Problem 3444: Minimum Increments for Target Multiples in an Array

Problem Information

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

Problem Description

You are given two arrays,

nums

and

target

.

In a single operation, you may increment any element of

nums

by 1.

Return

the minimum number

of operations required so that each element in

target

has

at least

one multiple in

nums

.

Example 1:

Input:

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

Output:

1

Explanation:

The minimum number of operations required to satisfy the condition is 1.

Increment 3 to 4 with just one operation, making 4 a multiple of itself.

Example 2:

Input:

nums = [8,4], target = [10,5]

Output:

2

Explanation:

The minimum number of operations required to satisfy the condition is 2.

Increment 8 to 10 with 2 operations, making 10 a multiple of both 5 and 10.

Example 3:

Input:

nums = [7,9,10], target = [7]

Output:

0

Explanation:

Target 7 already has a multiple in nums, so no additional operations are needed.

Constraints:

1 <= nums.length <= 5 * 10

4

1 <= target.length <= 4

target.length <= nums.length

1 <= nums[i], target[i] <= 10

4

Code Snippets

C++:

```
class Solution {
public:
int minimumIncrements(vector<int>& nums, vector<int>& target) {
```

```
}
};
```

Java:

```
class Solution {
public int minimumIncrements(int[] nums, int[] target) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function minimumIncrements(nums: number[], target: number[]): number {

};
```

C#:

```
public class Solution {
public int MinimumIncrements(int[] nums, int[] target) {

}
}
```

C:

```
int minimumIncrements(int* nums, int numsSize, int* target, int targetSize) {

}
```

Go:

```
func minimumIncrements(nums []int, target []int) int {

}
```

Kotlin:

```
class Solution {
fun minimumIncrements(nums: IntArray, target: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func minimumIncrements(_ nums: [Int], _ target: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn minimum_increments(nums: Vec<i32>, target: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @param {Integer[]} target
# @return {Integer}
def minimum_increments(nums, target)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer[] $target
* @return Integer
*/
function minimumIncrements($nums, $target) {

}
}
```

Dart:

```
class Solution {
int minimumIncrements(List<int> nums, List<int> target) {

}
}
```

Scala:

```
object Solution {
def minimumIncrements(nums: Array[Int], target: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec minimum_increments(nums :: [integer], target :: [integer]) :: integer
```

```
def minimum_increments(nums, target) do

end
end
```

Erlang:

```
-spec minimum_increments(Nums :: [integer()], Target :: [integer()]) ->
integer().
minimum_increments(Nums, Target) ->
.
```

Racket:

```
(define/contract (minimum-increments nums target)
(-> (listof exact-integer?) (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Increments for Target Multiples in an Array
* Difficulty: Hard
* Tags: array, dp, math
*
* 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 minimumIncrements(vector<int>& nums, vector<int>& target) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Increments for Target Multiples in an Array
* Difficulty: Hard
* Tags: array, dp, math
*
* 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 minimumIncrements(int[] nums, int[] target) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Increments for Target Multiples in an Array
Difficulty: Hard
Tags: array, dp, math

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 minimumIncrements(self, nums: List[int], target: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Minimum Increments for Target Multiples in an Array
* Difficulty: Hard
* Tags: array, dp, math
*
* 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[]} nums
* @param {number[]} target
* @return {number}
*/
var minimumIncrements = function(nums, target) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Increments for Target Multiples in an Array
* Difficulty: Hard
* Tags: array, dp, math
*
* 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 minimumIncrements(nums: number[], target: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Increments for Target Multiples in an Array
* Difficulty: Hard
```

```
* Tags: array, dp, math
*
* 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 MinimumIncrements(int[] nums, int[] target) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Increments for Target Multiples in an Array
* Difficulty: Hard
* Tags: array, dp, math
*
* 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 minimumIncrements(int* nums, int numsSize, int* target, int targetSize) {

}
```

Go Solution:

```
// Problem: Minimum Increments for Target Multiples in an Array
// Difficulty: Hard
// Tags: array, dp, math
//
// 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 minimumIncrements(nums []int, target []int) int {
```

```
}
```

Kotlin Solution:

```
class Solution {
fun minimumIncrements(nums: IntArray, target: IntArray): Int {

}
}
```

Swift Solution:

```
class Solution {
func minimumIncrements(_ nums: [Int], _ target: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Minimum Increments for Target Multiples in an Array
// Difficulty: Hard
// Tags: array, dp, math
//
// 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 minimum_increments(nums: Vec<i32>, target: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @param {Integer[]} target
# @return {Integer}
def minimum_increments(nums, target)
```

```
end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer[] $target
* @return Integer
*/
function minimumIncrements($nums, $target) {

}
}
```

Dart Solution:

```
class Solution {
int minimumIncrements(List<int> nums, List<int> target) {

}
}
```

Scala Solution:

```
object Solution {
def minimumIncrements(nums: Array[Int], target: Array[Int]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec minimum_increments(nums :: [integer], target :: [integer]) :: integer
def minimum_increments(nums, target) do

end
end
```

Erlang Solution:

```
-spec minimum_increments(Nums :: [integer()], Target :: [integer()]) ->
integer().
minimum_increments(Nums, Target) ->
.
```

Racket Solution:

```
(define/contract (minimum-increments nums target)
(-> (listof exact-integer?) (listof exact-integer?) exact-integer?)
)
```