Problem 363: Max Sum of Rectangle No Larger Than K

Problem Information

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

Problem Description

Given an

m x n

matrix

matrix

and an integer

k

, return

the max sum of a rectangle in the matrix such that its sum is no larger than

k

.

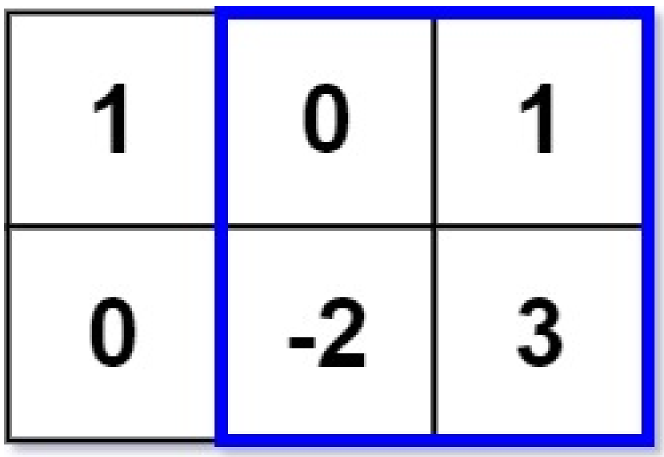
It is

guaranteed

that there will be a rectangle with a sum no larger than k

.

Example 1:



Input:

matrix = [[1,0,1],[0,-2,3]], k = 2

Output:

2

Explanation:

Because the sum of the blue rectangle [[0, 1], [-2, 3]] is 2, and 2 is the max number no larger than k (k = 2).

Example 2:

Input:

matrix = [[2,2,-1]], k = 3

Output:

3

Constraints:

m == matrix.length

n == matrix[i].length

1 <= m, n <= 100

-100 <= matrix[i][j] <= 100

-10

5

<= k <= 10

5

Follow up:

What if the number of rows is much larger than the number of columns?

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

```
end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (max-sum-submatrix matrix k)
(-> (listof (listof exact-integer?)) exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Max Sum of Rectangle No Larger Than K
* Difficulty: Hard
* Tags: array, search
*
* 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 maxSumSubmatrix(vector<vector<int>>& matrix, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Max Sum of Rectangle No Larger Than K
* Difficulty: Hard
* Tags: array, search
*
```

```
* 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 maxSumSubmatrix(int[][] matrix, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Max Sum of Rectangle No Larger Than K
Difficulty: Hard
Tags: array, search

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Max Sum of Rectangle No Larger Than K
* Difficulty: Hard
* Tags: array, search
*
* 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 maxSumSubmatrix = function(matrix, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Max Sum of Rectangle No Larger Than K
* Difficulty: Hard
* Tags: array, search
*
* 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 maxSumSubmatrix(matrix: number[][], k: number): number {

};
```

C# Solution:

```
/*
* Problem: Max Sum of Rectangle No Larger Than K
* Difficulty: Hard
* Tags: array, search
*
```

```
* 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 MaxSumSubmatrix(int[][] matrix, int k) {

}
}
```

C Solution:

```
/*
* Problem: Max Sum of Rectangle No Larger Than K
* Difficulty: Hard
* Tags: array, search
*
* 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 maxSumSubmatrix(int** matrix, int matrixSize, int* matrixColSize, int k)
{

}
```

Go Solution:

```
// Problem: Max Sum of Rectangle No Larger Than K
// Difficulty: Hard
// Tags: array, search
//
// 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 maxSumSubmatrix(matrix [][]int, k int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Max Sum of Rectangle No Larger Than K
// Difficulty: Hard
// Tags: array, search
//
// 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 max_sum_submatrix(matrix: Vec<Vec<i32>>, k: i32) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (max-sum-submatrix matrix k)
(-> (listof (listof exact-integer?)) exact-integer? exact-integer?)
)
```