Square Game

Part A

Whether or not a board state is optimal (i.e.; whether the state, if left after a player's turn, guarantees that player can win), can be determined using the following algorithm:

1.) Write the number of squares remaining in each row as a binary integer

2.) Compute the bitwise XOR of all of these binary counts

3.) If the result has only zeroes, the board state is optimal. If not, the board state is not optimal.

Steps 1, 2 and 3 are equivalent to considering each bit individually, (1s, 2s, 4s, etc.), summing the ones in every column corresponding to that bit, and checking if the result is divisible by 2. If this is true for each bit, the board is optimal.

Proof:

For this to be an optimal strategy, it must be true that ① every possible move on an optimal board leads to a suboptimal board, and ② there is always a move on a suboptimal board that leads to an optimal board.

By a "backwards" induction, we know that ① and ② imply that a player who leaves the board in an optimal state can always win.

Inductive step: if, with n total turns remaining, a player leaves the board in an optimal state, then with $n-2$ total turns remaining they can leave it in an optimal state. This is true because, if player 1 leaves a board in an optimal state with n turns remaining, player 2 must leave it in a suboptimal state with $n-1$, by ①, so player 1 will once again be able to leave it in an optimal state with n turns remaining by ②. ✓

Base case: the winning board is the board with 0 for every row count, which is evidently optimal. Since only player 1 can leave the board in an optimal state, player 1 wins. ✓

Prove ①: every move on an optimal board leads to a suboptimal board.

In making a deletion a player must remove at least one square from the board. Since no two quantities are represented by the same binary number, then, any turn must result in at least one 0 being flipped to a 1, or 1 being flipped to a 0.

Now, consider a turn on an optimal board. Since only one row is changed, and at least one bit must be flipped in that row, the sum of all of the ones corresponding to that bit's "column" (the 1s bits from each row, 2s bits from each row, etc.) must be either one more or one less than the previous sum, S_n.

Since the board was previously optimal, it must be true that $2 | S_n$ so S is even. So, since -1 and 1 are both odd, and an odd number plus an even number is odd, adding or taking away a 1 will necessarily produce an odd sum ($2 \nmid S_{n+1}$) Since 2 must divide every sum for a board to be optimal, then, there exists no move on an optimal board that can yield an optimal board. ✓

Prove ②: Every suboptimal board has a move that can lead to an optimal board.

For every suboptimal board, the bitwise XOR of that board's rows must be nonzero, e.g., 00001011. Designate this value X_i

By the nature of bitwise XOR, the fact that the most significant 1 of X_i is, in fact, 1 implies that the corresponding bit of at least one of the rows must have been 1; If all of the rows had 0 for that bit, the resulting bit would have been 0. designate this row as R_k

By transforming R_k into the bitwise XOR of R_k with X_i *

then the new bitwise XOR, X_{i+1}, of all rows will be 0, an optimal board.

(cont.)

This is true because, for a board with m rows: (Bitwise XOR: ^)

$$X_{i+1} = (R_k \wedge X_i) \wedge (R_1 \wedge R_2 \wedge R_3 \wedge \ldots \wedge R_{k-1} \wedge R_{k+1} \wedge \ldots \wedge R_m)$$
$$X_{i+1} = X_i \wedge R_k \wedge (R_1 \wedge R_2 \wedge R_3 \wedge \ldots \wedge R_{k-1} \wedge R_{k+1} \wedge \ldots \wedge R_m)$$
$$X_{i+1} = X_i \wedge (R_1 \wedge R_2 \wedge \ldots \wedge R_m)$$
$$X_{i+1} = X_i \wedge X_i$$
$X_{i+1} = 0$, because the bitwise XOR of two identical values is always 0.

* Finally, we must conclude that R_k can always be transformed into $(R_k \wedge X_i)$.
Since the most significant 1 of X_i is also 1 in R_k, this bit alongside all of its preceding bits must be 0. Thus, if this 1 is at the nth bit (starting at $n=0$, it follows that $R_k \geq 2^n$ and $(R_k \wedge X_i) < 2^n$, so $R_k > (R_k \wedge X_i)$.
So, transforming R_k to $(R_k \wedge X_i)$ is achievable if and only if some number is subtracted from R_k. ✓

Since $X_{i+1} = 0$ after this transformation, it follows that it is always possible to achieve an optimal board by subtracting some positive integer from a single row R. Because a turn is defined as removing at least one square from a single row, this implies that an optimal board can always be achieved by taking a turn on a suboptimal board. ✓

Since ① and ② both hold true, it then follows that a player who leaves a board in an optimal state can always win.

Part B

The procedure for a turn is as follows:

- For each row count, R_i:
  - For each integer n from 1 to R_i:
    - Create a temporary board with n squares removed from R_i, and all other rows left the same
    - Fold all of the row counts of the temporary board together over the binary XOR operation
      - If the result is 0, remove n squares from R_i of the actual board, and return
      - Otherwise, continue iterating
- If no turn was taken, choose a random row with at least 1 square remaining, and remove 1 square from that row.