Problem 2: Dungeon

(Proposed by Lucian Bicsi. Primarily prepared by Teodor-Gabrial Tulba-Lecu. We thank Tamio-Vesa Nakajima for this editorial.)

To solve this problem, first note that the dungeon explorer's "mental state" consists of:

- The current position relative to the starting position.
- The set of possible starting positions.

Thus initially the state consists of position $(+0, +0)$ relative to the starting position, and the set $\mathcal{S} = \{1, \ldots, S\}$ of starting position (represented by their indices).

Observe that if we know that we started at some *subset* of starting positions $\mathcal{S}'$, then we will never go to a relative position that corresponds to a mine relative to *any* starting position from $\mathcal{S}'$. This is because doing this would risk getting 0 coins. Other than this we can visit any relative position.

Thus, we can imagine the following algorithm for the explorer:

- Consider the current set of starting positions that we could have started at $\mathcal{S}'$.
- Visit all possible squares that do not require us to visit a relative position that could correspond to a bomb for any of the starting positions in $\mathcal{S}'$.
- Check if we see any wall or coin that only appears in some *proper* subset of starting positions $\mathcal{S}'' \subset \mathcal{S}'$.
- If such a position exists, continue the search from that subset.
- Otherwise give up with the coins we can collect at this point.

How can we simulate this algorithm in our case? It is not difficult to make a version that does $O(NMS)$ complexity for each starting set (simply do a breadth first search, checking if a position is visitable, or respectively is a wall or a coin that only appears for some starting positions, by iterating over the current starting positions). To optimise this to use $O(NM)$ time for each set of starting positions that we check, store the map "relative to each starting position" in a bit mask. Thus we will have several matrices `wall`/`bomb`/`coin` that, at position (i, j) will contain a bit mask that have bit k equal to 1 if and only if position (i, j) relative to starting position k contains a wall/bomb/coin. This allows us to check (a) if a certain relative position contains a bomb for a subset of starting positions, and (b) if a certain relative position contains a wall/coin in some starting positions but not others. These can be done by representing the set of starting positions with a bit mask, and the using a bitwise "and" operation. Then for (a) we check if the result is nonzero, and for (b) we check if it is neither zero or equal to the bit mask that represents the set of starting positions.

Thus with this approach we can find, in $O(NM)$, for any set of starting positions:

- The number of coins we can safely collect.
- If any wall or coin is visible for only some subset of starting positions.
- What that subset of starting positions is.

Now suppose $f(\mathcal{S})$ gives us the result for some set of starting positions $\mathcal{S}$. The full result will be $f(\{1, \ldots, S\})$. To compute $f(\mathcal{S})$ use the following algorithm:

- Check if some coin or wall is visible only in some subset $\mathcal{S}' \subset \mathcal{S}$.
- If so, then return $\min(f(\mathcal{S}'), f(\mathcal{S} - \mathcal{S}'))$.
- Otherwise return the number of coins safely collectable if we would start at $\mathcal{S}$.

It can be proved that this does at most S matrix traversals. Thus the final complexity is $O(NMS)$.