Problem 378: Kth Smallest Element in a Sorted Matrix

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

Given an

n x n

matrix

where each of the rows and columns is sorted in ascending order, return

the

k

th

smallest element in the matrix

.

Note that it is the

k

th

smallest element

in the sorted order

, not the

k

th

distinct

element.

You must find a solution with a memory complexity better than

O(n

2

)

.

Example 1:

Input:

matrix = [[1,5,9],[10,11,13],[12,13,15]], k = 8

Output:

13

Explanation:

The elements in the matrix are [1,5,9,10,11,12,13,

13

,15], and the 8

th

smallest number is 13

Example 2:

Input:

matrix = [[-5]], k = 1

Output:

-5

Constraints:

n == matrix.length == matrix[i].length

1 <= n <= 300

-10

9

<= matrix[i][j] <= 10

9

All the rows and columns of

matrix

are

guaranteed

to be sorted in

non-decreasing order

.

1 <= k <= n

2

Follow up:

Could you solve the problem with a constant memory (i.e.,

O(1)

memory complexity)?

Could you solve the problem in

O(n)

time complexity? The solution may be too advanced for an interview but you may find reading

this paper

fun.

Code Snippets

C++:

```
class Solution {
public:
int kthSmallest(vector<vector<int>>& matrix, int k) {

}
};
```

Java:

```
class Solution {
public int kthSmallest(int[][] matrix, int k) {

}
}
```

Python3:

```
class Solution:
def kthSmallest(self, matrix: List[List[int]], k: int) -> int:
```

Python:

```
class Solution(object):
def kthSmallest(self, matrix, k):
"""
:type matrix: List[List[int]]
:type k: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[][]} matrix
* @param {number} k
* @return {number}
*/
var kthSmallest = function(matrix, k) {

};
```

TypeScript:

```
function kthSmallest(matrix: number[][], k: number): number {

};
```

C#:

```
public class Solution {
public int KthSmallest(int[][] matrix, int k) {
```

```
}
}
```

C:

```
int kthSmallest(int** matrix, int matrixSize, int* matrixColSize, int k) {

}
```

Go:

```
func kthSmallest(matrix [][]int, k int) int {

}
```

Kotlin:

```
class Solution {
fun kthSmallest(matrix: Array<IntArray>, k: Int): Int {

}
}
```

Swift:

```
class Solution {
func kthSmallest(_ matrix: [[Int]], _ k: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn kth_smallest(matrix: Vec<Vec<i32>>, k: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[][]} matrix
# @param {Integer} k
# @return {Integer}
def kth_smallest(matrix, k)

end
```

PHP:

```
class Solution {

/**
* @param Integer[][] $matrix
* @param Integer $k
* @return Integer
*/
function kthSmallest($matrix, $k) {

}
}
```

Dart:

```
class Solution {
int kthSmallest(List<List<int>> matrix, int k) {

}
}
```

Scala:

```
object Solution {
def kthSmallest(matrix: Array[Array[Int]], k: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec kth_smallest(matrix :: [[integer]], k :: integer) :: integer
def kth_smallest(matrix, k) do
```

```
end
end
```

Erlang:

```
-spec kth_smallest(Matrix :: [[integer()]], K :: integer()) -> integer().
kth_smallest(Matrix, K) ->
.
```

Racket:

```
(define/contract (kth-smallest matrix k)
(-> (listof (listof exact-integer?)) exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Kth Smallest Element in a Sorted Matrix
* Difficulty: Medium
* Tags: array, sort, search, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int kthSmallest(vector<vector<int>>& matrix, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Kth Smallest Element in a Sorted Matrix
```

```
* Difficulty: Medium
* Tags: array, sort, search, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int kthSmallest(int[][] matrix, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Kth Smallest Element in a Sorted Matrix
Difficulty: Medium
Tags: array, sort, search, queue, heap

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def kthSmallest(self, matrix: List[List[int]], k: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def kthSmallest(self, matrix, k):
"""
:type matrix: List[List[int]]
:type k: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Kth Smallest Element in a Sorted Matrix
* Difficulty: Medium
* Tags: array, sort, search, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number[][]} matrix
* @param {number} k
* @return {number}
*/
var kthSmallest = function(matrix, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Kth Smallest Element in a Sorted Matrix
* Difficulty: Medium
* Tags: array, sort, search, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function kthSmallest(matrix: number[][], k: number): number {

};
```

C# Solution:

```
/*
* Problem: Kth Smallest Element in a Sorted Matrix
* Difficulty: Medium
* Tags: array, sort, search, queue, heap
```

```
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int KthSmallest(int[][] matrix, int k) {

}
}
```

C Solution:

```
/*
* Problem: Kth Smallest Element in a Sorted Matrix
* Difficulty: Medium
* Tags: array, sort, search, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

int kthSmallest(int** matrix, int matrixSize, int* matrixColSize, int k) {

}
```

Go Solution:

```
// Problem: Kth Smallest Element in a Sorted Matrix
// Difficulty: Medium
// Tags: array, sort, search, queue, heap
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func kthSmallest(matrix [][]int, k int) int {

}
```

Kotlin Solution:

```
class Solution {
fun kthSmallest(matrix: Array<IntArray>, k: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func kthSmallest(_ matrix: [[Int]], _ k: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Kth Smallest Element in a Sorted Matrix
// Difficulty: Medium
// Tags: array, sort, search, queue, heap
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn kth_smallest(matrix: Vec<Vec<i32>>, k: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[][]} matrix
# @param {Integer} k
# @return {Integer}
def kth_smallest(matrix, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[][] $matrix
* @param Integer $k
* @return Integer
*/
function kthSmallest($matrix, $k) {

}
}
```

Dart Solution:

```
class Solution {
int kthSmallest(List<List<int>> matrix, int k) {

}
}
```

Scala Solution:

```
object Solution {
def kthSmallest(matrix: Array[Array[Int]], k: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec kth_smallest(matrix :: [[integer]], k :: integer) :: integer
def kth_smallest(matrix, k) do

end
end
```

Erlang Solution:

```
-spec kth_smallest(Matrix :: [[integer()]], K :: integer()) -> integer().
kth_smallest(Matrix, K) ->
.
```

Racket Solution:

```
(define/contract (kth-smallest matrix k)
(-> (listof (listof exact-integer?)) exact-integer? exact-integer?)
)
```