Problem 3344: Maximum Sized Array

Problem Information

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

Problem Description

Given a positive integer

s

, let

A

be a 3D array of dimensions

n × n × n

, where each element

A[i][j][k]

is defined as:

A[i][j][k] = i * (j OR k)

, where

0 <= i, j, k < n

.

Return the

maximum

possible value of

n

such that the

sum

of all elements in array

A

does not exceed

s

.

Example 1:

Input:

s = 10

Output:

2

Explanation:

Elements of the array

A

for

n = 2

:

A[0][0][0] = 0 * (0 OR 0) = 0

A[0][0][1] = 0 * (0 OR 1) = 0

A[0][1][0] = 0 * (1 OR 0) = 0

A[0][1][1] = 0 * (1 OR 1) = 0

A[1][0][0] = 1 * (0 OR 0) = 0

A[1][0][1] = 1 * (0 OR 1) = 1

A[1][1][0] = 1 * (1 OR 0) = 1

A[1][1][1] = 1 * (1 OR 1) = 1

The total sum of the elements in array

A

is 3, which does not exceed 10, so the maximum possible value of

n

is 2.

Example 2:

Input:

s = 0

Output:

1

Explanation:

Elements of the array

A

for

n = 1

:

A[0][0][0] = 0 * (0 OR 0) = 0

The total sum of the elements in array

A

is 0, which does not exceed 0, so the maximum possible value of

n

is 1.

Constraints:

0 <= s <= 10

15

Code Snippets

C++:

```
class Solution {
public:
```

```
int maxSizedArray(long long s) {

}
};
```

Java:

```
class Solution {
public int maxSizedArray(long s) {

}
}
```

Python3:

```
class Solution:
def maxSizedArray(self, s: int) -> int:
```

Python:

```
class Solution(object):
def maxSizedArray(self, s):
"""
:type s: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} s
* @return {number}
*/
var maxSizedArray = function(s) {

};
```

TypeScript:

```
function maxSizedArray(s: number): number {

};
```

```
}
```

C#:

```
public class Solution {
public int MaxSizedArray(long s) {

}
}
```

C:

```
int maxSizedArray(long long s) {

}
```

Go:

```
func maxSizedArray(s int64) int {

}
```

Kotlin:

```
class Solution {
fun maxSizedArray(s: Long): Int {

}
}
```

Swift:

```
class Solution {
func maxSizedArray(_ s: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn max_sized_array(s: i64) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} s
# @return {Integer}
def max_sized_array(s)

end
```

PHP:

```
class Solution {

/**
* @param Integer $s
* @return Integer
*/
function maxSizedArray($s) {

}
}
```

Dart:

```
class Solution {
int maxSizedArray(int s) {

}
}
```

Scala:

```
object Solution {
def maxSizedArray(s: Long): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec max_sized_array(s :: integer) :: integer
def max_sized_array(s) do
```

```
end
end
```

Erlang:

```
-spec max_sized_array(S :: integer()) -> integer().
max_sized_array(S) ->
.
```

Racket:

```
(define/contract (max-sized-array s)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Sized Array
* Difficulty: Medium
* 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 maxSizedArray(long long s) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Sized Array
* Difficulty: Medium
* 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 maxSizedArray(long s) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Sized Array
Difficulty: Medium
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 maxSizedArray(self, s: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def maxSizedArray(self, s):
"""
:type s: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Maximum Sized Array
* Difficulty: Medium
* 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} s
* @return {number}
*/
var maxSizedArray = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Sized Array
* Difficulty: Medium
* 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 maxSizedArray(s: number): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Sized Array
* Difficulty: Medium
* 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 MaxSizedArray(long s) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Sized Array
* Difficulty: Medium
* 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 maxSizedArray(long long s) {

}
```

Go Solution:

```
// Problem: Maximum Sized Array
// Difficulty: Medium
// 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 maxSizedArray(s int64) int {

}
```

Kotlin Solution:

```
class Solution {
fun maxSizedArray(s: Long): Int {

}
}
```

Swift Solution:

```
class Solution {
func maxSizedArray(_ s: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Maximum Sized Array
// Difficulty: Medium
// 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_sized_array(s: i64) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} s
# @return {Integer}
def max_sized_array(s)

end
```

PHP Solution:

```
class Solution {
```

```
/**
* @param Integer $s
* @return Integer
*/
function maxSizedArray($s) {

}
}
```

Dart Solution:

```
class Solution {
int maxSizedArray(int s) {

}
}
```

Scala Solution:

```
object Solution {
def maxSizedArray(s: Long): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec max_sized_array(s :: integer) :: integer
def max_sized_array(s) do

end
end
```

Erlang Solution:

```
-spec max_sized_array(S :: integer()) -> integer().
max_sized_array(S) ->
.
```

Racket Solution:

```
(define/contract (max-sized-array s)
(-> exact-integer? exact-integer?)
)
```