Documentation for searchMatrix Method

Overview

The searchMatrix method is designed to search for a target integer in a 2D matrix that has the following properties:

1. Each row is sorted in non-decreasing order.
2. The first integer of each row is greater than the last integer of the previous row.

The goal is to determine whether the target integer exists within the matrix using an efficient algorithm with a time complexity of O(log(m * n)).

Problem Statement

Given an m x n integer matrix matrix with the properties mentioned above, and an integer target, return true if target is in the matrix or false otherwise.

Input

- matrix: A list of lists of integers, where matrix[i][j] is the element at row i and column j.
- target: An integer to be searched within the matrix.

Output

- True if target is found in the matrix.
- False if target is not found in the matrix.

Constraints

- m == matrix.length
- n == matrix[i].length
- 1 <= m, n <= 100
- -10^4 <= matrix[i][j], target <= 10^4

Solution

The solution employs a binary search algorithm to achieve the required time complexity of O(log(m * n)). Here's a detailed step-by-step explanation:

Explanation

1. **Check for Empty Matrix:** The method first checks if the matrix or the first row of the matrix is empty. If either is empty, the method returns False.

2. **Matrix Dimensions:** The dimensions m (number of rows) and n (number of columns) are determined.

3. **Binary Search Initialization:** Two pointers, left and right, are initialized to represent the range of indices in the matrix, starting from 0 to m * n - 1.

4. **Binary Search Loop:** The main loop performs binary search:

- The middle index mid is calculated as the average of left and right.
- The value at the mid index is found by converting mid to 2D coordinates using matrix[mid // n][mid % n].
- If mid_value equals target, the method returns True.
- If mid_value is less than target, the search continues in the right half by updating left.
- If mid_value is greater than target, the search continues in the left half by updating right.

5. **Target Not Found:** If the loop terminates without finding the target, the method returns False.

Example Usage

```
solution = Solution()

matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]]


target = 3

print(solution.searchMatrix(matrix, target))  # Output: True


target = 13

print(solution.searchMatrix(matrix, target))  # Output: False
```

In these examples:

- The target 3 is present in the matrix, so the method returns True.
- The target 13 is not present in the matrix, so the method returns False.

This method efficiently searches the matrix using binary search, ensuring a logarithmic time complexity relative to the total number of elements in the matrix.