Problem 1802: Maximum Value at a Given Index in a Bounded Array

Problem Information

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

Problem Description

You are given three positive integers:

n

,

index

, and

maxSum

. You want to construct an array

nums

(

0-indexed

)

that satisfies the following conditions:

nums.length == n

nums[i]

is a

positive

integer where

0 <= i < n

.

abs(nums[i] - nums[i+1]) <= 1

where

0 <= i < n-1

.

The sum of all the elements of

nums

does not exceed

maxSum

.

nums[index]

is

maximized

.

Return

nums[index]

of the constructed array

.

Note that

abs(x)

equals

x

if

x >= 0

, and

-x

otherwise.

Example 1:

Input:

n = 4, index = 2, maxSum = 6

Output:

2

Explanation:

nums = [1,2,

2

,1] is one array that satisfies all the conditions. There are no arrays that satisfy all the conditions and have nums[2] == 3, so 2 is the maximum nums[2].

Example 2:

Input:

n = 6, index = 1, maxSum = 10

Output:

3

Constraints:

1 <= n <= maxSum <= 10

9

0 <= index < n

Code Snippets

C++:

```
class Solution {
public:
int maxValue(int n, int index, int maxSum) {

}
};
```

Java:

```
class Solution {
public int maxValue(int n, int index, int maxSum) {
```

```
}
}
```

Python3:

```
class Solution:
def maxValue(self, n: int, index: int, maxSum: int) -> int:
```

Python:

```
class Solution(object):
def maxValue(self, n, index, maxSum):
"""
:type n: int
:type index: int
:type maxSum: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} n
* @param {number} index
* @param {number} maxSum
* @return {number}
*/
var maxValue = function(n, index, maxSum) {

};
```

TypeScript:

```
function maxValue(n: number, index: number, maxSum: number): number {

};
```

C#:

```
public class Solution {
public int MaxValue(int n, int index, int maxSum) {
```

```
}
}
```

C:

```
int maxValue(int n, int index, int maxSum) {

}
```

Go:

```
func maxValue(n int, index int, maxSum int) int {

}
```

Kotlin:

```
class Solution {
fun maxValue(n: Int, index: Int, maxSum: Int): Int {

}
}
```

Swift:

```
class Solution {
func maxValue(_ n: Int, _ index: Int, _ maxSum: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn max_value(n: i32, index: i32, max_sum: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} n
# @param {Integer} index
# @param {Integer} max_sum
# @return {Integer}
def max_value(n, index, max_sum)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @param Integer $index
* @param Integer $maxSum
* @return Integer
*/
function maxValue($n, $index, $maxSum) {

}
}
```

Dart:

```
class Solution {
int maxValue(int n, int index, int maxSum) {

}
}
```

Scala:

```
object Solution {
def maxValue(n: Int, index: Int, maxSum: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec max_value(n :: integer, index :: integer, max_sum :: integer) ::
```

```
integer
def max_value(n, index, max_sum) do

end
end
```

Erlang:

```
-spec max_value(N :: integer(), Index :: integer(), MaxSum :: integer()) ->
integer().
max_value(N, Index, MaxSum) ->
.
```

Racket:

```
(define/contract (max-value n index maxSum)
(-> exact-integer? exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Value at a Given Index in a Bounded Array
* Difficulty: Medium
* Tags: array, greedy, math, 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 maxValue(int n, int index, int maxSum) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Value at a Given Index in a Bounded Array
* Difficulty: Medium
* Tags: array, greedy, math, 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 maxValue(int n, int index, int maxSum) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Value at a Given Index in a Bounded Array
Difficulty: Medium
Tags: array, greedy, math, 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 maxValue(self, n: int, index: int, maxSum: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def maxValue(self, n, index, maxSum):
"""
:type n: int
:type index: int
:type maxSum: int
:rtype: int
```

```
"""
```

JavaScript Solution:

```
/**
* Problem: Maximum Value at a Given Index in a Bounded Array
* Difficulty: Medium
* Tags: array, greedy, math, 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} n
* @param {number} index
* @param {number} maxSum
* @return {number}
*/
var maxValue = function(n, index, maxSum) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Value at a Given Index in a Bounded Array
* Difficulty: Medium
* Tags: array, greedy, math, 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 maxValue(n: number, index: number, maxSum: number): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Value at a Given Index in a Bounded Array
* Difficulty: Medium
* Tags: array, greedy, math, 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 MaxValue(int n, int index, int maxSum) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Value at a Given Index in a Bounded Array
* Difficulty: Medium
* Tags: array, greedy, math, 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 maxValue(int n, int index, int maxSum) {

}
```

Go Solution:

```
// Problem: Maximum Value at a Given Index in a Bounded Array
// Difficulty: Medium
// Tags: array, greedy, math, 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 maxValue(n int, index int, maxSum int) int {

}
```

Kotlin Solution:

```
class Solution {
fun maxValue(n: Int, index: Int, maxSum: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func maxValue(_ n: Int, _ index: Int, _ maxSum: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Maximum Value at a Given Index in a Bounded Array
// Difficulty: Medium
// Tags: array, greedy, math, 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_value(n: i32, index: i32, max_sum: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @param {Integer} index
# @param {Integer} max_sum
```

```
# @return {Integer}
def max_value(n, index, max_sum)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @param Integer $index
* @param Integer $maxSum
* @return Integer
*/
function maxValue($n, $index, $maxSum) {

}
}
```

Dart Solution:

```
class Solution {
int maxValue(int n, int index, int maxSum) {

}
}
```

Scala Solution:

```
object Solution {
def maxValue(n: Int, index: Int, maxSum: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec max_value(n :: integer, index :: integer, max_sum :: integer) ::
integer
```

```
def max_value(n, index, max_sum) do

end
end
```

Erlang Solution:

```
-spec max_value(N :: integer(), Index :: integer(), MaxSum :: integer()) ->
integer().
max_value(N, Index, MaxSum) ->
.
```

Racket Solution:

```
(define/contract (max-value n index maxSum)
(-> exact-integer? exact-integer? exact-integer? exact-integer?)
)
```