074. Search a 2D Matrix

074 Search a 2D Matrix

• Binary Search+array

Description

Write an efficient algorithm that searches for a value in an $m \times n$ 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.

For example,

Consider the following matrix:

```
[
[1, 3, 5, 7],
[10, 11, 16, 20],
[23, 30, 34, 50]
]
```

Given target = 3, return true.

1. Thought line

2. Binary Search+array

```
class Solution {
private:
    void binarySearchMatrix(vector<vector<int>>& matrix, int st, int ed, int target, bool& res){
        if (matrix.empty()||st>ed||res) return;
        if (st==ed){
           // corner case
            for (int i = 0; !matrix[st].empty()&&i<=matrix[st].size()-1; ++i)
               if (matrix[st][i]==target)
                   res = true;
            return;
       int midRow = (st+ed)/2, n = matrix[midRow].size()-1;
        if (target>matrix[midRow][n])
           binarySearchMatrix(matrix, midRow+1, ed, target, res);
       else
           binarySearchMatrix(matrix, st, midRow, target, res);
public:
    bool searchMatrix(vector<vector<int>>>& matrix, int target) {
       bool res = false;
       binarySearchMatrix(matrix, 0, matrix.size()-1, target, res);
        return res;
};
```