

Problem 3416: Subsequences with a Unique Middle Mode II

Problem Information

Difficulty: **Hard**

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

Given an integer array

nums

, find the number of

subsequences

of size 5 of

nums

with a

unique middle mode

.

Since the answer may be very large, return it

modulo

10

9

+ 7

.

A

mode

of a sequence of numbers is defined as the element that appears the

maximum

number of times in the sequence.

A sequence of numbers contains a

unique mode

if it has only one mode.

A sequence of numbers

seq

of size 5 contains a

unique middle mode

if the

middle element

(

seq[2]

) is a

unique mode

.

Example 1:

Input:

nums = [1,1,1,1,1,1]

Output:

6

Explanation:

[1, 1, 1, 1, 1]

is the only subsequence of size 5 that can be formed from this list, and it has a unique middle mode of 1.

Example 2:

Input:

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

Output:

4

Explanation:

[1, 2, 2, 3, 4]

and

[1, 2, 3, 3, 4]

have unique middle modes because the number at index 2 has the greatest frequency in the subsequence.

[1, 2, 2, 3, 3]

does not have a unique middle mode because 2 and 3 both appear twice in the subsequence.

Example 3:

Input:

nums = [0,1,2,3,4,5,6,7,8]

Output:

0

Explanation:

There does not exist a subsequence of length 5 with a unique middle mode.

Constraints:

$5 \leq \text{nums.length} \leq 10$

5

-10

9

$\text{nums}[i] \leq 10$

9

Code Snippets

C++:

```

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

    }
};

```

Java:

```

class Solution {
    public int subsequencesWithMiddleMode(int[] nums) {

    }
}

```

Python3:

```

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

```

Python:

```

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

```

JavaScript:

```

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

};

```

TypeScript:

```

function subsequencesWithMiddleMode(nums: number[]): number {

```

```
};
```

C#:

```
public class Solution {  
    public int SubsequencesWithMiddleMode(int[] nums) {  
  
    }  
}
```

C:

```
int subsequencesWithMiddleMode(int* nums, int numsSize) {  
  
}
```

Go:

```
func subsequencesWithMiddleMode(nums []int) int {  
  
}
```

Kotlin:

```
class Solution {  
    fun subsequencesWithMiddleMode(nums: IntArray): Int {  
  
    }  
}
```

Swift:

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

Rust:

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

```
}  
}
```

Ruby:

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

PHP:

```
class Solution {  
  
    /**  
     * @param Integer[] $nums  
     * @return Integer  
     */  
    function subsequencesWithMiddleMode($nums) {  
  
    }  
}
```

Dart:

```
class Solution {  
    int subsequencesWithMiddleMode(List<int> nums) {  
  
    }  
}
```

Scala:

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

Elixir:

```

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

  end

  end

```

Erlang:

```

-spec subsequences_with_middle_mode(Nums :: [integer()]) -> integer().
subsequences_with_middle_mode(Nums) ->
.

```

Racket:

```

(define/contract (subsequences-with-middle-mode nums)
  (-> (listof exact-integer?) exact-integer?)
  )

```

Solutions

C++ Solution:

```

/*
 * Problem: Subsequences with a Unique Middle Mode II
 * Difficulty: Hard
 * Tags: array, math, hash
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) for hash map
 */

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

    }

};

```

Java Solution:


```

/**
 * Problem: Subsequences with a Unique Middle Mode II
 * Difficulty: Hard
 * Tags: array, math, hash
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) for hash map
 */

class Solution {
public int subsequencesWithMiddleMode(int[] nums) {

}

}

```

Python3 Solution:

```

"""
Problem: Subsequences with a Unique Middle Mode II
Difficulty: Hard
Tags: array, math, hash

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

class Solution:
def subsequencesWithMiddleMode(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass

```

Python Solution:

```

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

```

JavaScript Solution:

```
/**
 * Problem: Subsequences with a Unique Middle Mode II
 * Difficulty: Hard
 * Tags: array, math, hash
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) for hash map
 */

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

};
```

TypeScript Solution:

```
/**
 * Problem: Subsequences with a Unique Middle Mode II
 * Difficulty: Hard
 * Tags: array, math, hash
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) for hash map
 */

function subsequencesWithMiddleMode(nums: number[]): number {

};
```

C# Solution:

```
/*
 * Problem: Subsequences with a Unique Middle Mode II
 * Difficulty: Hard
 * Tags: array, math, hash
 */
```

```

* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int SubsequencesWithMiddleMode(int[] nums) {

}

}

```

C Solution:

```

/*
* Problem: Subsequences with a Unique Middle Mode II
* Difficulty: Hard
* Tags: array, math, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

int subsequencesWithMiddleMode(int* nums, int numsSize) {

}

```

Go Solution:

```

// Problem: Subsequences with a Unique Middle Mode II
// Difficulty: Hard
// Tags: array, math, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func subsequencesWithMiddleMode(nums []int) int {

}

```

Kotlin Solution:

```
class Solution {  
    fun subsequencesWithMiddleMode(nums: IntArray): Int {  
  
    }  
}
```

Swift Solution:

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

Rust Solution:

```
// Problem: Subsequences with a Unique Middle Mode II  
// Difficulty: Hard  
// Tags: array, math, hash  
//  
// Approach: Use two pointers or sliding window technique  
// Time Complexity: O(n) or O(n log n)  
// Space Complexity: O(n) for hash map  
  
impl Solution {  
    pub fn subsequences_with_middle_mode(nums: Vec<i32>) -> i32 {  
  
    }  
}
```

Ruby Solution:

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

PHP Solution:

```

class Solution {

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

    }

}

```

Dart Solution:

```

class Solution {
  int subsequencesWithMiddleMode(List<int> nums) {

  }

}

```

Scala Solution:

```

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

  }

}

```

Elixir Solution:

```

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

  end

end

```

Erlang Solution:

```

-spec subsequences_with_middle_mode(Nums :: [integer()]) -> integer().
subsequences_with_middle_mode(Nums) ->
.

```

Racket Solution:

```
(define/contract (subsequences-with-middle-mode nums)
  (-> (listof exact-integer?) exact-integer?)
  )
```