USING MATHEMATICA TO SOLVE THE LIGHTS OUT GAME

David Nawrocki
Albright College
13th and Bern Streets
Reading, Pennsylvania 19612
dnawrocki@alb.edu

The Lights Out game is an electronic game consisting of a 5×5 grid of lighted buttons. Pressing one button will change the state of that button as well as the state of the neighboring buttons above and below and to the left and right. The game is begun with several of the twenty-five buttons light, and the object of the game is to find a sequence of button presses to turn all the lights off. To begin, we first consider the following simple Lights Out game consisting of a 2×2 grid of lights.

1	2
3	4

Figure 1 2×2 Lights Out Game

The buttons 1,2,3 and 4 are either lit or not (on or off) and they are connected as follows.

- Pressing button 1 changes its state as well as those of 2 and 3.
- Pressing button 2 changes its state as well as those of 1 and 4.
- Pressing button 3 changes its state as well as those of 1 and 4.
- Pressing button 4 changes its state as well as those of 2 and 3.

Given an initial state of the game in which some lights are on and some are off, the object of the game is to press a sequence of button to turn off all the lights. For example, suppose we are given the following initial state of the game as shown in Figure 2.

ON	ON
ON	OFF

.

Figure 2 Initial State of the 2×2 Lights Out Game

This initial state can be represented by the state vector $\vec{b} = (1\,1\,1\,0)^T$ and the solution would be to press button 1 which we represent with the action vector $\vec{x} = (1\,0\,0\,0)^T$. Here, 1 represents either the state of a button being on or the action of pressing a button and 0 represents either the state of a button being off or the action of not pressing a button.

We can obtain the adjacency matrix A in equation (1) for this game by considering its connected graph shown in Figure 3.

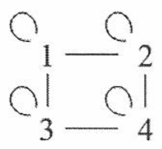

Figure 3 Connected Graph of the 2 x 2 Lights Out Game

$$A = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}. \tag{1}$$

Here, $A(i,j) = 1$ if button i is connected to button j and $A(i,j) = 0$ otherwise.

Suppose now that all the buttons are initially off and we press button 1. Then buttons 1,2, and 3 will turn on. This can be represented by matrix multiplication as follows.

$$A \cdot \vec{x} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \vec{b} \tag{2}$$

If we define a simple action to be that of pressing only a single button then we see that, starting with all the lights initially off, A multiplied by a simple action vector $\vec{x}$ gives the state of the game $\vec{b}$. Also, notice the following:

- If a button is off (state 0) and we press it (action 1) then we turn it on (state 1) so 0+1=1
- If a button is on (state 1) and we press it (action 1) then we turn it off (state 0) so 1+1=0
- If a button is off (state 0) and we do not press it (action 0) then it stays off (state 0) so 0+0=0
- If a button is on (state 1) and we do not press it (action 0) then it stays on (state 1) so 1+0=1.

So we see that for this game we are really performing arithmetic over $\mathbb{Z}_2$. Now, suppose we start with the game initially in the state $\vec{b} = (0\,0\,0\,0)^T$ (all the lights off) and we press buttons 1 and 2. If button 1 is pressed first then we arrive at the state $A \cdot (1\,0\,0\,0)^T = (1\,1\,1\,0)^T$. On the other hand, if button 2 were pressed first we would arrive at the state $A \cdot (0\,1\,0\,0)^T = (1\,1\,0\,1)^T$. To understand what happens when we press buttons 1 then 2 in that order, we need to realize that once button 1 is pressed buttons 1,2, and 3 are now in state 1 (on). So, if we evaluate $A \cdot (0\,1\,0\,0)^T$ (A times the action of pressing button 2) whatever states we get for buttons 1, 2, and 3 should be reversed. Since we are performing addition over $\mathbb{Z}_2$ this can be accomplished by adding the state vectors $(1\,1\,1\,0)^T + (1\,1\,0\,1)^T = (0\,0\,1\,1)^T$ which is the initial state of the game after pressing buttons 1 and 2. We now see that starting with the game initially off the action of pressing buttons 1 then 2 can be evaluated as follows,

$$A \cdot \left(\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \right) = A \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \tag{3}$$

This also demonstrates the fact that since vector addition is obviously commutative the order in which the buttons are pressed does not affect the final state of the game.

If our game has an initial state vector $\vec{b}$ and we want to find a sequence of button presses represented by the vector $\vec{x}$ to turn all the lights off then we need to solve the equation $\vec{b} + A\vec{x} = \vec{0}$ for the action vector $\vec{x}$. Since we are working in $\mathbb{Z}_2$ we really need to solve $\vec{b} + A\vec{x} \equiv \vec{0} \pmod 2$ for the vector $\vec{x}$. Adding $\vec{b}$ to both sides of the previous equation and noting that $\vec{b}+\vec{b} \equiv \vec{0} \pmod 2$ yields $A\vec{x} \equiv \vec{b}$ (mod 2). Another way to understand this is to realize that if we start the game with all the lights off (state $\vec{0}$) and we find a sequence of button presses $\vec{x}$ to change the game from state $\vec{0}$ to state $\vec{b}$ then the sequence of button presses $\vec{x}$ will also change the game back from state $\vec{b}$ to state $\vec{0}$. Therefore, solving $A\vec{x} \equiv \vec{b}$ (mod 2) for $\vec{x}$ will not only change the game from state $\vec{0}$ to state $\vec{b}$, but it will also allow us to change it from state $\vec{b}$ back to state $\vec{0}$. For example, if our simple 2×2 game were started with buttons 1, 2 and 3 lit then row reducing the augmented adjacency matrix over $\mathbb{Z}_2$ yields

$$\left(\begin{array}{cccc|c} 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{array} \right) \sim \left(\begin{array}{cccc|c} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{array} \right) \tag{4}$$

This indicates that our solution vector is $\vec{x} = (1\,0\,0\,0)^T$ or that we should press button 1 to turn all the lights off.

