# 题目

给定M×N矩阵，每一行、每一列都按升序排列，请编写代码找出某元素。

示例:

现有矩阵 matrix 如下：

[

[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]

]

给定 target = 5，返回 true。

给定 target = 20，返回 false。

# 分析

## 方法一：直接查找

这个问题可以使用一种有效的方法来解决，即从矩阵的右上角开始查找。这样可以利用矩阵的特性，如果当前元素比目标值大，则可以向左移动一列，因为当前列的所有元素都比目标值大；如果当前元素比目标值小，则可以向下移动一行，因为当前行的所有元素都比目标值小。通过这种方式，可以逐步缩小搜索范围，直到找到目标值或者搜索范围为空。

具体步骤如下：

1、初始化行索引 `row` 为 0，列索引 `col` 为矩阵的列数减一。

2、不断循环，直到行索引超出矩阵的行数或者列索引小于 0：

- 如果当前元素等于目标值，则返回 true。

- 如果当前元素大于目标值，则将列索引减一，向左移动一列。

- 如果当前元素小于目标值，则将行索引加一，向下移动一行。

3、如果循环结束仍未找到目标值，则返回 false。

代码：

class Solution {

public:

bool searchMatrix(vector<vector<int>>& matrix, int target) {

if (matrix.empty() || matrix[0].empty()) {

return false;

}

int m = matrix.size();

int n = matrix[0].size();

int row = 0;

int col = n - 1;

while (row < m && col >= 0) {

if (matrix[row][col] == target) {

return true;

} else if (matrix[row][col] > target) {

col--;

} else {

row++;

}

}

return false;

}

};

在这个实现中，我们从矩阵的右上角开始查找，然后根据当前元素与目标值的大小关系来移动行索引和列索引，最终可以找到目标值或者确定不存在。

## 方法二：二分查找

对于这个问题，我们可以在每一行使用二分查找来查找目标值。由于每一行都是按升序排列的，因此可以利用这一特性进行二分查找。

具体步骤如下：

1、对于每一行，使用二分查找来查找目标值。如果找到目标值，则返回 true；如果找不到，则返回 false。

2、如果在任何一行找到了目标值，则返回 true；否则返回 false。

下面是具体的实现：

class Solution {

public:

bool searchMatrix(vector<vector<int>>& matrix, int target) {

if (matrix.empty() || matrix[0].empty()) {

return false;

}

int m = matrix.size();

int n = matrix[0].size();

for (int i = 0; i < m; ++i) {

// 对当前行进行二分查找

int left = 0;

int right = n - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (matrix[i][mid] == target) {

return true;

} else if (matrix[i][mid] < target) {

left = mid + 1;

} else {

right = mid - 1;

}

}

}

return false;

}

};

在这个实现中，我们对每一行都使用二分查找来查找目标值。如果在任何一行找到了目标值，则返回 true；否则返回 false。