剑指offer 04 二维数组中的查找

Label：双指针、数组
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。

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。

- 暴力 遍历每一个元素
- 双指针 从右上角

```java
class Solution {
    public boolean findNumberIn2DArray(int[][] matrix, int target) {
        if (matrix == null || matrix.length == 0 || matrix[0].length == 0) {
            return false;
        }
        int rows = matrix.length, columns = matrix[0].length;
        int row = 0, column = columns - 1;
        while (row < rows && column >= 0) {
            int num = matrix[row][column];
            if (num == target) {
                return true;
            } else if (num > target) {
                column--;
            } else {
                row++;
            }
        }
        return false;
    }
}
```

- 双指针 从左下角

```
class Solution {
    public boolean findNumberIn2DArray(int[][] matrix, int target) {
        if (matrix == null || matrix.length == 0 || matrix[0].length == 0) {
            return false;
        }
        int rows = matrix.length, columns = matrix[0].length;
        int row = 0, column = columns - 1;
        while (row < rows && column >= 0) {
            int num = matrix[row][column];
            if (num == target) {
                return true;
            } else if (num > target) {
                column--;
            } else {
                row++;
            }
        }
        return false;
    }
}
```

- 二分法 区域递归

```
public boolean searchMatrix2(int[][] matrix, int target) {
        if (matrix == null || matrix.length == 0)
            return false;
        return search(matrix, target, 0, matrix[0].length - 1, 0, matrix.length
- 1);
    }

    private boolean search(int[][] matrix, int target, int left, int right, int
top, int bottom) {
        if (left > right || top > bottom) // 已无迭代区域
            return false;
        if (target < matrix[top][left] || target > matrix[bottom][right]) // 目标
值比矩阵的左上角小或者比矩阵的右小角大，肯定无法不能在矩阵中找到该值
            return false;
        int mid = (left + right) / 2;
        int row = top;
        while (row <= bottom && matrix[row][mid] <= target) { // 搜索中间列是否能找
到target，如果找不到就使row停在该行中间元素比target大的位置
            if (matrix[row][mid] == target)
                return true;
            row++;
        }
        return search(matrix, target, left, mid - 1, row, bottom) ||
                search(matrix, target, mid + 1, right, top, row - 1);
    }
```