We now consider the more complicated Lights Out game consisting of a 5×5 grid of buttons. The connected graph for this game is shown in Figure 4. The 25×26 augmented adjacency matrix for the 5×5 Lights Out game is constructed in a similar manner as that given in (4). The Mathematica code used to row reduce the 25×26 augmented adjacency matrix over $\mathbb{Z}_2$ is shown below. The program is executed by inputting the initial state of the game into the 26^{th} column in the augmented adjacency matrix and running the program.

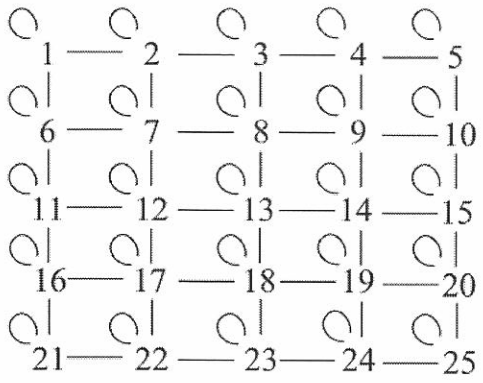

Figure 4 Connected Graph of the 5 × 5 Lights Out game

Mathematica Code Used to Perform Matrix Reduction over $\mathbb{Z}_2$

```
(*AugAdjacency is the 25 × 26 augmented adjacency matrix which is not shown here.*)
A = AugAdjacency;
r = 1; (* Fix the current row entry at 1 *)
c = 1; (* Fix the current column entry at 1 *)
found = 0; (* Used to determine is a non - zero entry has been found *)
(* Begin indexing down the rows. *)
For[r = 1, r <= 25, r++,
  found = 0;
  If[A[[r, c]] = = 1,
    found = 1;
    (* The following for loops will annihilate all non-zero entries above and below *)
    (* the current row and column position. *)
    For[i = 1, i <= 25, i++,
      If[A[[i, c]] = = 0 || i = = r, Continue[];];
      For[j = 1, j <= 26, j++,
        temp = A[[i, j]]; (* temp is a temporary holder for the value A[[i, j]] *)
        If [temp = = 1 && A[[r, j]] = = 1, A[[i, j]] = 0;]; (*note the arithmetic mod 2*)
        If [temp = = 1 && A[[r, j]] = = 0, A[[i, j]] = 1;];
        If [temp = = 0 && A[[r, j]] = = 1, A[[i, j]] = 1;];
        If [temp = = 0 && A[[r, j]] = = 0, A[[i, j]] = 0;];
        ]; (*end for j*)
    ] ;(*end for i*)
  ] ; (*end if*)
  (* If A[[r, c]] = = 0 then we should search for a row below the current one to swap *)
  If[A[[r, c]] = = 0,
    For[i = r + 1, i <= 25, i++,
      If[A[[i, c]] = = 1,
        found = 1;
        (* Swap the rows *)
        temp = A[[r]];
        A[[r]] = A[[i]];
        A[[i]] = temp;
        (*Annihilate non-zero entries above and below the current row and column*)
        For[i2 = 1, i2 <= 25, i2++,
          If[A[[i2, c]] = = 0 || i2 = = r, Continue[];];
          For[j = 1, j <= 26, j++,
            temp = A[[i2, j]];
            If [temp = = 1 && A[[r, j]] = = 1, A[[i2, j]] = 0;]; (*note the arithmetic mod 2*)
            If [temp = = 1 && A[[r, j]] = = 0, A[[i2, j]] = 1;];
            If [temp = = 0 && A[[r, j]] = = 1, A[[i2, j]] = 1;];
            If [temp = = 0 && A[[r, j]] = = 0, A[[i2, j]] = 0;];
          ]; (* end for j *)
        ];(* end for i2 *)
      ]; (* end if *)
    ]; (* end for i *)
  ];(* end if *)
  If[found = = 0, r = r - 1];
  If[r = = 25 || c = = 25, Break[];];
  c = c + 1; (* Increment the current column position *)
]; (* end for r *)
Print[MatrixForm[A]];
For[i = 1, i <= 25, i++,
    If[A[[i, 26]] = = 1, Solution[[i, 1]] = 1]; (* Write column 26 to the solution vector *)
]; (* end for i *)
Print[MatrixForm[Solution]];
(* The following is just a nice way of printing out the solution *)
solutionList = {0};
firstOne = 1;
For[i = 1, i <= 25, i++,
  If[A[[i, 26]] = = 1  && firstOne = = 0, solutionList = Append[solutionList, i]];
  If[A[[i, 26]] = = 1 && firstOne = = 1,
    solutionList = ReplacePart[solutionList, i, 1]; firstOne = 0;];
];(* end for i *)
Print[solutionList];
```

REFERENCES

[1] R. Trudeau, *Introduction to Graph Theory*, Dover Publications Inc. New York, 1994.

[2] R. Gould, *Graph Theory*, Benjamin-Cummings, New Jersey, 1988

[3] W. Winston, *Operations Research*, Duxbury Press, California, 1994