Proof. For $0 < q < 1$,

$$1 = \sum_{\sigma \in S_n} P(\sigma) = \sum_{\sigma \in S_n} \frac{q^{|\sigma|}}{D_n(q)} = \frac{1}{D_n(q)} \sum_{k=0}^{M(n)} \left\langle {n \atop k} \right\rangle q^k.$$

Thus, $D_n(q) = \sum_{k=0}^{M(n)} \left\langle {n \atop k} \right\rangle q^k$, for $0 < q < 1$. For polynomial functions, this can only happen if they are identical.

As mentioned earlier, we do not know of a simple formula for the numbers $\left\langle {n \atop k} \right\rangle$. It would also be interesting to know if these numbers arise in other contexts.

REFERENCE

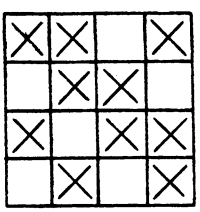
1. Curtis Cooper, Geometric series and a probability problem, *Amer. Math. Monthly* 93 (1986), 126–127.

The Game of Quatrainment

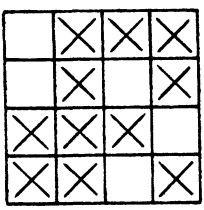
T. E. GANTNER
The University of Dayton
Dayton, OH 45469

The concept of a finite-state machine is important in a discrete mathematics course because of its role in the study of formal languages. Typical textbook examples often have just a few states and inputs: for example, parity check machines and binary adders. It is desirable to have a few more-complex examples to present to the student as supplementary material; at the same time, such examples should be on a fairly elementary level. One such example, Think-A-Dot, was discussed in detail in [1]. The example that we discuss here is the game of Quatrainment [2], which may be implemented on a variety of home computers, but which may equally well be played on ordinary graph paper. It may be completely analyzed using such elementary tools as: mathematical modeling, Boolean matrices, modulo 2 arithmetic, and algorithms. It is also interesting as an example of matrix theory over a finite field.

One begins playing Quatrainment by placing a random pattern of X's in each of two 4-by-4 grids called A and B. For example:



A



B

FIGURE 1

The object of the game is to modify the pattern of A, using a finite sequence of moves, so that it matches the pattern of B. Each move is made by selecting one of the 16 cells of A, which we label from 0 to 15, starting with the upper left corner, as indicated in FIGURE 2.

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

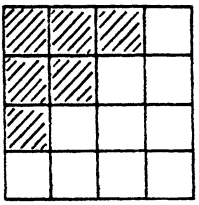
FIGURE 2

Each cell of A has one of two states: it is marked by an X, or else it is blank. Since there are 4 corner cells, 4 center cells, and 8 edge cells, there are three types of moves, which are defined by the following rules:

If a corner cell is selected, then reverse the states of the six cells in the triangle at that cell (FIGURE 3a).

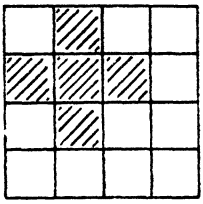
If a center cell is selected, then reverse its state as well as the states of its four neighbors (FIGURE 3b).

If an edge cell is selected, then reverse the states of its three neighbors (FIGURE 3c).



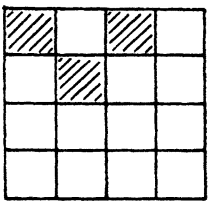
Choosing Cell 0

FIGURE 3a



Choosing Cell 5

FIGURE 3b



Choosing Cell 1

FIGURE 3c

In addition, we could suppose that the game is played on a computer and that the computer keeps track of the elapsed time as well as the number of moves. The task should be accomplished in minimum time using the least number of moves. We show below that there is always a unique solution using a minimum number of moves.

