Problem 2599: Make the Prefix Sum Non-negative

Problem Information

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

Problem Description

You are given a

0-indexed

integer array

nums

. You can apply the following operation any number of times:

Pick any element from

nums

and put it at the end of

nums

.

The prefix sum array of

nums

is an array

prefix

of the same length as

nums

such that

prefix[i]

is the sum of all the integers

nums[j]

where

j

is in the inclusive range

[0, i]

.

Return

the minimum number of operations such that the prefix sum array does not contain negative integers

. The test cases are generated such that it is always possible to make the prefix sum array non-negative.

Example 1:

Input:

nums = [2,3,-5,4]

Output:

0

Explanation:

we do not need to do any operations. The array is [2,3,-5,4]. The prefix sum array is [2, 5, 0, 4].

Example 2:

Input:

nums = [3,-5,-2,6]

Output:

1

Explanation:

we can do one operation on index 1. The array after the operation is [3,-2,6,-5]. The prefix sum array is [3, 1, 7, 2].

Constraints:

1 <= nums.length <= 10

5

-10

9

<= nums[i] <= 10

9

Code Snippets

C++:

```
class Solution {
public:
int makePrefSumNonNegative(vector<int>& nums) {

}
};
```

Java:

```
class Solution {
public int makePrefSumNonNegative(int[] nums) {

}
}
```

Python3:

```
class Solution:
def makePrefSumNonNegative(self, nums: List[int]) -> int:
```

Python:

```
class Solution(object):
def makePrefSumNonNegative(self, nums):
"""
:type nums: List[int]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} nums
* @return {number}
*/
var makePrefSumNonNegative = function(nums) {

};
```

TypeScript:

```
function makePrefSumNonNegative(nums: number[]): number {

};
```

C#:

```
public class Solution {
public int MakePrefSumNonNegative(int[] nums) {

}
}
```

C:

```
int makePrefSumNonNegative(int* nums, int numsSize) {

}
```

Go:

```
func makePrefSumNonNegative(nums []int) int {

}
```

Kotlin:

```
class Solution {
fun makePrefSumNonNegative(nums: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func makePrefSumNonNegative(_ nums: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn make_pref_sum_non_negative(nums: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @return {Integer}
def make_pref_sum_non_negative(nums)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer
*/
function makePrefSumNonNegative($nums) {

}
}
```

Dart:

```
class Solution {
int makePrefSumNonNegative(List<int> nums) {

}
}
```

Scala:

```
object Solution {
def makePrefSumNonNegative(nums: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec make_pref_sum_non_negative(nums :: [integer]) :: integer
def make_pref_sum_non_negative(nums) do

end
end
```

Erlang:

```
-spec make_pref_sum_non_negative(Nums :: [integer()]) -> integer().
make_pref_sum_non_negative(Nums) ->
.
```

Racket:

```
(define/contract (make-pref-sum-non-negative nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Make the Prefix Sum Non-negative
* Difficulty: Medium
* Tags: array, greedy, 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 makePrefSumNonNegative(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Make the Prefix Sum Non-negative
* Difficulty: Medium
* Tags: array, greedy, 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 makePrefSumNonNegative(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Make the Prefix Sum Non-negative
Difficulty: Medium
Tags: array, greedy, 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 makePrefSumNonNegative(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def makePrefSumNonNegative(self, nums):
"""
:type nums: List[int]
:rtype: int
```

```
"""
```

JavaScript Solution:

```
/**
* Problem: Make the Prefix Sum Non-negative
* Difficulty: Medium
* Tags: array, greedy, 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[]} nums
* @return {number}
*/
var makePrefSumNonNegative = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Make the Prefix Sum Non-negative
* Difficulty: Medium
* Tags: array, greedy, 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 makePrefSumNonNegative(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Make the Prefix Sum Non-negative
* Difficulty: Medium
* Tags: array, greedy, 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 MakePrefSumNonNegative(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Make the Prefix Sum Non-negative
* Difficulty: Medium
* Tags: array, greedy, 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 makePrefSumNonNegative(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Make the Prefix Sum Non-negative
// Difficulty: Medium
// Tags: array, greedy, 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 makePrefSumNonNegative(nums []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun makePrefSumNonNegative(nums: IntArray): Int {

}
}
```

Swift Solution:

```
class Solution {
func makePrefSumNonNegative(_ nums: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Make the Prefix Sum Non-negative
// Difficulty: Medium
// Tags: array, greedy, 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 make_pref_sum_non_negative(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @return {Integer}
def make_pref_sum_non_negative(nums)
```

```
end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer
*/
function makePrefSumNonNegative($nums) {

}
}
```

Dart Solution:

```
class Solution {
int makePrefSumNonNegative(List<int> nums) {

}
}
```

Scala Solution:

```
object Solution {
def makePrefSumNonNegative(nums: Array[Int]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec make_pref_sum_non_negative(nums :: [integer]) :: integer
def make_pref_sum_non_negative(nums) do

end
end
```

Erlang Solution:

```
-spec make_pref_sum_non_negative(Nums :: [integer()]) -> integer().
make_pref_sum_non_negative(Nums) ->
.
```

Racket Solution:

```
(define/contract (make-pref-sum-non-negative nums)
(-> (listof exact-integer?) exact-integer?)
)
```