**难度: 简单**

在一个 n \* m 的二维数组中，每一行都按照从左到右递增的顺序排序，每一列都按照从上到下递增的顺序排序。请完成一个函数，输入这样的一个二维数组和一个整数，判断数组中是否含有该整数。

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。

限制：

0 <= n <= 1000

0 <= m <= 1000

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/er-wei-shu-zu-zhong-de-cha-zhao-lcof

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

**思路:**

1). 先判断整体的二维数组行数是否为0, 为0直接返回

2). 整体循环对二维数组中的行数进行二分查找

3). 找到直接返回true

4). 直到最后还没找到直接返回false

|  |
| --- |
| public boolean findNumberIn2DArray(int[][] matrix, int target) {  //判断行数是否为0  if(matrix.length == 0) {  return false;  }  //不为0就遍历行数  for (int i = 0; i < matrix.length; i++) {  //行数里面的列不为0再继续  if (matrix[i].length != 0) {  //左为0  int left = 0;  //右为行的列长度-1  int right = matrix[i].length -1;  //二分查找  while(left <= right) {  //取中间  int mid = (left + right )/2;    //找到直接返回true  if (matrix[i][mid] == target) {  return true;    //大于直接向左移  }else if (matrix[i][mid] > target) {  right = mid -1;  }else {  //小于 向右移  left = mid + 1;  }  }  }  }  //能全部执行完,说明没有,直接返回false  return false;  } |