Problem 2365: Task Scheduler II

Problem Information

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

Problem Description

You are given a

0-indexed

array of positive integers

tasks

, representing tasks that need to be completed

in order

, where

tasks[i]

represents the

type

of the

i

th

task.

You are also given a positive integer

space

, which represents the

minimum

number of days that must pass

after

the completion of a task before another task of the

same

type can be performed.

Each day, until all tasks have been completed, you must either:

Complete the next task from

tasks

, or

Take a break.

Return

the

minimum

number of days needed to complete all tasks

.

Example 1:

Input:

tasks = [1,2,1,2,3,1], space = 3

Output:

9

Explanation:

One way to complete all tasks in 9 days is as follows: Day 1: Complete the 0th task. Day 2: Complete the 1st task. Day 3: Take a break. Day 4: Take a break. Day 5: Complete the 2nd task. Day 6: Complete the 3rd task. Day 7: Take a break. Day 8: Complete the 4th task. Day 9: Complete the 5th task. It can be shown that the tasks cannot be completed in less than 9 days.

Example 2:

Input:

tasks = [5,8,8,5], space = 2

Output:

6

Explanation:

One way to complete all tasks in 6 days is as follows: Day 1: Complete the 0th task. Day 2: Complete the 1st task. Day 3: Take a break. Day 4: Take a break. Day 5: Complete the 2nd task. Day 6: Complete the 3rd task. It can be shown that the tasks cannot be completed in less than 6 days.

Constraints:

1 <= tasks.length <= 10

5

1 <= tasks[i] <= 10

9

1 <= space <= tasks.length

Code Snippets

C++:

```
class Solution {
public:
long long taskSchedulerII(vector<int>& tasks, int space) {

}
};
```

Java:

```
class Solution {
public long taskSchedulerII(int[] tasks, int space) {

}
}
```

Python3:

```
class Solution:
def taskSchedulerII(self, tasks: List[int], space: int) -> int:
```

Python:

```
class Solution(object):
def taskSchedulerII(self, tasks, space):
"""
:type tasks: List[int]
:type space: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} tasks
* @param {number} space
* @return {number}
*/
var taskSchedulerII = function(tasks, space) {

};
```

TypeScript:

```
function taskSchedulerII(tasks: number[], space: number): number {

};
```

C#:

```
public class Solution {
public long TaskSchedulerII(int[] tasks, int space) {

}
}
```

C:

```
long long taskSchedulerII(int* tasks, int tasksSize, int space) {

}
```

Go:

```
func taskSchedulerII(tasks []int, space int) int64 {

}
```

Kotlin:

```
class Solution {
fun taskSchedulerII(tasks: IntArray, space: Int): Long {

}
}
```

Swift:

```
class Solution {
func taskSchedulerII(_ tasks: [Int], _ space: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn task_scheduler_ii(tasks: Vec<i32>, space: i32) -> i64 {

}
}
```

Ruby:

```
# @param {Integer[]} tasks
# @param {Integer} space
# @return {Integer}
def task_scheduler_ii(tasks, space)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $tasks
* @param Integer $space
* @return Integer
*/
function taskSchedulerII($tasks, $space) {

}
}
```

Dart:

```
class Solution {
int taskSchedulerII(List<int> tasks, int space) {

}
}
```

Scala:

```
object Solution {
def taskSchedulerII(tasks: Array[Int], space: Int): Long = {

}
}
```

Elixir:

```
defmodule Solution do
@spec task_scheduler_ii(tasks :: [integer], space :: integer) :: integer
def task_scheduler_ii(tasks, space) do

end
end
```

Erlang:

```
-spec task_scheduler_ii(Tasks :: [integer()], Space :: integer()) ->
integer().
task_scheduler_ii(Tasks, Space) ->
.
```

Racket:

```
(define/contract (task-scheduler-ii tasks space)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Task Scheduler II
* Difficulty: Medium
* Tags: array, 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:
long long taskSchedulerII(vector<int>& tasks, int space) {

}
};
```

Java Solution:

```
/**
* Problem: Task Scheduler II
* Difficulty: Medium
* Tags: array, 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 long taskSchedulerII(int[] tasks, int space) {

}
}
```

Python3 Solution:

```
"""
Problem: Task Scheduler II
Difficulty: Medium
Tags: array, 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 taskSchedulerII(self, tasks: List[int], space: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def taskSchedulerII(self, tasks, space):
"""
:type tasks: List[int]
:type space: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Task Scheduler II
* Difficulty: Medium
* Tags: array, 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[]} tasks
* @param {number} space
* @return {number}
*/
var taskSchedulerII = function(tasks, space) {

};
```

TypeScript Solution:

```
/**
* Problem: Task Scheduler II
* Difficulty: Medium
* Tags: array, 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 taskSchedulerII(tasks: number[], space: number): number {

};
```

C# Solution:

```
/*
* Problem: Task Scheduler II
* Difficulty: Medium
* Tags: array, 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 long TaskSchedulerII(int[] tasks, int space) {

}
}
```

C Solution:

```
/*
* Problem: Task Scheduler II
* Difficulty: Medium
* Tags: array, 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
*/

long long taskSchedulerII(int* tasks, int tasksSize, int space) {

}
```

Go Solution:

```go
// Problem: Task Scheduler II
// Difficulty: Medium
// Tags: array, 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 taskSchedulerII(tasks []int, space int) int64 {

}
```

Kotlin Solution:

```kotlin
class Solution {
fun taskSchedulerII(tasks: IntArray, space: Int): Long {

}
}
```

Swift Solution:

```swift
class Solution {
func taskSchedulerII(_ tasks: [Int], _ space: Int) -> Int {

}
}
```

Rust Solution:

```rust
// Problem: Task Scheduler II
// Difficulty: Medium
// Tags: array, 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 task_scheduler_ii(tasks: Vec<i32>, space: i32) -> i64 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} tasks
# @param {Integer} space
# @return {Integer}
def task_scheduler_ii(tasks, space)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $tasks
* @param Integer $space
* @return Integer
*/
function taskSchedulerII($tasks, $space) {

}
}
```

Dart Solution:

```
class Solution {
int taskSchedulerII(List<int> tasks, int space) {

}
}
```

Scala Solution:

```
object Solution {
def taskSchedulerII(tasks: Array[Int], space: Int): Long = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec task_scheduler_ii(tasks :: [integer], space :: integer) :: integer
def task_scheduler_ii(tasks, space) do

end
end
```

Erlang Solution:

```
-spec task_scheduler_ii(Tasks :: [integer()], Space :: integer()) ->
integer().
task_scheduler_ii(Tasks, Space) ->
.
```

Racket Solution:

```
(define/contract (task-scheduler-ii tasks space)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```