

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

```
1 class Solution {
2 private:
3     void binarySearchMatrix(vector<vector<int>>& matrix, int st, int ed, int target, bool& res){
4         if (matrix.empty() || st>ed || res) return;
5
6         if (st==ed){
7             // corner case
8             for (int i = 0; !matrix[st].empty()&&i<=matrix[st].size()-1; ++i)
9                 if (matrix[st][i]==target)
10                     res = true;
11             return;
12         }
13         int midRow = (st+ed)/2, n = matrix[midRow].size()-1;
14         if (target>matrix[midRow][n])
15             binarySearchMatrix(matrix, midRow+1, ed, target, res);
16         else
17             binarySearchMatrix(matrix, st, midRow, target, res);
18     }
19 }
20 public:
21     bool searchMatrix(vector<vector<int>>& matrix, int target) {
22         bool res = false;
23         binarySearchMatrix(matrix, 0, matrix.size()-1, target, res);
24         return res;
25     }
26 };
```