1. Search a 2D Matrix II

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.

**解** 从右上角到左下角，往左搜索变小，往下搜索变大，假设在位置 i, j处的元素值为n，搜索过程为：

* target == n ：找到，结束
* target > n：往下走，增加n
* target < n：往左走，降低n

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