二维数组查找

在一个二维数组中，每一行都按照从左到右递增的顺序排列，每一列都按照从上到下的顺序排列，请完成一个函数，输入这样一个二维数组和一个整数，判断数组中是否包含该整数。

例如：

1	**2**	**8**	**9**
2	4	9	12
4	7	10	13
6	8	11	15

假如我们要找数字 7

- 选取右上角的数字 9， 由于 9 大于 7，并且 9 还是 第 4 列最小的数，所以排除第4列。
- 选取 8，同理，排除第3列。
- 选取 2，2小于7，则 数字一定不在 2 所在行，排除第1行。
- 选取 4， 4 也小于 7，排除第2行。
- 选取 7 ，7 等于 7，找到了，结束遍历。

```
public class FindInPartiallySortedMatrix_04 {
    public static void main(String[] args) {
        int[][] matrix = {{1,2,8,9}, {2,4,9,12},{4,7,10,13},{6,8,11,15}};
        System.out.println("find 7 : " + find(matrix, 4, 4, 7));
    }

    public static boolean find(int[][] matrix, int rows, int columns, int number) {
        boolean found = false;
        if (matrix != null && rows > 0 && columns > 0) {
            int row = 0;
            int column = columns - 1;
            // row column 的值表示从最右端开始遍历
            while (row < rows && column >= 0) {
                if (matrix[row][column] == number) {
                    found = true;
                    break;
                } else if (matrix[row][column] > number) {
                    -- column;
                } else {
                    ++ row;
                }
            }
        }
        return found;
    }
}
```