail. For now, let's consider an actual game.

Suppose we have already completed the first chase and the only lights that remain on are in row 5. Let $\vec{s}_b$ (b for bottom) be the state vector of row 5 at this stage of the game. Note that $\vec{s}_b$ is the state of row 5 immediately after the *first* chase is completed, while $\vec{s}_5$ is the state of row 5 after the *second* chase is completed. It is our goal to determine the value of $\vec{p}_1$ that will result in $\vec{s}_5 = \vec{0}$ for the given value of $\vec{s}_b$. Thus we need to solve the equation $S_5\vec{p}_1 + \vec{s}_b = \vec{0}$ for $\vec{p}_1$, which we rearrange to get

$$S_5\vec{p}_1 = -\vec{s}_b. \tag{5}$$

Since $\text{rank}(S_5) = 3$ and $\mathbb{Z}_2$ has only two possible values for each coefficient, the column space of S_5 contains exactly eight vectors–precisely the seven bottom-row vectors listed in the lookup table along with the zero vector.

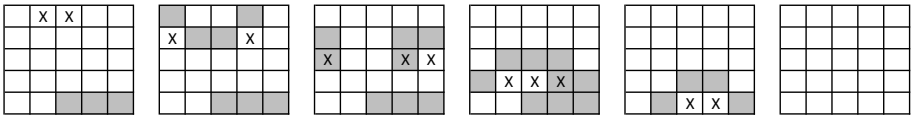
Note that the top row given in the lookup table is not the only solution possible. For example, suppose that the bottom row is $\vec{s}_b = [0\,0\,1\,1\,1]$, and let $\vec{v}_i$ be the i^{th} column of S_5. Solving

$$c_1\vec{v}_1 + c_2\vec{v}_2 + c_3\vec{v}_3 + c_4\vec{v}_4 + c_5\vec{v}_5 = [0\,0\,1\,1\,1] \tag{6}$$

gives the general solution

$$\vec{p}_1 = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} + c_4 \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} + c_5 \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \tag{7}$$

for any choice of c_4 and c_5. If we let both be zero, we get $\vec{p}_1 = [0\,1\,1\,0\,0]$, and the game proceeds as follows:



If we choose $c_4 = 1$ and $c_5 = 0$, we get $\vec{p}_1 = [0\,0\,0\,1\,0]$, as given in the lookup table.

Variations of the game

Grids with five columns and m rows. First, let's keep the number of columns at 5, and let m be the number of rows. As before we can alternately apply equations (1) and (2) to find matrix S_m for any value of m that we like and proceed exactly as above. However, if m is large, this becomes tedious. It would be preferable to find a closed-form formula for S_m. One way to compute S_m is by using the Fibonacci polynomials.

The Fibonacci polynomials are defined by $F_1(x) = 1$, $F_2(x) = x$, and $F_n(x) = xF_{n-1}(x) + F_{n-2}(x)$ for $n \geq 3$, and are given by the well-known formula

$$F_n(x) = \sum_{i=0}^{\lfloor (n-1)/2 \rfloor} \binom{n-i-1}{i} x^{n-2i-1}. \tag{8}$$

Several authors [**2, 3, 13**] have shown that S_m, when reduced modulo 2, is equivalent to $F_{m+1}(A) (\text{mod } 2)$. Making the necessary substitutions into equation (8) gives

$$S_m = F_{m+1}(A) = \sum_{i=0}^{\lfloor m/2 \rfloor} \binom{m-i}{i} A^{m-2i} \ (\text{mod } 2). \tag{9}$$

Another way to compute S_m is by combining equations (1) and (2) to get the second-order recurrence relation

$$\vec{p}_{i+1} = -A\vec{p}_i - \vec{p}_{i-1} \tag{10}$$

with given initial conditions $\vec{p}_0 = \vec{0}$ and $\vec{p}_1$. In order to solve this for $\vec{p}_i$, we rewrite the recurrence as the first-order recurrence

$$\begin{bmatrix} \vec{p}_{i+1} \\ \vec{p}_i \end{bmatrix} = \begin{bmatrix} -A & -I \\ I & \vec{0} \end{bmatrix} \begin{bmatrix} \vec{p}_i \\ \vec{p}_{i-1} \end{bmatrix}. \tag{11}$$

Let L denote the square matrix in recurrence (11), and note that L is a 10×10 partitioned matrix. Repeatedly applying recurrence (11) amounts to repeated multiplication, giving the following general formulas for $\vec{p}_i$ and $\vec{s}_i$:

$$\begin{bmatrix} \vec{p}_{i+1} \\ \vec{p}_i \end{bmatrix} = L^i \begin{bmatrix} \vec{p}_1 \\ \vec{p}_0 \end{bmatrix} \qquad \begin{bmatrix} \vec{s}_i \\ \vec{s}_{i-1} \end{bmatrix} = -L^i \begin{bmatrix} \vec{p}_1 \\ \vec{p}_0 \end{bmatrix} \tag{12}$$

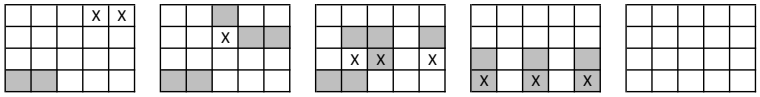
From equation (12) we can extract the equation $\vec{s}_i = S_i\vec{p}_1$, where S_i is the upper left quadrant of the matrix $-L^i$.

