# Problem 1526: Minimum Number of Increments on Subarrays to Form a Target Array

## Problem Information

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

## Problem Description

You are given an integer array

target

. You have an integer array

initial

of the same size as

target

with all elements initially zeros.

In one operation you can choose

any

subarray from

initial

and increment each value by one.

Return

the minimum number of operations to form a

target

array from

initial

.

The test cases are generated so that the answer fits in a 32-bit integer.

Example 1:

Input:

target = [1,2,3,2,1]

Output:

3

Explanation:

We need at least 3 operations to form the target array from the initial array. [

0,0,0,0,0

] increment 1 from index 0 to 4 (inclusive). [1,

1,1,1

,1] increment 1 from index 1 to 3 (inclusive). [1,2,

2

,2,1] increment 1 at index 2. [1,2,3,2,1] target array is formed.

Example 2:

Input:

target = [3,1,1,2]

Output:

4

Explanation:

[

0,0,0,0

] -> [1,1,1,

1

] -> [

1

,1,1,2] -> [

2

,1,1,2] -> [3,1,1,2]

Example 3:

Input:

target = [3,1,5,4,2]

Output:

7

Explanation:

[

0,0,0,0,0

] -> [

1

,1,1,1,1] -> [

2

,1,1,1,1] -> [3,1,

1,1,1

] -> [3,1,

2,2

,2] -> [3,1,

3,3

,2] -> [3,1,

4

,4,2] -> [3,1,5,4,2].

Constraints:

1 <= target.length <= 10

5

1 <= target[i] <= 10

5

The input is generated such that the answer fits inside a 32 bit integer.

## Code Snippets

**C++:**

```
class Solution {
public:
int minNumberOperations(vector<int>& target) {

}
};
```

**Java:**

```
class Solution {
public int minNumberOperations(int[] target) {

}
}
```

**Python3:**

```
class Solution:
def minNumberOperations(self, target: List[int]) -> int:
```

**Python:**

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

**JavaScript:**

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

};
```

**TypeScript:**

```
function minNumberOperations(target: number[]): number {

};
```

**C#:**

```
public class Solution {
public int MinNumberOperations(int[] target) {

}
}
```

**C:**

```
int minNumberOperations(int* target, int targetSize) {

}
```

**Go:**

```
func minNumberOperations(target []int) int {

}
```

**Kotlin:**

```
class Solution {
fun minNumberOperations(target: IntArray): Int {

}
}
```

**Swift:**

```
class Solution {
func minNumberOperations(_ target: [Int]) -> Int {

}
}
```

**Rust:**

```
impl Solution {
pub fn min_number_operations(target: Vec<i32>) -> i32 {

}
}
```

**Ruby:**

```
# @param {Integer[]} target
# @return {Integer}
def min_number_operations(target)

end
```

**PHP:**

```
class Solution {

/**
* @param Integer[] $target
* @return Integer
*/
function minNumberOperations($target) {

}
}
```

**Dart:**

```
class Solution {
int minNumberOperations(List<int> target) {

}
}
```

### Scala:

```
object Solution {
def minNumberOperations(target: Array[Int]): Int = {

}
}
```

### Elixir:

```
defmodule Solution do
@spec min_number_operations(target :: [integer]) :: integer
def min_number_operations(target) do

end
end
```

### Erlang:

```
-spec min_number_operations(Target :: [integer()]) -> integer().
min_number_operations(Target) ->
.
```

### Racket:

```
(define/contract (min-number-operations target)
(-> (listof exact-integer?) exact-integer?)
)
```

## Solutions

### C++ Solution:

```
/*
* Problem: Minimum Number of Increments on Subarrays to Form a Target Array
* Difficulty: Hard
* Tags: array, dp, greedy, stack
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/
```

```
class Solution {
public:
int minNumberOperations(vector<int>& target) {

}
};
```

### Java Solution:

```
/**
* Problem: Minimum Number of Increments on Subarrays to Form a Target Array
* Difficulty: Hard
* Tags: array, dp, greedy, stack
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int minNumberOperations(int[] target) {

}
}
```

### Python3 Solution:

```
"""
Problem: Minimum Number of Increments on Subarrays to Form a Target Array
Difficulty: Hard
Tags: array, dp, greedy, stack

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

class Solution:
def minNumberOperations(self, target: List[int]) -> int:
# TODO: Implement optimized solution
```

```
pass
```

### Python Solution:

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

### JavaScript Solution:

```
/**
* Problem: Minimum Number of Increments on Subarrays to Form a Target Array
* Difficulty: Hard
* Tags: array, dp, greedy, stack
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {number[]} target
* @return {number}
*/
var minNumberOperations = function(target) {

};
```

### TypeScript Solution:

```
/**
* Problem: Minimum Number of Increments on Subarrays to Form a Target Array
* Difficulty: Hard
* Tags: array, dp, greedy, stack
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
```

```
*/

function minNumberOperations(target: number[]): number {

};
```

**C# Solution:**

```
/*
* Problem: Minimum Number of Increments on Subarrays to Form a Target Array
* Difficulty: Hard
* Tags: array, dp, greedy, stack
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public int MinNumberOperations(int[] target) {

}
}
```

**C Solution:**

```
/*
* Problem: Minimum Number of Increments on Subarrays to Form a Target Array
* Difficulty: Hard
* Tags: array, dp, greedy, stack
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

int minNumberOperations(int* target, int targetSize) {

}
```

**Go Solution:**

```
// Problem: Minimum Number of Increments on Subarrays to Form a Target Array
// Difficulty: Hard
// Tags: array, dp, greedy, stack
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func minNumberOperations(target []int) int {

}
```

### Kotlin Solution:

```
class Solution {
fun minNumberOperations(target: IntArray): Int {

}
}
```

### Swift Solution:

```
class Solution {
func minNumberOperations(_ target: [Int]) -> Int {

}
}
```

### Rust Solution:

```
// Problem: Minimum Number of Increments on Subarrays to Form a Target Array
// Difficulty: Hard
// Tags: array, dp, greedy, stack
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn min_number_operations(target: Vec<i32>) -> i32 {

}
```

```
}
```

### Ruby Solution:

```
# @param {Integer[]} target
# @return {Integer}
def min_number_operations(target)

end
```

### PHP Solution:

```
class Solution {

/**
* @param Integer[] $target
* @return Integer
*/
function minNumberOperations($target) {

}
}
```

### Dart Solution:

```
class Solution {
int minNumberOperations(List<int> target) {

}
}
```

### Scala Solution:

```
object Solution {
def minNumberOperations(target: Array[Int]): Int = {

}
}
```

### Elixir Solution:

```
defmodule Solution do
@spec min_number_operations(target :: [integer]) :: integer
def min_number_operations(target) do

end
end
```

### Erlang Solution:

```
-spec min_number_operations(Target :: [integer()]) -> integer().
min_number_operations(Target) ->
.
```

### Racket Solution:

```
(define/contract (min-number-operations target)
(-> (listof exact-integer?) exact-integer?)
)
```