链滴

LeetCode 剑指 Offer #04 二维数组中的查找

作者：matthewhan

原文链接：https://ld246.com/article/1597395776037

来源网站：链滴

Problem Description

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。

e.g.

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。

note

- 0 <= n <= 1000
- 0 <= m <= 1000

Solution

方法有很多，单纯双for循环暴力肯定太low。主要还是双指针、二分法这些。不过有个思路很好，站

该矩阵的右上角来看，这货就是一个「二叉搜索树」。

二叉搜索树的性质：

1. 节点的左子树上的所有节点的值都小于等于节点的值；
2. 节点的右子树上的所有节点的值都大于等于节点的值；
3. 左子树和右子树也都是BST。

那就模拟一颗二叉搜索树来做咯：

```
public class LcOf04 {

    boolean flag = false;

    /**
     * 当成二叉搜索树来做
     *
     * @param matrix
     * @param target
     * @return
     */
    public boolean findNumberIn2DArray(int[][] matrix, int target) {
        dfs(matrix, matrix.length - 1, 0, target);
        return flag;
    }

    public void dfs(int[][] matrix, int i, int j, int target) {
        if (i >= 0 && i < matrix.length && j >= 0 && j < matrix[0].length) {
            if (matrix[i][j] == target) {
                flag = true;
        return;
            } else if (matrix[i][j] < target) {
                dfs(matrix, i, j + 1, target);
            } else {
                dfs(matrix, i - 1, j, target);
            }
        }
    }

}
```

其中注意下，如果用双指针、二分法最好从右上角为起点。