Problem 1524: Number of Sub-arrays With Odd Sum

Problem Information

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

Problem Description

Given an array of integers

arr

, return

the number of subarrays with an

odd

sum

.

Since the answer can be very large, return it modulo

10

9

+ 7

.

Example 1:

Input:

arr = [1,3,5]

Output:

4

Explanation:

All subarrays are [[1],[1,3],[1,3,5],[3],[3,5],[5]] All sub-arrays sum are [1,4,9,3,8,5]. Odd sums are [1,9,3,5] so the answer is 4.

Example 2:

Input:

arr = [2,4,6]

Output:

0

Explanation:

All subarrays are [[2],[2,4],[2,4,6],[4],[4,6],[6]] All sub-arrays sum are [2,6,12,4,10,6]. All sub-arrays have even sum and the answer is 0.

Example 3:

Input:

arr = [1,2,3,4,5,6,7]

Output:

16

Constraints:

1 <= arr.length <= 10

5

1 <= arr[i] <= 100

Code Snippets

C++:

```
class Solution {
public:
int numOfSubarrays(vector<int>& arr) {

}
};
```

Java:

```
class Solution {
public int numOfSubarrays(int[] arr) {

}
}
```

Python3:

```
class Solution:
def numOfSubarrays(self, arr: List[int]) -> int:
```

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function numOfSubarrays(arr: number[]): number {

};
```

C#:

```
public class Solution {
public int NumOfSubarrays(int[] arr) {

}
}
```

C:

```
int numOfSubarrays(int* arr, int arrSize) {

}
```

Go:

```
func numOfSubarrays(arr []int) int {

}
```

Kotlin:

```
class Solution {
fun numOfSubarrays(arr: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func numOfSubarrays(_ arr: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn num_of_subarrays(arr: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} arr
# @return {Integer}
def num_of_subarrays(arr)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $arr
* @return Integer
*/
function numOfSubarrays($arr) {

}
}
```

Dart:

```
class Solution {
int numOfSubarrays(List<int> arr) {

}
}
```

Scala:

```
object Solution {
def numOfSubarrays(arr: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec num_of_subarrays(arr :: [integer]) :: integer
def num_of_subarrays(arr) do

end
end
```

Erlang:

```
-spec num_of_subarrays(Arr :: [integer()]) -> integer().
num_of_subarrays(Arr) ->
.
```

Racket:

```
(define/contract (num-of-subarrays arr)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Sub-arrays With Odd Sum
* Difficulty: Medium
* 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 numOfSubarrays(vector<int>& arr) {

}
};
```

Java Solution:

```
/**
* Problem: Number of Sub-arrays With Odd Sum
* Difficulty: Medium
* 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 numOfSubarrays(int[] arr) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Sub-arrays With Odd Sum
Difficulty: Medium
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 numOfSubarrays(self, arr: List[int]) -> int:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Number of Sub-arrays With Odd Sum
* Difficulty: Medium
* 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[]} arr
* @return {number}
*/
var numOfSubarrays = function(arr) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Sub-arrays With Odd Sum
* Difficulty: Medium
* 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 numOfSubarrays(arr: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Number of Sub-arrays With Odd Sum
* Difficulty: Medium
* 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 NumOfSubarrays(int[] arr) {

}
}
```

C Solution:

```
/*
* Problem: Number of Sub-arrays With Odd Sum
* Difficulty: Medium
* 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 numOfSubarrays(int* arr, int arrSize) {

}
```

Go Solution:

```
// Problem: Number of Sub-arrays With Odd Sum
// Difficulty: Medium
// 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 numOfSubarrays(arr []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun numOfSubarrays(arr: IntArray): Int {

}
}
```

Swift Solution:

```
class Solution {
func numOfSubarrays(_ arr: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Number of Sub-arrays With Odd Sum
// Difficulty: Medium
// 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 num_of_subarrays(arr: Vec<i32>) -> i32 {

}
```

```
}
```

Ruby Solution:

```
# @param {Integer[]} arr
# @return {Integer}
def num_of_subarrays(arr)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $arr
* @return Integer
*/
function numOfSubarrays($arr) {

}
}
```

Dart Solution:

```
class Solution {
int numOfSubarrays(List<int> arr) {

}
}
```

Scala Solution:

```
object Solution {
def numOfSubarrays(arr: Array[Int]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec num_of_subarrays(arr :: [integer]) :: integer
def num_of_subarrays(arr) do

end
end
```

Erlang Solution:

```
-spec num_of_subarrays(Arr :: [integer()]) -> integer().
num_of_subarrays(Arr) ->
.
```

Racket Solution:

```
(define/contract (num-of-subarrays arr)
(-> (listof exact-integer?) exact-integer?)
)
```