<https://leetcode.com/problems/search-a-2d-matrix/>

**Search a 2D Matrix**

**You are given an m x n integer 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:

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

Output: true

Constraints:

m == matrix.length

n == matrix[i].length

1 <= m, n <= 100

-104 <= matrix[i][j], target <= 104

**Method 1: ()**

Time Complexity: O(log m +log n) *[]*

Space Complexity: O(1) *[]*

bool searchMatrix(vector<vector<int>>& matrix, int target) {

        //rowSearch

        int m=matrix.size(), n=matrix[0].size();

        int l=0, r=m-1, mid, row=-1;

        while(l<=r){

            mid = l + (r-l)/2;

            if(target>=matrix[mid][0] && target<=matrix[mid][n-1]){

                row = mid;

                break;

            }

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

                r = mid - 1;

            else l = mid + 1;

        }

        if(row != -1) {

            l = 0, r = n-1;

            while(l<=r){

                mid = l + (r-l)/2;

                if(target==matrix[row][mid])

                    return true;

                else if(target<matrix[row][mid])

                    r = mid - 1;

                else l = mid + 1;

            }

        }

        return false;

    }