Problem 1439: Find the Kth Smallest Sum of a Matrix With Sorted Rows

Problem Information

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

Problem Description

You are given an

m x n

matrix

mat

that has its rows sorted in non-decreasing order and an integer

k

.

You are allowed to choose

exactly one element

from each row to form an array.

Return

the

k

th

smallest array sum among all possible arrays

.

Example 1:

Input:

mat = [[1,3,11],[2,4,6]], k = 5

Output:

7

Explanation:

Choosing one element from each row, the first k smallest sum are: [1,2], [1,4], [3,2], [3,4], [1,6]. Where the 5th sum is 7.

Example 2:

Input:

mat = [[1,3,11],[2,4,6]], k = 9

Output:

17

Example 3:

Input:

mat = [[1,10,10],[1,4,5],[2,3,6]], k = 7

Output:

9

Explanation:

Choosing one element from each row, the first k smallest sum are: [1,1,2], [1,1,3], [1,4,2], [1,4,3], [1,1,6], [1,5,2], [1,5,3]. Where the 7th sum is 9.

Constraints:

m == mat.length

n == mat.length[i]

1 <= m, n <= 40

1 <= mat[i][j] <= 5000

1 <= k <= min(200, n

m

)

mat[i]

is a non-decreasing array.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int kthSmallest(int** mat, int matSize, int* matColSize, int k) {

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

```
end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Find the Kth Smallest Sum of a Matrix With Sorted Rows
* Difficulty: Hard
* 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>>& mat, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Kth Smallest Sum of a Matrix With Sorted Rows
```

```
* Difficulty: Hard
* 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[][] mat, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Kth Smallest Sum of a Matrix With Sorted Rows
Difficulty: Hard
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, mat: List[List[int]], k: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Find the Kth Smallest Sum of a Matrix With Sorted Rows
* Difficulty: Hard
* 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[][]} mat
* @param {number} k
* @return {number}
*/
var kthSmallest = function(mat, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Kth Smallest Sum of a Matrix With Sorted Rows
* Difficulty: Hard
* 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(mat: number[][], k: number): number {

};
```

C# Solution:

```
/*
* Problem: Find the Kth Smallest Sum of a Matrix With Sorted Rows
* Difficulty: Hard
* 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[][] mat, int k) {

}
}
```

C Solution:

```
/*
* Problem: Find the Kth Smallest Sum of a Matrix With Sorted Rows
* Difficulty: Hard
* 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** mat, int matSize, int* matColSize, int k) {

}
```

Go Solution:

```
// Problem: Find the Kth Smallest Sum of a Matrix With Sorted Rows
// Difficulty: Hard
// 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(mat [][]int, k int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Find the Kth Smallest Sum of a Matrix With Sorted Rows
// Difficulty: Hard
// 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(mat: Vec<Vec<i32>>, k: i32) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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