Problem 1011: Capacity To Ship Packages Within D Days

Problem Information

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

Problem Description

A conveyor belt has packages that must be shipped from one port to another within

days

days.

The

i

th

package on the conveyor belt has a weight of

weights[i]

. Each day, we load the ship with packages on the conveyor belt (in the order given by

weights

). We may not load more weight than the maximum weight capacity of the ship.

Return the least weight capacity of the ship that will result in all the packages on the conveyor belt being shipped within days

days.

Example 1:

Input:

weights = [1,2,3,4,5,6,7,8,9,10], days = 5

Output:

15

Explanation:

A ship capacity of 15 is the minimum to ship all the packages in 5 days like this: 1st day: 1, 2, 3, 4, 5 2nd day: 6, 7 3rd day: 8 4th day: 9 5th day: 10

Note that the cargo must be shipped in the order given, so using a ship of capacity 14 and splitting the packages into parts like (2, 3, 4, 5), (1, 6, 7), (8), (9), (10) is not allowed.

Example 2:

Input:

weights = [3,2,2,4,1,4], days = 3

Output:

6

Explanation:

A ship capacity of 6 is the minimum to ship all the packages in 3 days like this: 1st day: 3, 2 2nd day: 2, 4 3rd day: 1, 4

Example 3:

Input:

weights = [1,2,3,1,1], days = 4

Output:

3

Explanation:

1st day: 1 2nd day: 2 3rd day: 3 4th day: 1, 1

Constraints:

1 <= days <= weights.length <= 5 * 10

4

1 <= weights[i] <= 500

Code Snippets

C++:

```
class Solution {
public:
int shipWithinDays(vector<int>& weights, int days) {

}
};
```

Java:

```
class Solution {
public int shipWithinDays(int[] weights, int days) {

}
}
```

Python3:

```
class Solution:
def shipWithinDays(self, weights: List[int], days: int) -> int:
```

Python:

```
class Solution(object):
def shipWithinDays(self, weights, days):
"""
:type weights: List[int]
:type days: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} weights
* @param {number} days
* @return {number}
*/
var shipWithinDays = function(weights, days) {

};
```

TypeScript:

```
function shipWithinDays(weights: number[], days: number): number {

};
```

C#:

```
public class Solution {
public int ShipWithinDays(int[] weights, int days) {

}
}
```

C:

```
int shipWithinDays(int* weights, int weightsSize, int days) {

}
```

Go:

```
func shipWithinDays(weights []int, days int) int {

}
```

Kotlin:

```
class Solution {
fun shipWithinDays(weights: IntArray, days: Int): Int {

}
}
```

Swift:

```
class Solution {
func shipWithinDays(_ weights: [Int], _ days: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn ship_within_days(weights: Vec<i32>, days: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} weights
# @param {Integer} days
# @return {Integer}
def ship_within_days(weights, days)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $weights
* @param Integer $days
* @return Integer
*/
function shipWithinDays($weights, $days) {

}
}
```

Dart:

```
class Solution {
int shipWithinDays(List<int> weights, int days) {

}
}
```

Scala:

```
object Solution {
def shipWithinDays(weights: Array[Int], days: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec ship_within_days(weights :: [integer], days :: integer) :: integer
def ship_within_days(weights, days) do

end
end
```

Erlang:

```
-spec ship_within_days(Weights :: [integer()], Days :: integer()) ->
integer().
```

```
ship_within_days(Weights, Days) ->
.
```

Racket:

```
(define/contract (ship-within-days weights days)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Capacity To Ship Packages Within D Days
* Difficulty: Medium
* Tags: array, 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 shipWithinDays(vector<int>& weights, int days) {

}
};
```

Java Solution:

```
/**
* Problem: Capacity To Ship Packages Within D Days
* Difficulty: Medium
* Tags: array, 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 shipWithinDays(int[] weights, int days) {

}
}
```

Python3 Solution:

```
"""
Problem: Capacity To Ship Packages Within D Days
Difficulty: Medium
Tags: array, 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 shipWithinDays(self, weights: List[int], days: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def shipWithinDays(self, weights, days):
"""
:type weights: List[int]
:type days: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Capacity To Ship Packages Within D Days
* Difficulty: Medium
* Tags: array, 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[]} weights
* @param {number} days
* @return {number}
*/
var shipWithinDays = function(weights, days) {

};
```

TypeScript Solution:

```
/**
* Problem: Capacity To Ship Packages Within D Days
* Difficulty: Medium
* Tags: array, 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 shipWithinDays(weights: number[], days: number): number {

};
```

C# Solution:

```
/*
* Problem: Capacity To Ship Packages Within D Days
* Difficulty: Medium
* Tags: array, 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 ShipWithinDays(int[] weights, int days) {

}
}
```

C Solution:

```
/*
* Problem: Capacity To Ship Packages Within D Days
* Difficulty: Medium
* Tags: array, 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 shipWithinDays(int* weights, int weightsSize, int days) {

}
```

Go Solution:

```
// Problem: Capacity To Ship Packages Within D Days
// Difficulty: Medium
// Tags: array, 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 shipWithinDays(weights []int, days int) int {

}
```

Kotlin Solution:

```
class Solution {
fun shipWithinDays(weights: IntArray, days: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func shipWithinDays(_ weights: [Int], _ days: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Capacity To Ship Packages Within D Days
// Difficulty: Medium
// Tags: array, 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 ship_within_days(weights: Vec<i32>, days: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} weights
# @param {Integer} days
# @return {Integer}
def ship_within_days(weights, days)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $weights
* @param Integer $days
* @return Integer
*/
function shipWithinDays($weights, $days) {

}
}
```

Dart Solution:

```
class Solution {
int shipWithinDays(List<int> weights, int days) {

}
}
```

Scala Solution:

```
object Solution {
def shipWithinDays(weights: Array[Int], days: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec ship_within_days(weights :: [integer], days :: integer) :: integer
def ship_within_days(weights, days) do

end
end
```

Erlang Solution:

```
-spec ship_within_days(Weights :: [integer()], Days :: integer()) ->
integer().
ship_within_days(Weights, Days) ->
```

```
.
```

Racket Solution:

```
(define/contract (ship-within-days weights days)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```