Problem 2834: Find the Minimum Possible Sum of a Beautiful Array

Problem Information

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

Problem Description

You are given positive integers

n

and

target

.

An array

nums

is

beautiful

if it meets the following conditions:

nums.length == n

.

nums

consists of pairwise

distinct

positive

integers.

There doesn't exist two

distinct

indices,

i

and

j

, in the range

[0, n - 1]

, such that

nums[i] + nums[j] == target

.

Return

the

minimum

possible sum that a beautiful array could have modulo

10

9

+ 7

.

Example 1:

Input:

n = 2, target = 3

Output:

4

Explanation:

We can see that nums = [1,3] is beautiful. - The array nums has length n = 2. - The array nums consists of pairwise distinct positive integers. - There doesn't exist two distinct indices, i and j, with nums[i] + nums[j] == 3. It can be proven that 4 is the minimum possible sum that a beautiful array could have.

Example 2:

Input:

n = 3, target = 3

Output:

8

Explanation:

We can see that nums = [1,3,4] is beautiful. - The array nums has length n = 3. - The array nums consists of pairwise distinct positive integers. - There doesn't exist two distinct indices, i

and j, with nums[i] + nums[j] == 3. It can be proven that 8 is the minimum possible sum that a beautiful array could have.

Example 3:

Input:

n = 1, target = 1

Output:

1

Explanation:

We can see, that nums = [1] is beautiful.

Constraints:

1 <= n <= 10

9

1 <= target <= 10

9

Code Snippets

C++:

```
class Solution {
public:
int minimumPossibleSum(int n, int target) {

}
};
```

Java:

```
class Solution {
public int minimumPossibleSum(int n, int target) {

}
}
```

Python3:

```
class Solution:
def minimumPossibleSum(self, n: int, target: int) -> int:
```

Python:

```
class Solution(object):
def minimumPossibleSum(self, n, target):
"""
:type n: int
:type target: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} n
* @param {number} target
* @return {number}
*/
var minimumPossibleSum = function(n, target) {

};
```

TypeScript:

```
function minimumPossibleSum(n: number, target: number): number {

};
```

C#:

```
public class Solution {
public int MinimumPossibleSum(int n, int target) {
```

```
}
}
```

C:

```
int minimumPossibleSum(int n, int target) {

}
```

Go:

```
func minimumPossibleSum(n int, target int) int {

}
```

Kotlin:

```
class Solution {
fun minimumPossibleSum(n: Int, target: Int): Int {

}
}
```

Swift:

```
class Solution {
func minimumPossibleSum(_ n: Int, _ target: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn minimum_possible_sum(n: i32, target: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} n
# @param {Integer} target
# @return {Integer}
def minimum_possible_sum(n, target)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @param Integer $target
* @return Integer
*/
function minimumPossibleSum($n, $target) {

}
}
```

Dart:

```
class Solution {
int minimumPossibleSum(int n, int target) {

}
}
```

Scala:

```
object Solution {
def minimumPossibleSum(n: Int, target: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec minimum_possible_sum(n :: integer, target :: integer) :: integer
def minimum_possible_sum(n, target) do
```

```
end
end
```

Erlang:

```
-spec minimum_possible_sum(N :: integer(), Target :: integer()) -> integer().
minimum_possible_sum(N, Target) ->
.
```

Racket:

```
(define/contract (minimum-possible-sum n target)
(-> exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the Minimum Possible Sum of a Beautiful Array
* Difficulty: Medium
* Tags: array, greedy, math
*
* 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 minimumPossibleSum(int n, int target) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Minimum Possible Sum of a Beautiful Array
```

```
* Difficulty: Medium
* Tags: array, greedy, math
*
* 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 minimumPossibleSum(int n, int target) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Minimum Possible Sum of a Beautiful Array
Difficulty: Medium
Tags: array, greedy, math

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 minimumPossibleSum(self, n: int, target: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def minimumPossibleSum(self, n, target):
"""
:type n: int
:type target: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Find the Minimum Possible Sum of a Beautiful Array
* Difficulty: Medium
* Tags: array, greedy, math
*
* 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} n
* @param {number} target
* @return {number}
*/
var minimumPossibleSum = function(n, target) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Minimum Possible Sum of a Beautiful Array
* Difficulty: Medium
* Tags: array, greedy, math
*
* 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 minimumPossibleSum(n: number, target: number): number {

};
```

C# Solution:

```
/*
* Problem: Find the Minimum Possible Sum of a Beautiful Array
* Difficulty: Medium
* Tags: array, greedy, math
```

```
*
* 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 MinimumPossibleSum(int n, int target) {

}
}
```

C Solution:

```
/*
* Problem: Find the Minimum Possible Sum of a Beautiful Array
* Difficulty: Medium
* Tags: array, greedy, math
*
* 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 minimumPossibleSum(int n, int target) {

}
```

Go Solution:

```
// Problem: Find the Minimum Possible Sum of a Beautiful Array
// Difficulty: Medium
// Tags: array, greedy, math
//
// 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 minimumPossibleSum(n int, target int) int {

}
```

Kotlin Solution:

```
class Solution {
fun minimumPossibleSum(n: Int, target: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func minimumPossibleSum(_ n: Int, _ target: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Find the Minimum Possible Sum of a Beautiful Array
// Difficulty: Medium
// Tags: array, greedy, math
//
// 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 minimum_possible_sum(n: i32, target: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @param {Integer} target
# @return {Integer}
def minimum_possible_sum(n, target)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @param Integer $target
* @return Integer
*/
function minimumPossibleSum($n, $target) {

}
}
```

Dart Solution:

```
class Solution {
int minimumPossibleSum(int n, int target) {

}
}
```

Scala Solution:

```
object Solution {
def minimumPossibleSum(n: Int, target: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec minimum_possible_sum(n :: integer, target :: integer) :: integer
def minimum_possible_sum(n, target) do

end
end
```

Erlang Solution:

```
-spec minimum_possible_sum(N :: integer(), Target :: integer()) -> integer().
minimum_possible_sum(N, Target) ->
.
```

Racket Solution:

```
(define/contract (minimum-possible-sum n target)
(-> exact-integer? exact-integer? exact-integer?)
)
```