The Finite State Machine and Its Matrix Representation

As a finite state machine, each cell of A is either blank or is marked with an X, so A is capable of having 2^{16} distinct states. Each of these states may be represented as a 4-by-4 matrix of 0's and 1's; an entry 1 occupies the position of each cell that is marked with an X, and an entry 0 occupies the position of a blank cell. Thus the set of states is just the set $\mathcal{S}$ of 4-by-4 matrices with entries 0 or 1. The inputs of the finite state machine are the cell numbers $I = \{0, 1, 2, \ldots, 15\}$, and for each state S and

input $k \in I$, the next state $f_k(S)$ is the state derived from S by selecting the input cell $k \in I$. For example, if $k = 0$ and if S is the state A of FIGURE 1, then

$$S = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}, \quad \text{and} \quad f_0(S) = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}.$$

We may define output functions, if desired, to be identical with the next-state functions.

We further note that the next-state functions can be described in terms of matrix addition. For example, associated with the moves that correspond to cell choices 0, 5, 1 of FIGURE 3, we have matrices

$$M_0 = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad M_5 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad M_1 = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Similarly we have a matrix M_k for each $k \in I$. If we add the entries of matrices using modulo 2 arithmetic, then we have

$$f_k(S) = S + M_k, \quad \text{for all } k \in I \quad \text{and} \quad S \in \mathscr{S}.$$

Using the fact that matrix addition is both associative and commutative, two facts are now evident:

$$(f_k \circ f_k)(S) = S + M_k + M_k = S, \quad \text{so } f_k \circ f_k = id, \tag{1}$$

where id is the identity function on $\mathscr{S}$, and

$$(f_k \circ f_j)(S) = S + M_j + M_k = S + M_k + M_j = (f_j \circ f_k)(S), \quad \text{so } f_k \circ f_j = f_j \circ f_k. \tag{2}$$

Thus the semigroup I^* of the machine consisting of all finite input sequences is actually a commutative group in which each input element is its own inverse. Also, associated with each word $k_1 k_2 \cdots k_n \in I^*$ is the matrix

$$h(k_1 k_2 \cdots k_n) = M_{k_1} + M_{k_2} + \cdots + M_{k_n},$$

and this formula defines an isomorphism of I^* with the additive subgroup (or linear subspace) $\mathscr{M}$ of $\mathscr{S}$ that is generated by the selection matrices $M_0, M_1, \ldots, M_{15}$.

A Solution Algorithm

In terms of finite state machines, the object of Quatrainment may be rephrased as follows. Given two states A and B, determine a sequence $k_1 k_2 \ldots k_n$ of inputs such that $(f_{k_n} \circ \cdots \circ f_{k_2} \circ f_{k_1})(A) = B$. A solution algorithm exists if we can find a set of input sequences, each of which will change the state of just one chosen cell. By rotation and reflection, it suffices to obtain a sequence that will change the state of just one corner cell, center cell, or edge cell. For example, starting with the zero-matrix state, we can change the state of cell 0 by the sequence of moves indicated below in (3). Similarly, the sequence (4) will change the state of cell 5, and the sequence (5) will change the state of cell 1.

$$\begin{pmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{pmatrix} \xrightarrow{f_0} \begin{pmatrix}1&1&1&0\\1&1&0&0\\1&0&0&0\\0&0&0&0\end{pmatrix} \xrightarrow{f_5} \begin{pmatrix}1&0&1&0\\0&0&1&0\\1&1&0&0\\0&0&0&0\end{pmatrix} \xrightarrow{f_6} \begin{pmatrix}1&0&0&0\\0&1&0&1\\1&1&1&0\\0&0&0&0\end{pmatrix}$$

$$\xrightarrow{f_9} \begin{pmatrix}1&0&0&0\\0&0&0&1\\0&0&0&0\\0&1&0&0\end{pmatrix} \xrightarrow{f_{11}} \begin{pmatrix}1&0&0&0\\0&0&0&0\\0&0&1&0\\0&1&0&1\end{pmatrix} \xrightarrow{f_{14}} \begin{pmatrix}1&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{pmatrix} \tag{3}$$

