Previous Next

74. Search in 2D Matrix

April 14, 2019 | 29.3K views

Average Rating: 4.56 (18 votes)

Write an efficient algorithm that searches for a value in an *m* x *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.

Example 1:

```
Input:
matrix = [
  [1,   3,  5,  7],
  [10, 11, 16, 20],
  [23, 30, 34, 50]
]
target = 3
Output: true
```

Example 2:

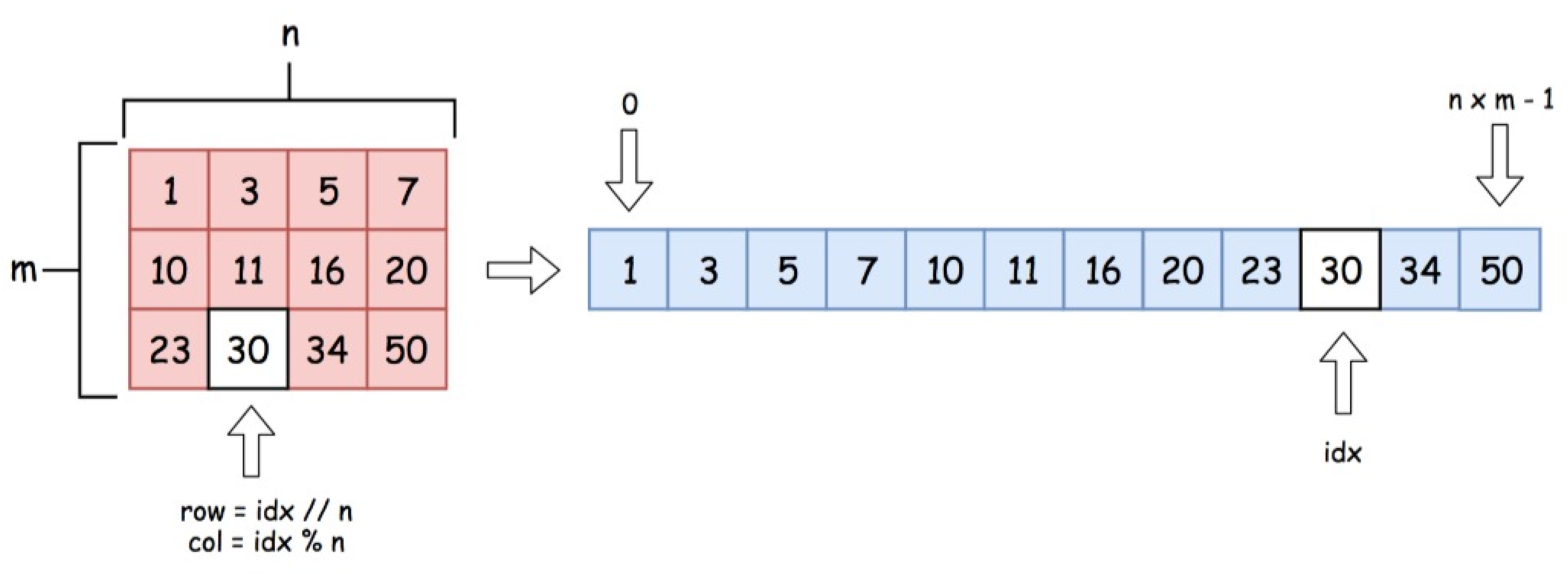
```
Input:
matrix = [
  [1,   3,  5,  7],
  [10, 11, 16, 20],
  [23, 30, 34, 50]
]
target = 13
Output: false
```

Solution

Approach 1: Binary search

Intuition

One could notice that the input matrix `m x n` could be considered as a sorted array of length `m x n`.



Sorted array is a perfect candidate for the binary search because the element index in this *virtual* array (*for sure we're not going to construct it for real*) could be easily transformed into the row and column in the initial matrix

