Problem 1714: Sum Of Special Evenly-Spaced Elements In Array

Problem Information

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

Problem Description

You are given a

0-indexed

integer array

nums

consisting of

n

non-negative integers.

You are also given an array

queries

, where

queries[i] = [x

i

, y

i

]

. The answer to the

i

th

query is the sum of all

nums[j]

where

x

i

<= j < n

and

(j - x

i

)

is divisible by

y

i

.

Return

an array

answer

where

answer.length == queries.length

and

answer[i]

is the answer to the

i

th

query

modulo

10

9

+ 7

.

Example 1:

Input:

nums = [0,1,2,3,4,5,6,7], queries = [[0,3],[5,1],[4,2]]

Output:

[9,18,10]

Explanation:

The answers of the queries are as follows: 1) The j indices that satisfy this query are 0, 3, and 6. nums[0] + nums[3] + nums[6] = 9 2) The j indices that satisfy this query are 5, 6, and 7. nums[5] + nums[6] + nums[7] = 18 3) The j indices that satisfy this query are 4 and 6. nums[4] + nums[6] = 10

Example 2:

Input:

nums = [100,200,101,201,102,202,103,203], queries = [[0,7]]

Output:

[303]

Constraints:

n == nums.length

1 <= n <= 5 * 10

4

0 <= nums[i] <= 10

9

1 <= queries.length <= 1.5 * 10

5

0 <= x

i

< n

1 <= y

i

<= 5 * 10

4

Code Snippets

C++:

```
class Solution {
public:
vector<int> solve(vector<int>& nums, vector<vector<int>>& queries) {

}
};
```

Java:

```
class Solution {
public int[] solve(int[] nums, int[][] queries) {

}
}
```

Python3:

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

Python:

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

```
:type queries: List[List[int]]
:rtype: List[int]
"""
```

JavaScript:

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

};
```

TypeScript:

```
function solve(nums: number[], queries: number[][]): number[] {

};
```

C#:

```
public class Solution {
public int[] Solve(int[] nums, int[][] queries) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* solve(int* nums, int numsSize, int** queries, int queriesSize, int*
queriesColSize, int* returnSize) {

}
```

Go:

```
func solve(nums []int, queries [][]int) []int {

}
```

Kotlin:

```
class Solution {
fun solve(nums: IntArray, queries: Array<IntArray>): IntArray {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @param {Integer[][]} queries
# @return {Integer[]}
def solve(nums, queries)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
```

```
* @param Integer[][] $queries
* @return Integer[]
*/
function solve($nums, $queries) {

}
}
```

Dart:

```
class Solution {
List<int> solve(List<int> nums, List<List<int>> queries) {

}
}
```

Scala:

```
object Solution {
def solve(nums: Array[Int], queries: Array[Array[Int]]): Array[Int] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec solve(nums :: [integer], queries :: [[integer]]) :: [integer]
def solve(nums, queries) do

end
end
```

Erlang:

```
-spec solve(Nums :: [integer()], Queries :: [[integer()]]) -> [integer()].
solve(Nums, Queries) ->
.
```

Racket:

```
(define/contract (solve nums queries)
(-> (listof exact-integer?) (listof (listof exact-integer?)) (listof
exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Sum Of Special Evenly-Spaced Elements In Array
* Difficulty: Hard
* Tags: array, dp
*
* 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:
vector<int> solve(vector<int>& nums, vector<vector<int>>& queries) {

}
};
```

Java Solution:

```
/**
* Problem: Sum Of Special Evenly-Spaced Elements In Array
* Difficulty: Hard
* Tags: array, dp
*
* 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[] solve(int[] nums, int[][] queries) {
```

```
}
}
```

Python3 Solution:

```
"""
Problem: Sum Of Special Evenly-Spaced Elements In Array
Difficulty: Hard
Tags: array, dp

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Sum Of Special Evenly-Spaced Elements In Array
* Difficulty: Hard
* Tags: array, dp
*
* 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[][]} queries
* @return {number[]}
*/
var solve = function(nums, queries) {

};
```

TypeScript Solution:

```
/**
* Problem: Sum Of Special Evenly-Spaced Elements In Array
* Difficulty: Hard
* Tags: array, dp
*
* 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 solve(nums: number[], queries: number[][]): number[] {

};
```

C# Solution:

```
/*
* Problem: Sum Of Special Evenly-Spaced Elements In Array
* Difficulty: Hard
* Tags: array, dp
*
* 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[] Solve(int[] nums, int[][] queries) {
```

```
}
}
```

C Solution:

```
/*
* Problem: Sum Of Special Evenly-Spaced Elements In Array
* Difficulty: Hard
* Tags: array, dp
*
* 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
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* solve(int* nums, int numsSize, int** queries, int queriesSize, int*
queriesColSize, int* returnSize) {

}
```

Go Solution:

```
// Problem: Sum Of Special Evenly-Spaced Elements In Array
// Difficulty: Hard
// Tags: array, dp
//
// 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 solve(nums []int, queries [][]int) []int {

}
```

Kotlin Solution:

```
class Solution {
fun solve(nums: IntArray, queries: Array<IntArray>): IntArray {
```

```
}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Sum Of Special Evenly-Spaced Elements In Array
// Difficulty: Hard
// Tags: array, dp
//
// 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 solve(nums: Vec<i32>, queries: Vec<Vec<i32>>) -> Vec<i32> {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @param {Integer[][]} queries
# @return {Integer[]}
def solve(nums, queries)

end
```

PHP Solution:

```
class Solution {
```

```
/**
* @param Integer[] $nums
* @param Integer[][] $queries
* @return Integer[]
*/
function solve($nums, $queries) {

}
}
```

Dart Solution:

```
class Solution {
List<int> solve(List<int> nums, List<List<int>> queries) {

}
}
```

Scala Solution:

```
object Solution {
def solve(nums: Array[Int], queries: Array[Array[Int]]): Array[Int] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec solve(nums :: [integer], queries :: [[integer]]) :: [integer]
def solve(nums, queries) do

end
end
```

Erlang Solution:

```
-spec solve(Nums :: [integer()], Queries :: [[integer()]]) -> [integer()].
solve(Nums, Queries) ->
.
```

Racket Solution:

```
(define/contract (solve nums queries)
(-> (listof exact-integer?) (listof (listof exact-integer?)) (listof
exact-integer?))
)
```