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.

For 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 1

We start search the matrix from top right corner, initialize the current position to top right corner, if the target is greater than the value in current position, then the target can not be in entire row of current position because the row is sorted, if the target is less than the value in current position, then the target can not in the entire column because the column is sorted too. We can rule out one row or one column each time, so the time complexity is O(m+n).

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

written by chicm original link here

Solution 2

```cpp
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;
        else if (matrix[i][j] > target) {
            j--;
        } else
            i++;
    }
    return false;
}
```

written by loki2441 original link here

Solution 3

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

}

written by simkieu original link here