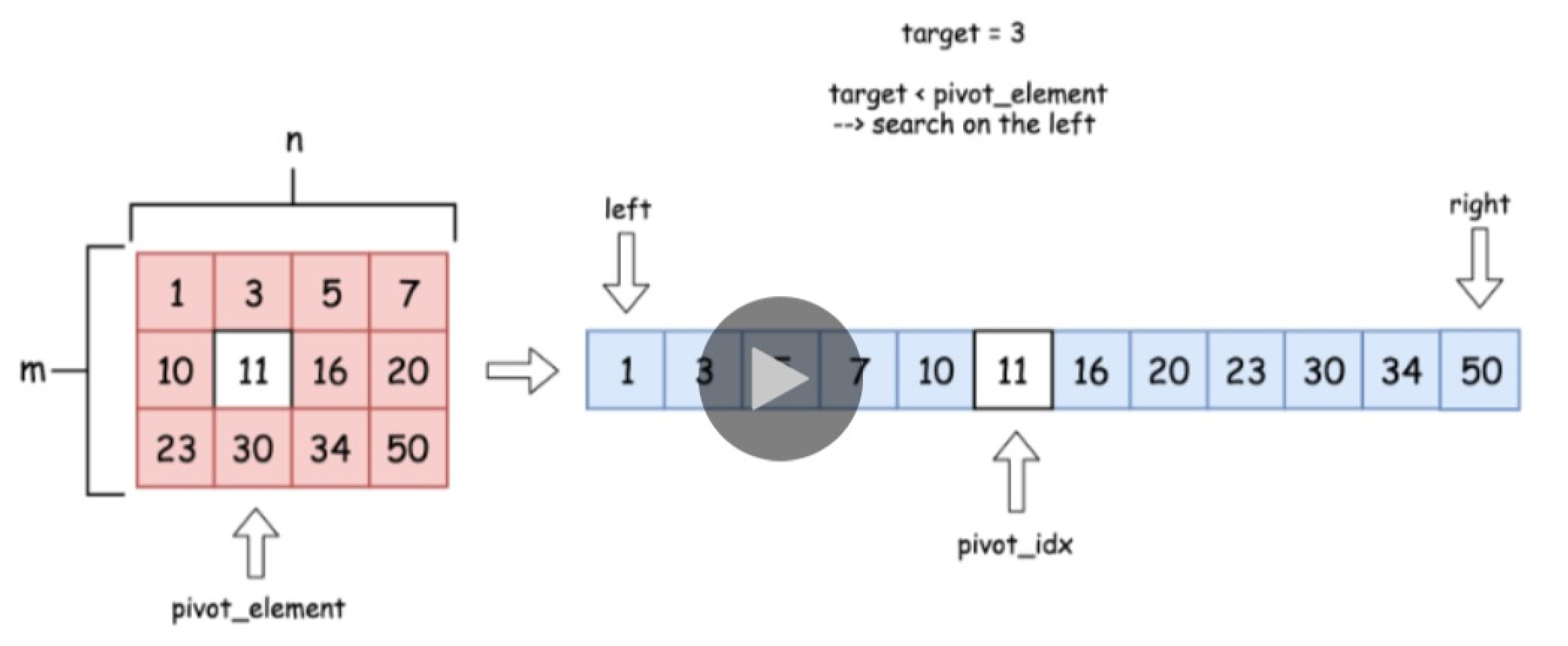
> `row = idx // n` and `col = idx % n`.

Algorithm

The algorithm is a standard binary search :

- Initialise left and right indexes `left = 0` and `right = m x n - 1`.
- While `left < right` :
  - Pick up the index in the middle of the virtual array as a pivot index : `pivot_idx = (left + right) / 2`.
  - The index corresponds to `row = pivot_idx // n` and `col = pivot_idx % n` in the initial matrix, and hence one could get the `pivot_element`. This element splits the virtual array in two parts.
  - Compare `pivot_element` and `target` to identify in which part one has to look for `target`.

Implementation



C++ Java Python Copy

```python
class Solution:
    def searchMatrix(self, matrix: List[List[int]], target: int) -> bool:
        m = len(matrix)
        if m == 0:
            return False
        n = len(matrix[0])

        # binary search
        left, right = 0, m * n - 1
        while left <= right:
                pivot_idx = (left + right) // 2
                pivot_element = matrix[pivot_idx // n][pivot_idx % n]
                if target == pivot_element:
                    return True
                else:
                    if target < pivot_element:
                        right = pivot_idx - 1
                    else:
                        left = pivot_idx + 1
        return False
```

Complexity Analysis

- Time complexity : $\mathcal{O}(\log(mn))$ since it's a standard binary search.
- Space complexity : $\mathcal{O}(1)$.

Rate this article:

Previous Next

Comments: 14

Sort By

Type comment here... (Markdown is supported)

Preview Post

witcher3 ★ 43 June 3, 2019 11:40 PM

we can do this in O(log(m)) + O(log(n)) if we do two binary searches. One for row and another for col.

```
def binSearch(self, matrix, target, l, h):
        while l <= h:
            m = (h + l) / 2
```

Read More

25 Share Reply

SHOW 9 REPLIES

dfsbfs ★ 12 August 23, 2019 9:31 PM Report

who on earth uses '//' as division sign.

10 Share Reply

SHOW 7 REPLIES

laplachar ★ 4 May 7, 2020 12:10 AM

I don't understand the intuition of this solution. Why is that you can extract the row and column from the pivot index by doing pivotIndex/columns and pivotIndex % columns?

3 Share Reply

SHOW 1 REPLY

Echo226 ★ 23 December 3, 2019 4:05 AM

For "(left + right) // 2", using "left + (right - left) // 2" might be better since left+right might cause overflow issue if both left and right is very large.

2 Share Reply

s961206 ★ 734 July 11, 2019 7:15 AM Report

O(logm + logn) solution for Java:

```
public boolean searchMatrix(int[][] matrix, int target) {
    if(matrix.length == 0 || matrix[0].length == 0) return false;
```

Read More

2 Share Reply

SHOW 2 REPLIES

darren782439809 ★ 14 June 17, 2020 7:44 AM Report

I have a easy-understanding solution without binary search
We start the left-bottom value V,
if (target == V) return true;
else if (target < v) we move upward;
else if (target > v) we move rightward (we can early stop in this case: once we meet the value > target ,

Read More

1 Share Reply

sddlpeter ★ 18 January 14, 2020 5:25 AM

thanks!

1 Share Reply

haoyangfan ★ 897 July 27, 2019 10:32 AM

two binary search, 0ms, java
Time: O(log(m) + log(n)
Space: O(1)

```
class Solution {
```

Read More

1 Share Reply

SHOW 1 REPLY

Reputation

suthee ★ 1 July 2, 2020 3:43 AM Report

I implemented 2D binary search and felt so stupid after I looked at the solution.

0 Share Reply

sainvamshid ★ 6 May 17, 2020 11:31 AM Report

```
bool searchMatrix(vector<vector<int>>& matrix, int target) {
    if (!matrix.size())
        return false;
    int i = 0, j = matrix[0].size()-1;
```

Read More

0 Share Reply

‹ 1 2 ›