Describe and analyze an efficient algorithm that either returns the minimum number of moves required to solve a given number maze, or correctly reports that the maze has no solution. Your running time should be in terms of $n$.

**Solution:** We'll index grid squares using pairs $(i, j)$ where $1 \leq i \leq n$ and $1 \leq j \leq n$. The upper left corner is $(1, 1)$ and the lower right corner is $(n, n)$. For the algorithm, we will build a configuration graph $G = (V, E)$. Vertex set $V$ is the set of grid squares $(i, j)$. There is an edge $(i, j) \rightarrow (i', j')$ for each pair of squares $(i, j)$ and $(i', j')$ where you can reach $(i', j')$ from $(i, j)$ in a single move (for example, if $k$ appears at $(i, j)$ and $i' = i$, $j' = j + k$, then there is an edge $(i, j) \rightarrow (i', j')$). There are $n^2$ vertices and at most $4n^2$ edges, and we can add these to an adjacency list representation of $G$ in constant time each, so it takes $O(n^2)$ time to build $G$.

After building $G$, we will run a breadth-first search with parent pointers from $(1, 1)$. One way to do so is to use the TRAVERSE procedure shown in lecture with a queue for the bag. If the search does not mark $(n, n)$, then we report the maze has no solution. Otherwise, we count the edges in the path $\text{parent}((n, n)), \text{parent}(\text{parent}((n, n))), \ldots$ leading back to $s$ and return that count.

To show correctness, we observe that any sequence of moves in the maze corresponds to a sequence of directed edges in $G$ with consecutive edges sharing a vertex (i.e., a walk). The same holds in the other direction; we can take a walk in $G$ and each edge of the walk is a move in the maze. The breadth-first search marks $(n, n)$ if and only if it is reachable in $G$ from $(1, 1)$ or there is a sequence of moves to reach the lower-right corner in the maze. So the algorithm is correct in cases it claims there is no solution. In other cases, the path found by the breadth-first search is the shortest path from $(1, 1)$, meaning the corresponding set of moves in the maze is as short as possible.

The running time of a breadth-first search is $O(E)$. Since $E = O(n^2)$, the running time of the whole algorithm is $O(n^2)$.

**Rubric:** 10 points total: 5 points for the algorithm, -2 if the algorithm fails to check for no solution, -3 if the algorithm fails to compute the minimum number of moves; 3 points for the proof; 2 points for the analysis.
Describe and analyze an algorithm to compute the length of the maximum-length monotonically increasing path with vertices in S. Assume you have a subroutine \( \text{LENGTH}(x, y, x', y') \) that returns the length of the segment from \((x, y)\) to \((x', y')\).

**Solution:** We will build a directed graph \( G = (V, E) \) where \( V = S \) is the set of \( n \) points and there is an edge \( a \rightarrow b \) if \( X[a] < X[b] \) and \( Y[a] < Y[b] \). Each edge \( a \rightarrow b \) is given the length \( \text{LENGTH}(x, y, x', y') \). We will argue later that \( G \) is a directed acyclic graph. There are \( n \) vertices in \( G \). We spend \( O(n^2) \) time checking each pair \( a, b \in S \) to see if edge \( a \rightarrow b \) exists, and if so, adding it to an adjacency list representation of \( G \). The time spend building \( G \) is \( O(n^2) \). Finally, we compute the length of the maximum-length directed path in \( G \) in \( O(V + E) = O(n^2) \) time and return that length. The total running time is \( O(n^2) \).

Edges correspond exactly to line segments that can appear in a monotonically increasing path with vertices in \( S \), so the paths in \( G \) correspond to monotonically increasing paths with vertices in \( S \). Further, the lengths of the edges are set so the lengths of these paths are the same in both \( G \) and in the plane. Finally, we can use the longest path algorithm, because \( G \) is a directed acyclic graph. Suppose to the contrary that \( G \) has a cycle \( a, b, \ldots, a \). The \( x \)-coordinates of the vertices along any walk in \( G \) are strictly increasing, but then point \( a \) cannot appear at the end of a non-trivial walk that starts with \( a \).

**Rubric:** 10 points total: 5 points for the algorithm; 3 points for the proof; 2 points for the analysis.