查阅更多的题解，请点击

Problem

240. Search a 2D Matrix II(Medium)

Write an efficient algorithm that searches for a value in an m x n matrix. This matrix has the following properties:

Integers in each row are sorted in ascending from left to right.
Integers in each column are sorted in ascending from top to bottom.
Example:

Consider the following matrix:

```
[
    [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]
]
```

Given target = 5, return true.

Given target = 20, return false.

Solution

设矩阵有m行n列

O(m+n) time, O(1) space

问题是在有序数据中查找给定元素是否存在，这类问题一般可以用**分治策略**来做（通过分治缩小我们查找的元素范围），最典型的就是二分查找。

注意这里的数据区间，最小值为matrix[0]][0]，最大值为matrix[m-1][n-1]，希望每次能缩小我们查找的元素范围。二分查找**是利用中位数**，每次减少一半的搜索空间。该题解法是类似二分搜索，可以认为matrix[0][n-1]是一个广义上的中位数，利用该值reduce原问题，如下例（target为5）：

```
[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]
```

middle=15>target，可以排除一定比15大的部分

```
[1,   4,  7, 11]
[2,   5,  8, 12]
[3,   6,  9, 16]
[10, 13, 14, 17]
[18, 21, 23, 26]
```

middle=7>target，可以排除一定比7大的部分

```
[1,   4,  7]
[2,   5,  8]
[3,   6,  9]
[10, 13, 14]
[18, 21, 23]
```

同理，

```
[1,   4]
[2,   5]
[3,   6]
[10, 13]
[18, 21]
```

命中：

```
[2,   5]
[3,   6]
[10, 13]
[18, 21]
```

```
class Solution
{
public:
    bool searchMatrix(vector<vector<int>> &matrix, int target)
    {
        int m = matrix.size();
        if (m == 0)
            return false;
        int n = matrix.front().size();
        int i = 0, j = n - 1;
        while (i < m && j >= 0)
        {
            if (matrix[i][j] == target)
                return true;
            matrix[i][j] > target ? j-- : i++;
        }
        return false;
    }
};
```

GitHub传送门