# Medium

31 December 2022 11:09

① Search in a 2D Matrix

## Search in a matrix

Easy Accuracy: 41.62% Submissions: 88K+ Points: 2



Given a matrix **mat**[][] of size **N** x **M**, where every row and column is sorted in increasing order, and a number **X** is given. The task is to find whether element **X** is present in the matrix or not.

**Example 1:**

**Input**:
N = 3, M = 3
mat[][] = 3 30 38
44 52 54
57 60 69
X = 62
**Output**:
0
**Explanation**:
62 is not present in the matrix, so output is 0

Screen clipping taken: 31-12-2022 11:18

```java
class Sol
{
    public static int matSearch(int mat[][], int n, int m, int x)
    {
        // your code here
        int si = 0;
        int ei = n-1;
        int ridx = -1;
        while(si <= ei){
            int mid  = (si  + ei)/2;
            if(mat[mid][m-1] == x){
                return 1;
            }
            else if(mat[mid][m-1] > x){
                ridx = mid;
                ei = mid -1;
            }
            else{
                si = mid +1;
            }
        }
        if(ridx == -1){
            return 0;
        }
        si =0;
        ei = m-1;
        while(si <= ei){
            int mid = (si + ei)/2;
            if(mat[ridx][mid] == x){
                return 1;
            }
            else if(mat[ridx][mid] > x){
                ei = mid -1;
            }
            else{
                si = mid +1;
            }
        }
        return 0;
    }
}
```

Screen clipping taken: 31-12-2022 11:20

Search in a 2D matrix II

Screen clipping taken: 31-12-2022 11:24

## 240. Search a 2D Matrix II

Medium 9.8K 158

Companies

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

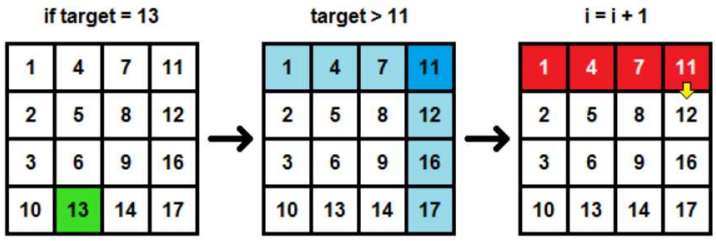
- Integers in each row are sorted in ascending from left to right.
- Integers in each column are sorted in ascending from top to bottom.

**Example 1:**

| | | | | |
|---|---|---|---|---|
| 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 |

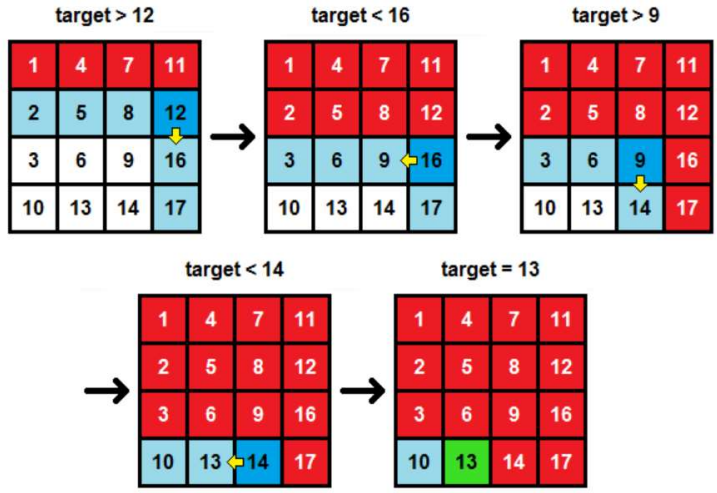
**Input:** matrix = [[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[10,13,14,17,24],[18,21,23,26,30]], target = 5
**Output:** true

If we start from the top right corner of **M** and treat this like a modified binary search, we can eliminate an entire row or an entire column each time we check a **cell**:




We'll then just need to adjust our **i** or **j** value to move to the top right corner "midpoint" of the remaining matrix each time to narrow in on our target (**T**):

Screen clipping taken: 31-12-2022 11:25




This will drop the time complexity to **O(m + n)**.

Screen clipping taken: 31-12-2022 11:25

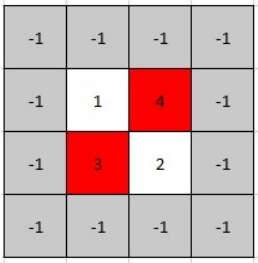
```java
class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
        int i = 0, j= matrix[0].length-1;
        while(i<matrix.length && j>=0){
            int cell = matrix[i][j];
            if(cell == target){
                return true;
            }
            else if(cell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

Screen clipping taken: 31-12-2022 11:29

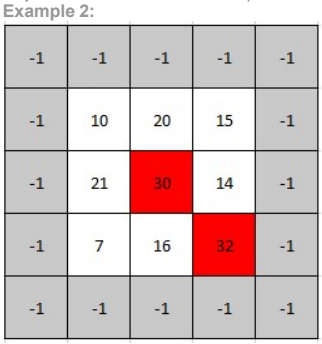
Question 2 : ----> find the peak element II

1901. Find a Peak Element II
Medium
1.2K
87
Companies
A **peak** element in a 2D grid is an element that is **strictly greater** than all of its **adjacent** neighbors to the left, right, top, and bottom.
Given a **0-indexed** m x n matrix mat where **no two adjacent cells are equal**, find **any** peak element mat[i][j] and return *the length 2 array* [i,j].
You may assume that the entire matrix is surrounded by an **outer perimeter** with the value -1 in each cell.
You must write an algorithm that runs in O(m log(n)) or O(n log(m)) time.

**Example 1:**




**Input:** mat = [[1,4],[3,2]]
**Output:** [0,1]
**Explanation:** Both 3 and 4 are peak elements so [1,0] and [0,1] are both acceptable answers.
**Example 2:**




**Input:** mat = [[10,20,15],[21,30,14],[7,16,32]]
**Output:** [1,1]
**Explanation:** Both 30 and 32 are peak elements so [1,1] and [2,2] are both acceptable answers.

```java
class Solution {
    public int findmax(int arr[]){
        int maxi = 0;
        int index=0;
        for(int i=0;i<arr.length;i++){
            if(maxi < arr[i]){
                maxi = arr[i];
                index =i;
            }
        }
        return index;
    }
    public int[] findPeakGrid(int[][] mat) {
        int m = mat.length;
        int n = mat[0].length;
        int low = 0, high = m-1, maxindex =0;
        while(low < high){
            int mid = (low + high)/2;
            maxindex = findmax(mat[mid]);
            if(mat[mid][maxindex] < mat[mid +1][maxindex]){
                low = mid +1;
            }
            else{
                high = mid;

            }
        }
        maxindex = findmax(mat[low]);
        return new int[]{low , maxindex};
    }
}
```

Question 3
Median in a row-wise sorted Matrix
Given a row wise sorted matrix of size **R*C** where R and C are always **odd**, find the median of the matrix.
**Example 1:**
**Input**:
R = 3, C = 3
M = [[1, 3, 5],
[2, 6, 9],
[3, 6, 9]]
**Output:** 5
**Explanation**: Sorting matrix elements gives us {1,2,3,3,5,6,6,9,9}. Hence, 5 is median.