April 2023 PonderThis solution

Daniel Copeland and Raphael Deem

Portland, OR, USA

24x24 Solution, Number of moves = 280

[(2, 2), (13, 24), (24, 15), (2, 6), (1, 21), (17, 3), (1, 24), (5, 7), (4, 17), (2, 13), (13, 19), (19, 15), (22, 5), (17, 13), (7, 12), (11, 9), (9, 22), (13, 2), (12, 17), (15, 5), (1, 15), (4, 7), (1, 12), (23, 2), (24, 1), (18, 13), (17, 17), (11, 22), (10, 11), (12, 21), (14, 23), (15, 7), (5, 1), (21, 19), (19, 3), (14, 17), (17, 14), (24, 11), (2, 10), (11, 18), (18, 21), (23, 12), (1, 7), (24, 10), (1, 17), (19, 6), (11, 20), (18, 3), (14, 14), (20, 16), (23, 19), (3, 4), (4, 11), (5, 12), (23, 4), (3, 2), (24, 16), (5, 17), (19, 14), (12, 10), (14, 11), (11, 3), (4, 6), (15, 24), (1, 22), (22, 23), (18, 19), (14, 15), (7, 24), (12, 3), (23, 6), (21, 1), (3, 23), (24, 22), (22, 17), (8, 18), (19, 9), (1, 14), (16, 2), (2, 5), (18, 4), (13, 7), (4, 21), (20, 4), (19, 21), (6, 9), (1, 20), (7, 1), (16, 17), (24, 13), (2, 14), (23, 5), (12, 24), (3, 15), (18, 23), (10, 7), (21, 18), (4, 22), (22, 12), (8, 2), (6, 21), (22, 20), (20, 13), (2, 18), (16, 22), (9, 24), (3, 17), (11, 16), (4, 2), (10, 4), (19, 1), (7, 19), (1, 8), (23, 11), (15, 12), (12, 5), (24, 14), (5, 23), (3, 16), (4, 5), (22, 21), (9, 20), (11, 24), (16, 1), (24, 4), (7, 6), (20, 2), (1, 23), (12, 14), (23, 3), (15, 18), (14, 13), (5, 19), (13, 22), (6, 8), (10, 17), (3, 1), (17, 23), (8, 19), (4, 13), (14, 2), (6, 3), (10, 16), (22, 7), (20, 17), (24, 6), (11, 4), (5, 5), (12, 22), (21, 10), (15, 21), (18, 24), (1, 9), (16, 20), (16, 23), (10, 13), (12, 9), (11, 5), (15, 11), (8, 12), (20, 1), (2, 16), (22, 19), (5, 21), (3, 22), (14, 10), (21, 20), (6, 6), (19, 7), (4, 4), (18, 17), (1, 3), (4, 8), (9, 12), (16, 21), (13, 17), (7, 18), (20, 20), (19, 23), (23, 16), (10, 3), (21, 6), (22, 22), (11, 13), (6, 14), (2, 7), (1, 5), (12, 19), (5, 11), (14, 1), (15, 4), (18, 15), (15, 2), (4, 1), (6, 20), (17, 12), (1, 19), (5, 4), (24, 3), (20, 14), (22, 18), (19, 17), (14, 21), (7, 13), (9, 6), (11, 23), (10, 8), (21, 24), (23, 22), (12, 7), (2, 15), (18, 5), (13, 16), (7, 10), (10, 20), (8, 21), (20, 6), (23, 24), (13, 18), (3, 14), (19, 5), (18, 22), (9, 1), (17, 15), (11, 8), (24, 23), (2, 17), (4, 19), (1, 4), (5, 2), (22, 16), (15, 3), (21, 7), (6, 13), (12, 12), (4, 12), (3, 7), (12, 11), (6, 1), (22, 14), (24, 21), (1, 13), (8, 4), (17, 19), (7, 5), (5, 20), (15, 6), (16, 8), (2, 22), (18, 16), (20, 24), (19, 10), (9, 3), (21, 15), (13, 23), (11, 2), (10, 18), (15, 23), (9, 10), (5, 8), (2, 20), (4, 15), (12, 18), (19, 24), (20, 19), (6, 11), (10, 9), (13, 3), (24, 12), (7, 7), (11, 21), (22, 2), (17, 4), (1, 1), (14, 5), (16, 6), (18, 14), (8, 22), (3, 13), (21, 16)]