Example: Suppose $m = 4$ and $\vec{s}_b = [1\,1\,0\,0\,0]$. We need to solve the equation $S_4\vec{p}_1 = [1\,1\,0\,0\,0]$. To obtain S_4, we take the upper left quadrant of $-\begin{bmatrix} -A & -I \\ I & 0 \end{bmatrix}^4$, which is

$$S_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{13}$$

This particular matrix is quite nice for a few reasons: Since $\text{rank}(S_4) = 5$, all configurations are solvable; since the back diagonal contains all ones, $S_4\vec{x}$ reverses the vector $\vec{x}$, making for a very easy-to-remember strategy.

Solving $S_4\vec{p}_1 = [1\,1\,0\,0\,0]$ gives $\vec{p}_1 = [0\,0\,0\,1\,1]$, and the game proceeds as follows:



Grids with other than 5 columns. Changing the number of columns affects the sizes of the matrices and vectors involved, but the analysis of the game is otherwise unchanged. If we let n denote the number of columns, then A is the $n \times n$ matrix defined by $a_{ij} = 1$ if $|i - j| \leq 1$ and 0 otherwise, and vectors $\vec{p}_i$ and $\vec{s}_i$ now have n entries instead of 5. With these modifications, we can proceed exactly as before.

Lights with more than two states. Another common variation on the lights out game is to have lights that take on more than two states. The states are sometimes represented with different colors, but we will consider them to be the numbers in $\mathbb{Z}_p = \{0, \ldots, p-1\}$, with 0 being considered off. We will restrict ourselves to the case where p is a prime number so that each element of $\mathbb{Z}_p$ has a multiplicative inverse, making $\mathbb{Z}_p$ a finite field. This restriction is not necessary, but it simplifies the tasks of solving $S_m\vec{p}_1 = -\vec{s}_b$ and finding $\text{rank}(S_m)$ on a computer algebra system, since S_m can be easily row-reduced.

A button's state value increases by 1 (mod p) whenever it or one of its neighbors is pressed. The 0 state is considered off. As in the $\mathbb{Z}_2$ case, the order in which the buttons are pressed is unimportant. In optimal play, no button would need to be pressed more than $p - 1$ times; with light chasing no button needs to be pressed more than $p - 1$ times during each chase. We can use equation (12) to compute S_m, by proceeding as before but performing all arithmetic in the field $\mathbb{Z}_p$.

If we want a closed-form expression for S_m, we can't use equation (9) for $p > 2$; however, we can modify it to get the following closed-form formula for S_m:

$$S_m = \sum_{i=0}^{\lfloor m/2 \rfloor} (-1)^{n-i+1} \binom{m-i}{i} A^{m-2i}. \tag{14}$$

This formula accounts for the sign changes that occur during the recursive calculation of S_m, whereas equation (9) simply discards them since sign changes have no effect modulo 2. This is essentially the method shown in [**7**] involving Chebyshev polynomials.

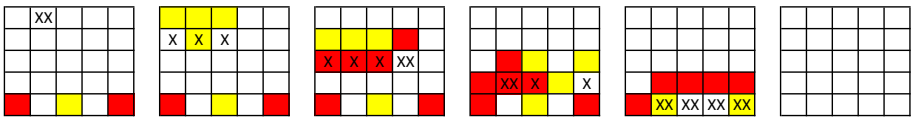
Let's examine the 5×5 grid over $\mathbb{Z}_3$. (This game is playable on Tiger Electronics' Lights Out 2000, and on the web at many places including [**8**] and [**9**]). The matrix A *looks* identical to the A from before, but the entries are now from $\mathbb{Z}_3$. Using equation (9) or equation (14) to compute S_5 over $\mathbb{Z}_3$ gives

$$S_5 = \begin{bmatrix} 2 & 1 & 1 & 1 & 2 \\ 1 & 0 & 2 & 0 & 1 \\ 1 & 2 & 2 & 2 & 1 \\ 1 & 0 & 2 & 0 & 1 \\ 2 & 1 & 1 & 1 & 2 \end{bmatrix}.$$

Example: Suppose that the bottom row is $\vec{s}_b = [1\,0\,2\,0\,1]$. We wish to add $-\vec{s}_5 = [2\,0\,1\,0\,2]$ to the bottom row, so we solve $S_5\vec{p}_1 = [2\,0\,1\,0\,2]$ for $\vec{p}_1$. We get the general solution

$$\vec{p}_1 = \begin{bmatrix} 0 \\ 2 \\ 0 \\ 0 \\ 0 \end{bmatrix} + c_3 \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + c_4 \begin{bmatrix} 0 \\ 2 \\ 0 \\ 1 \\ 0 \end{bmatrix} + c_5 \begin{bmatrix} 2 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{15}$$

where c_3, c_4, and c_5 are any scalars from $\mathbb{Z}_3$. If we choose them all to be zero, we get $\vec{p}_1 = [0\,2\,0\,0\,0]$. Letting red and yellow represent states 1 and 2 respectively, the game proceeds as follows (2 X's in a cell means the button is pressed twice):



