Problem 240: Search a 2D Matrix II

Problem Information

Difficulty: Medium
Acceptance Rate: 56.31%
Paid Only: No
Tags: Array, Binary Search, Divide and Conquer, Matrix

Problem Description

Write an efficient algorithm that searches for a value `target` in an `m x n` integer matrix `matrix`. This matrix has the following properties:

* Integers in each row are sorted in ascending from left to right. * Integers in each column are sorted in ascending from top to bottom.

Example 1:

![](https://assets.leetcode.com/uploads/2020/11/24/searchgrid2.jpg)

Input: matrix = [[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[10,13,14,17,24],[18,21,23,26,30]], target = 5 **Output:** true

Example 2:

![](https://assets.leetcode.com/uploads/2020/11/24/searchgrid.jpg)

Input: matrix = [[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[10,13,14,17,24],[18,21,23,26,30]], target = 20 **Output:** false

Constraints:

* `m == matrix.length` * `n == matrix[i].length` * `1 <= n, m <= 300` * `-109 <= matrix[i][j] <= 109` * All the integers in each row are **sorted** in ascending order. * All the integers in each column are **sorted** in ascending order. * `-109 <= target <= 109`

Code Snippets

C++:

```
class Solution {
public:
bool searchMatrix(vector<vector<int>>& matrix, int target) {

}
};
```

Java:

```
class Solution {
public boolean searchMatrix(int[][] matrix, int target) {

}
}
```

Python3:

```
class Solution:
def searchMatrix(self, matrix: List[List[int]], target: int) -> bool:
```