# **Search a 2D Matrix**

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 from left to right.
* The first integer of each row is greater than the last integer of the previous row.

**Example 1:**



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

**Output:** true

**Example 2:**



**Input:** matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]], target = 13

**Output:** false

**Constraints:**

* m == matrix.length
* n == matrix[i].length
* 1 <= m, n <= 100
* -104 <= matrix[i][j], target <= 104

public class Solution {

public bool SearchMatrix(int[][] matrix, int target) {

int rowVal = BSearchRow(matrix, target, 0, matrix.Length-1);

return BSearchValue(matrix[rowVal],target,0,matrix[rowVal].Length-1);

}

//Binary search for identifying the row : row[0] <= target <= row[length-1]

int BSearchRow(int[][] matrix, int target, int lowIndex, int highIndex)

{

int midIndex = (highIndex + lowIndex)/2;

int retVal = 0;

if(midIndex == lowIndex || midIndex == highIndex)

{

retVal = highIndex;

if(matrix[lowIndex][0] <= target && target <=matrix[midIndex][matrix[midIndex].Length-1])

{

retVal = lowIndex;

}

}

else if(target < matrix[midIndex][0])

{

retVal = BSearchRow(matrix, target, lowIndex, midIndex);

}

else if(target > matrix[midIndex][matrix[midIndex].Length-1])

{

retVal = BSearchRow(matrix, target, midIndex, highIndex);

}

else

{

retVal = midIndex;

}

return retVal;

}

//Binary search for identifying element in a single dimention array

bool BSearchValue(int[] matrix, int target, int lowIndex, int highIndex)

{

int midIndex = (highIndex + lowIndex)/2;

bool retVal = false;

if(matrix[midIndex] == target)

{

retVal = true;

}

else if(midIndex == lowIndex || midIndex == highIndex)

{

if(matrix[lowIndex] == target || matrix[highIndex] == target)

{

retVal = true;

}

else

{

retVal = false;

}

}

else if(matrix[midIndex] < target)

{

retVal = BSearchValue(matrix, target, midIndex, highIndex);

}

else if(matrix[midIndex] > target)

{

retVal = BSearchValue(matrix, target, lowIndex, midIndex);

}

else

{

retVal = false;

}

return retVal;

}

}