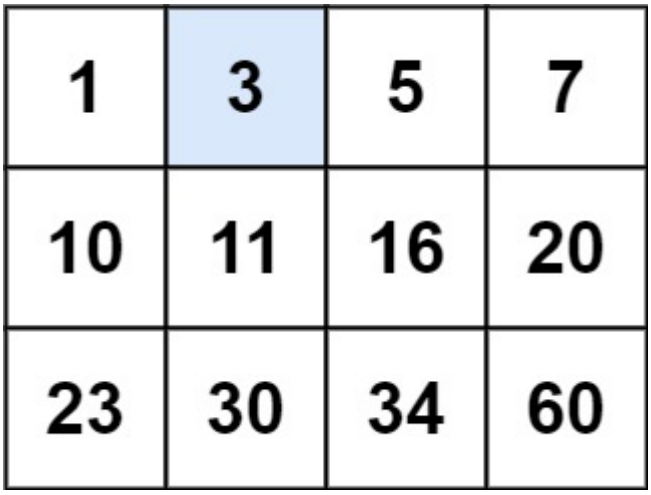
74. Search a 2D Matrix

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

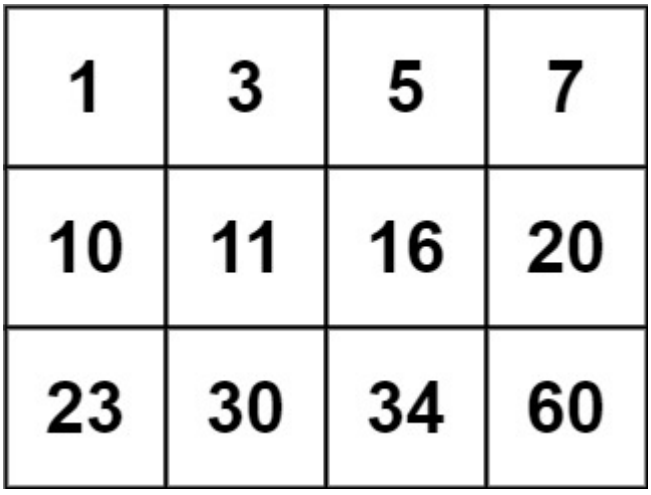
- Integers in each row are sorted from left to right.
- The first integer of each row is greater than the last integer of the previous row.

Example 1:




```
Input: matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]], target = 3
Output: true
```

Example 2:




```
Input: matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]], target = 13
Output: false
```

Constraints:

- `m == matrix.length`
- `n == matrix[i].length`
- `1 <= m, n <= 100`
- `-10<sup>4</sup> <= matrix[i][j], target <= 10<sup>4</sup>`

```python
class Solution:
    def searchMatrix(self, matrix: List[List[int]], target: int) -> bool:
        for i in range(len(matrix)):
            for j in range(len(matrix[0])):
                if matrix[i][j]==target:
                    return True
        return False
        #Method 2 : Efficient Approach
        i = 0
        j = len(matrix[0])-1

        while i<len(matrix) and j>=0:
            temp = matrix[i][j]
            if temp==target:
                return True
            elif temp>target:
                j-=1
            else:
                i+=1
        return False
```