

0074. 搜索二维矩阵

👤 ITCharge 🕒 大约 2 分钟

- 标签：数组、二分查找、矩阵
- 难度：中等

题目链接

- [0074. 搜索二维矩阵 - 力扣](#)

题目大意

描述： 给定一个 $m \times n$ 大小的有序二维矩阵 $matrix$ 。矩阵中每行元素从左到右升序排列，每列元素从上到下升序排列。再给定一个目标值 $target$ 。

要求： 判断矩阵中是否存在目标值 $target$ 。

说明：

- $m == matrix.length$ 。
- $n == matrix[i].length$ 。
- $1 \leq m, n \leq 100$ 。
- $-10^4 \leq matrix[i][j], target \leq 10^4$ 。

示例：

- 示例 1：

1	3	5	7
10	11	16	20
23	30	34	60

输入: `matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]]`, `target = 3`
输出: `True`

- 示例 2:

1	3	5	7
10	11	16	20
23	30	34	60

输入: `matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]]`, `target = 13`
输出: `False`

解题思路

思路 1：二分查找

二维矩阵是有序的，可以考虑使用二分搜索来进行查找。

1. 首先二分查找遍历对角线元素，假设对角线元素的坐标为 (row, col) 。把数组元素按对角线分为右上角部分和左下角部分。
2. 然后对于当前对角线元素右侧第 row 行、对角线元素下侧第 col 列进行二分查找。
 1. 如果找到目标，直接返回 `True`。
 2. 如果找不到目标，则缩小范围，继续查找。
 3. 直到所有对角线元素都遍历完，依旧没找到，则返回 `False`。

思路 1: 代码

py

```
class Solution:
    # 二分查找对角线元素
    def diagonalBinarySearch(self, matrix, diagonal, target):
        left = 0
        right = diagonal
        while left < right:
            mid = left + (right - left) // 2
            if matrix[mid][mid] < target:
                left = mid + 1
            else:
                right = mid
        return left

    def rowBinarySearch(self, matrix, begin, cols, target):
        left = begin
        right = cols
        while left < right:
            mid = left + (right - left) // 2
            if matrix[begin][mid] < target:
                left = mid + 1
            elif matrix[begin][mid] > target:
                right = mid - 1
            else:
                left = mid
                break
        return begin <= left <= cols and matrix[begin][left] == target

    def colBinarySearch(self, matrix, begin, rows, target):
        left = begin + 1
        right = rows
        while left < right:
            mid = left + (right - left) // 2
            if matrix[mid][begin] < target:
                left = mid + 1
            elif matrix[mid][begin] > target:
                right = mid - 1
            else:
                left = mid
```

```
        break
    return begin <= left <= rows and matrix[left][begin] == target

def searchMatrix(self, matrix: List[List[int]], target: int) -> bool:
    rows = len(matrix)
    if rows == 0:
        return False
    cols = len(matrix[0])
    if cols == 0:
        return False

    min_val = min(rows, cols)
    index = self.diagonalBinarySearch(matrix, min_val - 1, target)
    if matrix[index][index] == target:
        return True
    for i in range(index + 1):
        row_search = self.rowBinarySearch(matrix, i, cols - 1, target)
        col_search = self.colBinarySearch(matrix, i, rows - 1, target)
        if row_search or col_search:
            return True
    return False
```

思路 1：复杂度分析

- **时间复杂度：** $O(\log m + \log n)$ ，其中 m 、 n 分别是矩阵的行数和列数。
- **空间复杂度：** $O(1)$ 。