Now let's create a lookup table for this game. The column space of S_5 consists of all solvable bottom-row configurations; since $\text{rank}(S_5) = 2$, there are nine such vectors. Each vector has the form $c_1\vec{v}_1 + c_2\vec{v}_2$, where $\vec{v}_1 = [1\,0\,2\,0\,1]$ and $\vec{v}_2 = [0\,1\,0\,1\,0]$ form a basis of the column space.

Using all possible values of c_1 and c_2 (9 pairs total), we get the nine solvable bottom-row configurations. Each is a value of $\vec{s}_b$ for which $S_m\vec{p}_1 = -\vec{s}_b$ has a solution. For each of these, we solve for $\vec{p}_1$, which is listed in the table as the Top Row vector to be pressed. We get the table shown below.

Now that we have the tools, we leave the reader with a few exercises. With the help of a computer algebra system, create lookup tables for other grid sizes and prime numbers of colors. You might first try these particular values, and check your tables by playing online at a site such as [**9**].

1. the 4×4 game over $\mathbb{Z}_2$.
2. the 4×4 game over $\mathbb{Z}_5$.
3. the 7×7 game over $\mathbb{Z}_3$.

REFERENCES

1. Marlow Anderson and Todd Feil, Turning Lights Out with Liner Algebra, *Math. Mag* **71** (1998) no.4 300–303.
2. R. Barua and S. Ramakrishnan. σ-game, σ^+-game, and two-dimensional cellular automata. Theoretical Computer Science, 154(2):349-366, 1996.
3. J. Goldwasser, W. Klostermeyer, G. Trapp, and C.-Q. Zhang (1995), Setting Switches on a Grid, Technical Report TR-95-20, Dept. of Statistics and Computer Science, West Virginia University.
4. J. Goldwasser, W. Klostermeyer, and G. Trapp, (1997), Characterizing Switch-Setting Problems, Linear and Multilinear Algebra, vol. 43, pp. 121- 135.

		Bottom Row	Top Row
c_1	c_2	Configuration	Vector to Press
0	0	00000	00000
0	1	01010	22000
0	2	02020	11000
1	0	10201	02000
1	1	11211	21000
1	2	12221	10000
2	0	20102	01000
2	1	21112	20000
2	2	22122	12000

TABLE 2: Lookup table for 5×5 over $\mathbb{Z}_3$.

5. John Goldwasser and William Klostermeyer, Maximization Versions of Lights Out" Games in Grids and Graphs, Congressus Numerantium 126 (1997) 99-111.
6. John Goldwasser, William Klostermeyer & Henry Ware, Fibonacci Polynomials and Parity Domination in Grid Graphs, Graphs and Combinatorics 18 (2002) 271-283.
7. M. Hunziker, A. Machiavelli, J. Park, "Chebyshev polynomials over finite fields and reversibility of sigma-automata on square grids," Theoretical Computer Science 320 (2004), 465-483.
8. Jaap, Scherphuis, *Jaap's Puzzle Page,* `http://www.jaapsch.net/puzzles/lights.htm`.
9. Kato, N., *A Lights Out Puzzle with Solver* `http://www.ueda.info.waseda.ac.jp/~n-kato/lightsout/index.html`.
10. Jonas Sjstrand, Note on the Lamp Lighting Problem, Advances in Applied Math. 27 (2001) 357-366.
11. Lovsz, L. *Combinatorial problems and exercises*, Ch 5, problem 17c, North-Holland Publishing Co., Amsterdam-New York, (1979).
12. SageMath, the Sage Mathematics Software System, The Sage Developers, http://www.sagemath.org.
13. K. Sutner, σ-automata and Chebyshev polynomials, Theoretical Computer Science, 230:49-73, 2000. (Author claims this was written in 1996; see, for example, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.27.1593)
14. Lights Out, Wikipedia, `https://en.wikipedia.org/wiki/Lights_Out_(game)`.

Summary

Lights Out is a game played on a grid of light-up buttons. At the start of the game, the lights are lit in a seemingly random configuration. A button changes states whenever it or one of its neighbors is pressed. The goal is to turn all of the lights out. Light chasing is a simple strategy for winning Lights Out by using a lookup table. In this article we examine how to construct these lookup tables for several variations of the game.

Author Biographical Sketch

David Leach (MR Author ID: 684867) received his Ph.D. from Auburn University in 2002 and has been a member of the mathematics faculty at the University of West Georgia since then. His primary research areas are graph theory and combinatorics. He enjoys mixing mathematics with his interests in music, electronics, toys, and games, as evidenced by the present article on Lights Out. He and his wife Lori live in a house with several cats.