题目 二维数组中的查找

热点指数 147187 通过率 23.71% 考点 数组

具体题目

在一个二维数组中（每个一维数组的长度相同），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。

解决方案

一种是： 把每一行看成有序递增的数组， 利用二分查找， 通过遍历每一行得到答案， 时间复杂度是nlogn

```
public class Solution {
    public boolean Find(int [][] array,int target) {


        for(int i=0;i<array.length;i++){
            int low=0;
            int high=array[i].length-1;
            while(low<=high){
                int mid=(low+high)/2;
                if(target>array[i][mid])
                    low=mid+1;
                else if(target<array[i][mid])
                    high=mid-1;
                else
                    return true;
            }
        }
        return false;
    }
}
```

另外一种思路是： 利用二维数组由上到下，由左到右递增的规律， 那么选取右上角或者左下角的元素a[row][col]与target进行比较， 当target小于元素a[row][col]时，那么target必定在元素a所在行的左边, 即col--； 当target大于元素a[row][col]时，那么target必定在元素a所在列的下边, 即row++；

```
public class Solution {
    public boolean Find(int [][] array,int target) {
        int row=0;
        int col=array[0].length-1;
        while(row<=array.length-1&&col>=0){
            if(target==array[row][col])
                return true;
            else if(target>array[row][col])
                row++;
            else
                col--;
        }
        return false;
    }
}
```

Java版本

```
public class Solution {
    public boolean Find(int target, int [][] array) {
        int rows = array.length;
        int cols = array[0].length;
        int i=rows-1,j=0;
        while(i>=0 && j<cols){
            if(target<array[i][j])
                i--;
            else if(target>array[i][j])
                j++;
            else
                return true;
        }
        return false;
    }
}
```

最佳答案：没有之一。思路：首先我们选择从左下角开始搜寻，(为什么不从左上角开始搜寻，左上角向右和向下都是递增，那么对于一个点，对于向右和向下会产生一个岔路；如果我们选择从左下脚开始搜寻的话，如果大于就向右，如果小于就向下)。

```
public class Solution {
    public boolean Find(int [][] array,int target) {
    int len = array.length-1;
        int i = 0;
        while((len >= 0)&& (i < array[0].length)){
            if(array[len][i] > target){
                len--;
            }else if(array[len][i] < target){
                i++;
            }else{
                return true;
            }
        }
        return false;
    }
}


public class Solution {
    public boolean Find(int [][] array,int target) {
        int m = array.length - 1;
        int i = 0;
        while(m >= 0 && i < array[0].length){
            if(array[m][i] > target)
                m--;
            else if(array[m][i] < target)
                i++;
            else
                return true;
        }


        return false;
    }
}
```

```
public class Solution {
    public boolean Find(int [][] array,int target) {
        for(int[] i : array){
            for(int j : i){
                if(j==target)return true;
            }
        }
        return false;
    }
}
```

```
public class Solution {
    public boolean Find(int target, int [][] array) {
        boolean flag = false;
        int x=0;
        int y=0;
        for(int i=0;i < array.length;i++) {
            for(int j=0;j<array[i].length;j++) {
                if(target==array[i][j]){
                    flag = true;
                    break;
                }
            }
        }
        if(flag) {
            System.out.println("exist!"+target+",location:"+"array["+x+"]["+y+"]");
        }
        return flag;
    }
}
```

```
public class Solution {
    public boolean Find(int target, int [][] array) {
        if(array == null || array[0].length == 0){
            return false;
        }
        for(int i = 0;i < array.length;i++){
            int head = 0;
            int tail = array[i].length-1;
            while(head<=tail){
                int mid = (head+tail)/2;
                if(target < array[i][mid]){
                    tail = mid - 1;
                }
                else if(target > array[i][mid]){
                    head = mid + 1;
                }
                else
                    return true;
            }
        }
        return false;
    }
}
```