第18题——二维数组中的查找

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。请完成一个高效的函数，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。

示例:

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。

限制：

0 <= n <= 1000

0 <= m <= 1000

行尾查找法（1）

解题思路 核心思想： 每次从行尾进行比较 小于target则寻找下一行 大于target则从该元素向前挨个作比较

```
bool findNumberIn2DArray(int** matrix, int matrixSize, int* matrixColSize, int target){
    if(matrixSize==0||*matrixColSize==0)//行或列任意一个为0则二维数组不存在
    {
        return false;
    }
    int i=0,j=*matrixColSize-1;//每次都从行尾元素进行比较
    while(i<matrixSize&&j>=0)
    {
        if(target==matrix[i][j])//该元素等于target，返回true
          return true;
        else if(target>matrix[i][j])//如果该值小于target,行+1
          i++;
        else                        //如果该值大于target，列-1
          j--;
    }
    return false;
}
```

从右上角开始找

```
class Solution {
public:
    bool findNumberIn2DArray(vector<vector<int>>& matrix, int target) {
    if(matrix.empty())return false;
    int i=0;//行
    int j=matrix[0].size()-1;//列
    while(i<matrix.size() && j>=0)
    {
      if(matrix[i][j]==target)return true;
      else if(matrix[i][j]>target)j--;
      else i++;
    }
    return false;
    }
};
```

核心思路：

首先我们的矩阵是从左到右递增，从上到下递增，因此，如果小于右上角元素的话，那么肯定也小于这一列，因此，这一列可以剔除，然后我们再比较右上角左侧的，如果还是小，那么再剔除一列，如果更大，那么一定不在本行，因此，行+1；