# Problem 3247: Number of Subsequences with Odd Sum

## Problem Information

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

## Problem Description

Given an array

nums

, return the number of

subsequences

with an odd sum of elements.

Since the answer may be very large, return it

modulo

10

9

+ 7

.

Example 1:

Input:

nums = [1,1,1]

Output:

4

Explanation:

The odd-sum subsequences are:

[

1

, 1, 1]

,

[1,

1

, 1],

[1, 1,

1

]

,

[

1, 1, 1

]

.

Example 2:

Input:

nums = [1,2,2]

Output:

4

Explanation:

The odd-sum subsequences are:

[

1

, 2, 2]

,

[

1, 2

, 2],

[

1

, 2,

2

]

,

[

1, 2, 2

]

.

Constraints:

1 <= nums.length <= 10

5

1 <= nums[i] <= 10

9

## Code Snippets

**C++:**

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

}
};
```

**Java:**

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

}
}
```

**Python3:**

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

**Python:**

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

**JavaScript:**

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

};
```

**TypeScript:**

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

};
```

**C#:**

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

}
}
```

**C:**

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

}
```

**Go:**

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

}
```

**Kotlin:**

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

}
}
```

**Swift:**

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

}
}
```

**Rust:**

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

}
}
```

**Ruby:**

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

end
```

**PHP:**

```
class Solution {

/**
```

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

}
}
```

**Dart:**

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

}
}
```

**Scala:**

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

}
}
```

**Elixir:**

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

end
end
```

**Erlang:**

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

**Racket:**

```
(define/contract (subsequence-count nums)
(-> (listof exact-integer?) exact-integer?)
)
```

## Solutions

### C++ Solution:

```
/*
* Problem: Number of Subsequences 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 subsequenceCount(vector<int>& nums) {

}
};
```

### Java Solution:

```
/**
* Problem: Number of Subsequences 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 subsequenceCount(int[] nums) {

}
```

```
}
```

### Python3 Solution:

```
"""
Problem: Number of Subsequences 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 subsequenceCount(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

### Python Solution:

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

### JavaScript Solution:

```
/**
* Problem: Number of Subsequences 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[]} nums
* @return {number}
*/
var subsequenceCount = function(nums) {

};
```

### TypeScript Solution:

```
/**
* Problem: Number of Subsequences 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 subsequenceCount(nums: number[]): number {

};
```

### C# Solution:

```
/*
* Problem: Number of Subsequences 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 SubsequenceCount(int[] nums) {

}
}
```

### C Solution:

```
/*
* Problem: Number of Subsequences 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 subsequenceCount(int* nums, int numsSize) {

}
```

### Go Solution:

```
// Problem: Number of Subsequences 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 subsequenceCount(nums []int) int {

}
```

### Kotlin Solution:

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

}
}
```

### Swift Solution:

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

```
}
}
```

### Rust Solution:

```
// Problem: Number of Subsequences 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 subsequence_count(nums: Vec<i32>) -> i32 {

}
}
```

### Ruby Solution:

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

end
```

### PHP Solution:

```
class Solution {

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

}
}
```

### Dart Solution:

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

}
}
```

### Scala Solution:

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

}
}
```

### Elixir Solution:

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

end
end
```

### Erlang Solution:

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

### Racket Solution:

```
(define/contract (subsequence-count nums)
(-> (listof exact-integer?) exact-integer?)
)
```