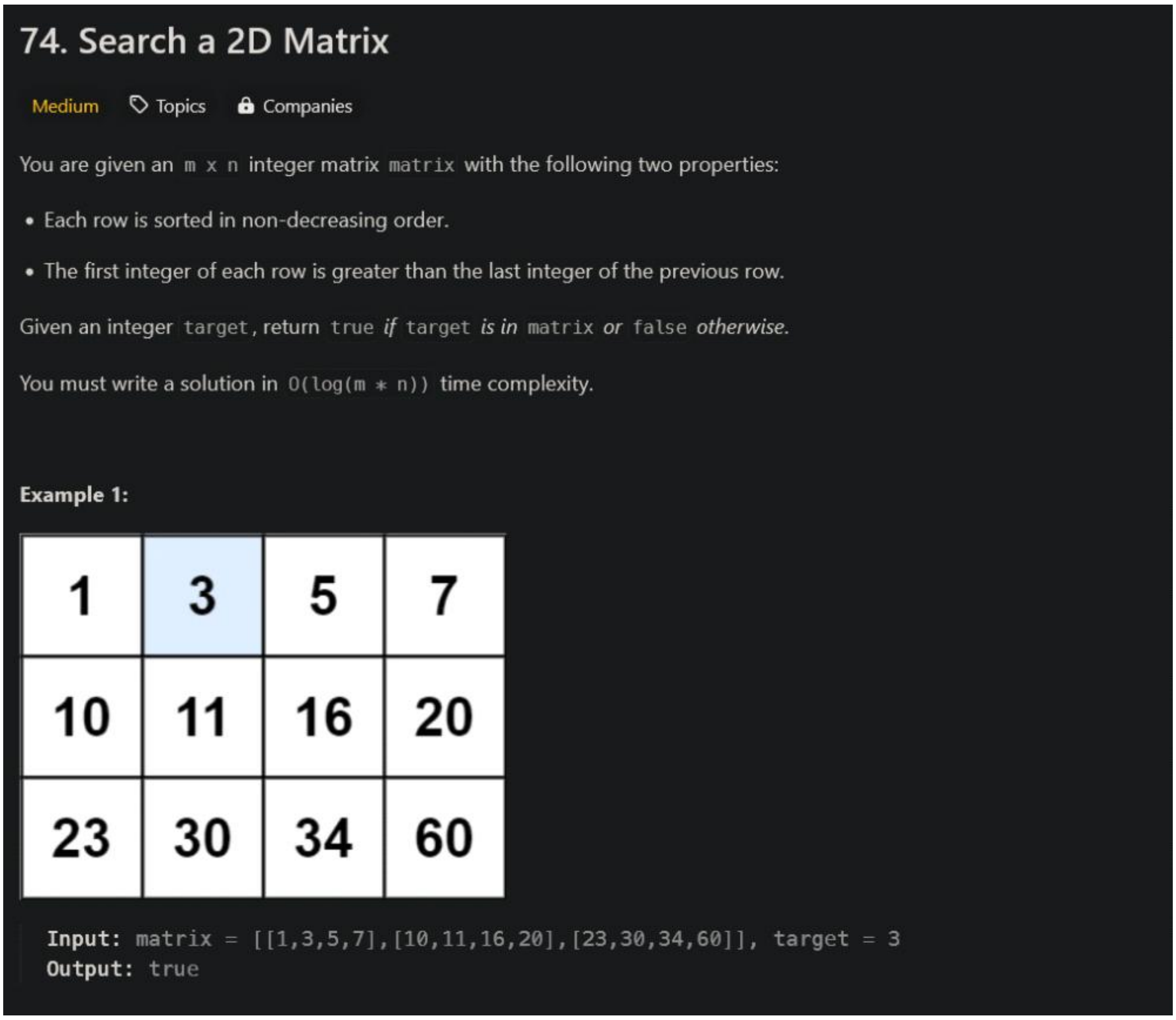
Search in a 2D matrix :-



74. Search a 2D Matrix

Medium Topics Companies

You are given an `m x n` integer matrix `matrix` with the following two properties:

- Each row is sorted in non-decreasing order.
- The first integer of each row is greater than the last integer of the previous row.

Given an integer `target`, return `true` *if* `target` *is in* `matrix` *or* `false` *otherwise.*

You must write a solution in `O(log(m * n))` time complexity.

Example 1:

1	3	5	7
10	11	16	20
23	30	34	60

```
Input: matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]], target = 3
Output: true
```

```
class Solution {
 public:
    bool searchMatrix (vector<vector<int>>& matrix,
    int target) {
    int m = matrix.size(), n = matrix[0].size();
    int lo = 0, hi = m;
    while(lo < hi) {
          int mid = (lo + hi)/2;
if(matrix[mid][n-1]>=target)hi=mid;
else lo = mid+1; }
if(lo = = m) return 0;
return binary_search(matrix[lo].begin(),
matrix[lo].end(), target); }};
```

T.C. O(mlogm + nlogn)

T.C, O(m+n) Approach :-

```
class solution {
public:
bool searchMatrix(vector<vector<int>> &matrix,
int target) {
 int m= matrix.size(), n=matrix[0].size();
 int row= 0;
 for (int i=0; i<=m; i++){
     if(i== m) return 0;
     if (matrix[i][n-1] > = target) {
           row = i;
           break; } }

      for (int i= 0; i<n; i++) {
  if(matrix [row][i]= = target) return 1;
```

```
        if(matrix[row][i]== target)return 1;
        }
      return 0;
    }
  };
```

Note:-

In C++, the standard binary_search function (from the <algorithm> header) returns a boolean.

```
bool binary_search(InputIterator first, InputIterator
 last, const T & value);
```

- when the value is not found, binary_search returns false.

- If you need the position (iterator) of the value, you should use std::lower_bound or std::equal_range instead.

- binary_search only tells you whether the value exist.