30x30 Solution, Number of moves = 429

[(29, 19), (11, 26), (17, 17), (5, 19), (6, 18), (11, 7), (3, 11), (25, 2), (10, 7), (3, 13), (17, 19), (6, 5), (25, 19), (3, 3), (15, 5), (6, 9), (16, 2), (10, 4), (29, 25), (15, 4), (26, 3), (29, 15), (16, 19), (18, 11), (6, 30), (11, 29), (10, 2), (30, 23), (4, 22), (2, 30), (8, 12), (28, 23), (27, 1), (30, 26), (29, 11), (16, 13), (11, 10), (17, 4), (1, 28), (14, 2), (4, 7), (23, 3), (24, 12), (1, 9), (30, 8), (20, 5), (27, 10), (5, 17), (29, 28), (23, 7), (7, 26), (17, 18), (2, 2), (21, 13), (16, 27), (11, 23), (3, 24), (22, 4), (25, 30), (6, 10), (15, 7), (13, 26), (21, 14), (29, 18), (22, 17), (5, 8), (10, 30), (18, 4), (24, 20), (17, 23), (19, 24), (23, 2), (12, 5), (16, 29), (3, 28), (26, 19), (1, 22), (7, 21), (13, 20), (26, 5), (23, 18), (7, 29), (12, 25), (1, 8), (6, 11), (10, 14), (17, 22), (29, 24), (21, 2), (28, 3), (16, 30), (22, 10), (14, 4), (4, 28), (24, 19), (5, 7), (19, 23), (25, 10), (13, 16), (24, 14), (29, 27), (6, 23), (27, 19), (10, 13), (9, 12), (14, 6), (19, 8), (1, 30), (4, 5), (26, 29), (20, 24), (22, 15), (5, 2), (23, 25), (7, 9), (12, 3), (30, 4), (16, 18), (28, 11), (11, 22), (26, 26), (6, 6), (5, 1), (20, 25), (25, 12), (23, 30), (24, 13), (9, 11), (12, 24), (16, 4), (8, 15), (1, 27), (28, 21), (18, 5), (27, 3), (13, 23), (11, 8), (4, 18), (7, 22), (22, 16), (30, 2), (19, 19), (29, 14), (14, 29), (4, 23), (19, 15), (25, 6), (28, 29), (23, 20), (24, 4), (12, 21), (16, 22), (13, 18), (29, 13), (26, 24), (2, 8), (22, 12), (6, 2), (18, 28), (14, 7), (9, 26), (21, 19), (8, 25), (1, 3), (11, 30), (27, 11), (5, 5), (11, 20), (26, 13), (14, 8), (3, 18), (19, 12), (12, 28), (23, 10), (10, 22), (25, 5), (24, 25), (1, 11), (9, 3), (18, 7), (16, 9), (13, 2), (27, 15), (22, 30), (15, 29), (21, 6), (2, 23), (29, 21), (5, 24), (2, 28), (19, 11), (22, 1), (23, 16), (25, 20), (15, 24), (13, 22), (17, 9), (16, 7), (7, 6), (3, 27), (30, 30), (27, 12), (21, 8), (11, 25), (10, 5), (29, 23), (9, 29), (26, 10), (12, 14), (5, 21), (24, 3), (14, 15), (1, 18), (18, 13), (1, 23), (19, 22), (26, 30), (17, 5), (3, 15), (11, 28), (21, 27), (10, 17), (24, 7), (6, 1), (28, 18), (13, 11), (2, 29), (18, 8), (16, 6), (27, 24), (7, 10), (25, 14), (30, 3), (23, 4), (9, 21), (12, 20), (29, 16), (14, 9), (22, 25), (4, 12), (5, 26), (20, 6), (28, 28), (12, 2), (3, 1), (26, 12), (30, 22), (6, 3), (29, 7), (9, 14), (4, 15), (1, 4), (11, 17), (17, 11), (27, 8), (23, 29), (2, 27), (25, 16), (19, 20), (22, 21), (7, 5), (10, 25), (18, 9), (21, 30), (5, 18), (13, 24), (12, 17), (3, 8), (19, 30), (30, 21), (6, 27), (26, 16), (8, 20), (5, 6), (27, 28), (1, 5), (15, 25), (21, 11), (28, 14), (10, 19), (2, 22), (18, 3), (25, 24), (11, 18), (13, 4), (23, 15), (7, 12), (22, 13), (29, 1), (9, 2), (4, 9), (20, 7), (18, 14), (13, 30), (29, 10), (14, 27), (25, 28), (23, 23), (7, 20), (4, 4), (3, 17), (30, 11), (20, 26), (22, 5), (10, 21), (1, 13), (24, 2), (15, 18), (6, 19), (16, 15), (28, 24), (12, 8), (11, 12), (9, 16), (8, 3), (21, 25), (27, 6), (5, 9), (26, 22), (28, 4), (30, 25), (9, 9), (3, 5), (13, 10), (10, 26), (17, 20), (6, 13), (29, 12), (27, 16), (14, 14), (1, 17), (21, 21), (25, 15), (7, 3), (16, 11), (11, 2), (18, 6), (23, 8), (15, 27), (4, 30), (12, 19), (26, 18), (24, 24), (22, 22), (8, 23), (14, 18), (27, 14), (25, 4), (15, 19), (4, 13), (26, 21), (16, 3), (17, 15), (21, 16), (24, 6), (11, 24), (3, 23), (12, 11), (2, 25), (23, 9), (18, 2), (22, 8), (10, 12), (13, 7), (30, 17), (8, 27), (6, 20), (9, 10), (7, 30), (19, 5), (28, 1), (29, 22), (17, 16), (6, 28), (20, 12), (23, 14), (16, 5), (28, 22), (13, 3), (24, 27), (14, 13), (25, 8), (19, 21), (4, 25), (21, 29), (22, 24), (15, 20), (30, 7), (11, 6), (10, 23), (12, 1), (2, 4), (18, 10), (29, 30), (9, 17), (27, 9), (5, 11), (8, 19), (26, 15), (7, 18), (17, 26), (16, 25), (8, 14), (11, 1), (1, 15), (9, 20), (6, 7), (2, 19), (5, 4), (7, 24), (19, 27), (24, 17), (4, 16), (20, 8), (27, 30), (13, 28), (15, 23), (21, 10), (28, 12), (10, 6), (25, 21), (30, 29), (26, 11), (14, 22), (12, 13), (18, 18), (29, 9), (22, 3), (23, 5)]