$$\begin{pmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{pmatrix} \xrightarrow{f_3} \begin{pmatrix}0&1&1&1\\0&0&1&1\\0&0&0&1\\0&0&0&0\end{pmatrix} \xrightarrow{f_5} \begin{pmatrix}0&0&1&1\\1&1&0&1\\0&1&0&1\\0&0&0&0\end{pmatrix} \xrightarrow{f_6} \begin{pmatrix}0&0&0&1\\1&0&1&0\\0&1&1&1\\0&0&0&0\end{pmatrix}$$

$$\xrightarrow{f_7} \begin{pmatrix}0&0&0&0\\1&0&0&0\\0&1&1&0\\0&0&0&0\end{pmatrix} \xrightarrow{f_9} \begin{pmatrix}0&0&0&0\\1&1&0&0\\1&0&0&0\\0&1&0&0\end{pmatrix} \xrightarrow{f_{12}} \begin{pmatrix}0&0&0&0\\0&1&0&0\\0&1&0&0\\1&0&1&0\end{pmatrix} \tag{4}$$

$$\xrightarrow{f_{13}} \begin{pmatrix}0&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&0&0\end{pmatrix}$$

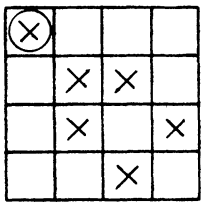
$$\begin{pmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{pmatrix} \xrightarrow{f_0} \begin{pmatrix}1&1&1&0\\1&1&0&0\\1&0&0&0\\0&0&0&0\end{pmatrix} \xrightarrow{f_2} \begin{pmatrix}1&0&1&1\\1&1&1&0\\1&0&0&0\\0&0&0&0\end{pmatrix} \xrightarrow{f_3} \begin{pmatrix}1&1&0&0\\1&1&0&1\\1&0&0&1\\0&0&0&0\end{pmatrix}$$

$$\xrightarrow{f_4} \begin{pmatrix}0&1&0&0\\1&0&0&1\\0&0&0&1\\0&0&0&0\end{pmatrix} \xrightarrow{f_8} \begin{pmatrix}0&1&0&0\\0&0&0&1\\0&1&0&1\\1&0&0&0\end{pmatrix} \xrightarrow{f_{13}} \begin{pmatrix}0&1&0&0\\0&0&0&1\\0&0&0&1\\0&0&1&0\end{pmatrix} \tag{5}$$

$$\xrightarrow{f_{14}} \begin{pmatrix}0&1&0&0\\0&0&0&1\\0&0&1&1\\0&1&1&1\end{pmatrix} \xrightarrow{f_{15}} \begin{pmatrix}0&1&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\end{pmatrix}$$

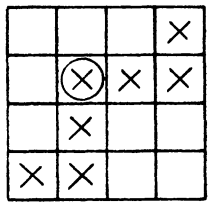
The above input sequences are most easily remembered in terms of the patterns indicated in FIGURE 4. If we select the six cells checked in FIGURE 4a, we change the state of only cell 0. Similarly, FIGURE 4b indicates the cell choices needed to change the state of only cell 5, and Figure 4c gives the choices needed to change only cell 1.

Thus a solution always exists. Simply change the state of each cell of A that does not match the corresponding cell of B by using a finite sequence of inputs as indicated by the above patterns, and the task is done. However, if we are required to change several cells of A, then in all likelihood some of the input choices will be made several times; but this is not necessary, because each input choice is its own group inverse.



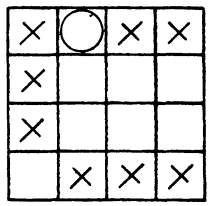
Change a Corner

FIGURE 4a



Change a Center

FIGURE 4b



Change an Edge

FIGURE 4c

Thus, each input need be used just once or not at all, and so there should be a solution which uses at most 16 moves.

