Problem 1498: Number of Subsequences That Satisfy the Given Sum Condition

Problem Information

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

Problem Description

You are given an array of integers

nums

and an integer

target

.

Return

the number of

non-empty

subsequences of

nums

such that the sum of the minimum and maximum element on it is less or equal to

target

. Since the answer may be too large, return it

modulo

10

9

+ 7

.

Example 1:

Input:

nums = [3,5,6,7], target = 9

Output:

4

Explanation:

There are 4 subsequences that satisfy the condition. [3] -> Min value + max value <= target (3 + 3 <= 9) [3,5] -> (3 + 5 <= 9) [3,5,6] -> (3 + 6 <= 9) [3,6] -> (3 + 6 <= 9)

Example 2:

Input:

nums = [3,3,6,8], target = 10

Output:

6

Explanation:

There are 6 subsequences that satisfy the condition. (nums can have repeated numbers). [3] , [3] , [3,3], [3,6] , [3,6] , [3,3,6]

Example 3:

Input:

nums = [2,3,3,4,6,7], target = 12

Output:

61

Explanation:

There are 63 non-empty subsequences, two of them do not satisfy the condition ([6,7], [7]).
Number of valid subsequences (63 - 2 = 61).

Constraints:

1 <= nums.length <= 10

5

1 <= nums[i] <= 10

6

1 <= target <= 10

6

Code Snippets

C++:

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

}
```

```
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int numSubseq(int* nums, int numsSize, int target) {

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

```
def num_subseq(nums, target) do

end
end
```

Erlang:

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

Racket:

```
(define/contract (num-subseq nums target)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Subsequences That Satisfy the Given Sum Condition
* Difficulty: Medium
* Tags: array, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Number of Subsequences That Satisfy the Given Sum Condition
* Difficulty: Medium
* Tags: array, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
}
```

Python3 Solution:

```
"""
Problem: Number of Subsequences That Satisfy the Given Sum Condition
Difficulty: Medium
Tags: array, sort, search

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Number of Subsequences That Satisfy the Given Sum Condition
* Difficulty: Medium
* Tags: array, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Number of Subsequences That Satisfy the Given Sum Condition
* Difficulty: Medium
* Tags: array, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function numSubseq(nums: number[], target: number): number {

};
```

C# Solution:

```
/*
* Problem: Number of Subsequences That Satisfy the Given Sum Condition
* Difficulty: Medium
```

```
* Tags: array, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
}
```

C Solution:

```
/*
* Problem: Number of Subsequences That Satisfy the Given Sum Condition
* Difficulty: Medium
* Tags: array, sort, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

int numSubseq(int* nums, int numsSize, int target) {

}
```

Go Solution:

```
// Problem: Number of Subsequences That Satisfy the Given Sum Condition
// Difficulty: Medium
// Tags: array, sort, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

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

```
}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Number of Subsequences That Satisfy the Given Sum Condition
// Difficulty: Medium
// Tags: array, sort, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

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

}
}
```

Ruby Solution:

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

```
end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (num-subseq nums target)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```