Solution Sketch

We used Python and the packages numpy and networkx.

Notation

We represent the state of the grid as an $n \times n$ matrix with entries in $\mathbb{F}_2 = \{0, 1\}$ (the field with 2 elements). If the state of the grid is G (an $n \times n$ matrix), then picking a bulb at coordinate (i, j) modifies the grid by a matrix addition mod 2:

$$G_{new} = G + M_{ij}$$

where M_{ij} is a matrix with 1's in the ith row and jth column and 0's everywhere else:

In [ ]:
```
def make_ij_mat(n, i, j):
    arr = np.array([[1 if a ==i or b == j else 0 for b in range(n)] for a in range(n)])
    return arr
```

Warning In the above paragraph, and below, we use numpy matrix indexing (start at 0 in the upper left corner). However, the solutions use the indexing from the original problem statement. It is easy to convert between indexing formats:

In [ ]:
```
def convert_coords(sols, n):
    return [(1 + coord[1], n - coord[0]) for coord in sols]
```

Outline

Our strategy worked in two stages.

Step 1: Find an unordered set of moves that could solve the game.

Step 2: Find a legal ordering of those moves.

To solve Step 1, we relaxed the constraints by letting the solver choose "on" bulbs at any time step (in addition to "off" bulbs). This simplifies the problem drastically, since the order of moves no longer matters. Furthermore, since each "move" consists of matrix addition mod 2, doing a move twice is the same as doing nothing. Finding a solution turns out to be a linear problem over $\mathbb{F}_2$ and is easily solved (especially pleasing is that the matrix we need to invert to solve the linear system is an involution, due to the parity of $n \in \{24, 30\}$, so it is not even necessary to invert a matrix). Furthermore, this approach actually proves that 280 (resp. 429) moves are required to solve the 24×24 (resp. 30×30) grid.

In [ ]:
```
# Step 1: unordered solve



def unordered_solve(grid):
    # Takes the grid and outputs list of coordinates (in arbitrary order) that may potentially be rearranged to a solution
    n = grid.shape[0]
    N = n**2

    def make_A_matrix(n):
    N = n**2
    A = np.array([[1 if ((int(i/n) == int(j/n)) or (i%n == j%n)) else 0 for j in range(N)] for i in range(N)])
    return A

    A = make_A_matrix(n)

    b = np.array([(1 + grid[int(i/n)][i%n]) %2 for i in range(N)])
    x = np.matmul(A,b)%2 # usually to solve for x we would use A^-1, but A^2 = 1 when n is even!

    unordered_sols = [(int(i/n), i%n) for i in range(N) if x[i]==1]

    return unordered_sols
```

The first step provides us with a set of distinct coordinates that characterize a sequence of winning moves, but we have to find an ordering such that every move is legal.

To solve Step 2, we use a greedy algorithm using some graph theory intuition and the package networkx. We try to find an ordering by iteratively finding a maximal set of disconnected legal moves amongst the ones we haven't made, then making these moves in any order. The subroutine we use from networkx (finding maximal independent sets) is sufficiently random that running a few hundred trials of the greedy algorithm resulted in a legal ordering for the 30x30. It only required 3 trials to find a legal ordering for the 24x24. The pseudocode for one trial is as follows:

```
# Step 2: reorder to get a sequence of legal moves


greedy_solve(unordered_sols, start_grid):

    current_grid = start_grid
    ordered_sols = []

    while unordered_sols is not empty:
        H = graph whose vertices consist of those moves in unordered_sols which are legal in current_grid
                and two moves are connected by an edge if they are in the same row or column
        H0 = maximal independent set of H
        ordered_sols += H0
        unordered_sols -= H0
        current_grid += sum( make_ij_mat(n, i, j) for (i,j) in H0)
```

One final twist that enabled this algorithm to work quickly is that we work backwards from the all-ones grid. In other words, the start_grid is given by the all-ones grid, and now the definition of "legal move" is changed to "bulb must be on". Step 1 guarantees that if we are able to make all the unordered moves, then the grid will be in the original starting position. We find it interesting that this modification worked!

In [ ]:
```
```