A Minimal Algorithm

Let us look again at our initial example, where

$$A = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}, \quad \text{and} \quad B = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}.$$

Given two such states, we construct the matrix M according to the following rule: an entry of M is 0 if the corresponding entries of A and B are the same, otherwise it is 1. In other words, $M = A + B$. Since entry addition is modulo 2, this means that $A + M = B$. In our example:

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

The question now is this: Do there exist distinct inputs $0 \leqslant k_1 < k_2 < \cdots < k_n \leqslant 15$ such that $M = M_{k_1} + M_{k_2} + \cdots + M_{k_n}$? In the terminology of vector spaces, do $M_0, M_1, \ldots, M_{15}$ form a basis for $\mathscr{S}$? The answer is yes. Above we saw how to express each of the standard basis matrices $E_0, E_1, \ldots, E_{15}$ in terms of $M_0, M_1, \ldots, M_{15}$, where E_k has a 1 in cell k and all of its other entries are zero. Thus $M_0, M_1, \ldots, M_{15}$ generate all of $\mathscr{S}$; and the linear dimension of $\mathscr{S}$ over the field of integers modulo 2 is 16, so they form a basis for $\mathscr{S}$. Since they do form a basis, each such $M \in \mathscr{S}$ is expressible *uniquely* as a sum of at most 16 of the selection matrices $M_0, M_1, \ldots, M_{15}$. Thus there is a unique solution having a minimum number of moves. In the above example, it is not difficult to verify that

$$M = M_0 + M_1 + M_3 + M_4 + M_7 + M_9 + M_{10} + M_{12}.$$

Unfortunately, from a practical point of view, in order to express an M as a unique sum of the matrices $M_0, M_1, \ldots, M_{15}$, one must either solve a 16-by-16 linear system, or one must make a chart like those of FIGURE 4 for each entry of A that must be changed and then determine from these charts which cell choices must be made an odd number of times. Both of these procedures are time consuming; keep in mind that the object of the game is to do the task in as few moves as possible *and* in minimal time. Therefore, it is still a challenging game for humans.

Some Modifications

We may try to play games with the game itself either by generalizing it to a larger (even higher-dimensional) grid, or by modifying the effect of the various cell choices. If we keep the same grid, there are two obvious cell choice modifications that may be made:

(a) Make an edge cell selection reverse the states of the selected cell as well as its three neighbors; for example, for cell 1 we would have

$$M_1 = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{instead of } M_1 = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix};$$

(b) Make the corner and center cell selections behave in the chess-board way that the edge cell selections behave, that is, do not change the states of two neighboring cells.

For example, for cells 0 and 5, we would have

$$M_0 = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{instead of } M_0 = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

and

$$M_5 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{instead of } M_5 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Unfortunately neither of these modified games is solvable in general because in neither case is the subspace $\mathscr{M}$ of $\mathscr{S}$ equal to all of $\mathscr{S}$. Thus there will exist states A and B so that $A + B = M \notin \mathscr{M}$, hence M will not be obtainable by any finite sequence of cell choices; it is a nice exercise for the reader to find such states. Here are the reasons for the claim.

In modification (a), it is easily verified that

$$M_0 + M_3 = M_4 + M_7, \qquad M_0 + M_{12} = M_1 + M_{13},$$
$$M_3 + M_{15} = M_2 + M_{14}, \qquad M_{12} + M_{15} = M_8 + M_{11},$$

so that $\mathscr{M}$ is generated by the 12 matrices $M_2, M_3, \ldots, M_{12}, M_{13}$; and these are linearly independent, so the dimension of $\mathscr{M}$ is 12.

In modification (b), it is easily seen that

$$M_5 + M_{10} = M_2 + M_7 + M_8 + M_{13},$$
$$M_6 + M_9 = M_1 + M_4 + M_{11} + M_{14},$$

so, for example, M_5 and M_6 can be eliminated to obtain a basis for $\mathscr{M}$; in this case the dimension of $\mathscr{M}$ is 14.

REFERENCES

1. John A. Beidler, Think-a-dot revisited, this MAGAZINE 46 (May-June, 1973), 128–136.
2. Sean Puckett, Quatrainment, *COMPUTE! Magazine* 6 (February, 1984), 76–91.