Problem 1196: How Many Apples Can You Put into the Basket

Problem Information

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

Problem Description

You have some apples and a basket that can carry up to

5000

units of weight.

Given an integer array

weight

where

weight[i]

is the weight of the

i

th

apple, return

the maximum number of apples you can put in the basket

.

Example 1:

Input:

weight = [100,200,150,1000]

Output:

4

Explanation:

All 4 apples can be carried by the basket since their sum of weights is 1450.

Example 2:

Input:

weight = [900,950,800,1000,700,800]

Output:

5

Explanation:

The sum of weights of the 6 apples exceeds 5000 so we choose any 5 of them.

Constraints:

1 <= weight.length <= 10

3

1 <= weight[i] <= 10

3

Code Snippets

C++:

```
class Solution {
public:
int maxNumberOfApples(vector<int>& weight) {

}
};
```

Java:

```
class Solution {
public int maxNumberOfApples(int[] weight) {

}
}
```

Python3:

```
class Solution:
def maxNumberOfApples(self, weight: List[int]) -> int:
```

Python:

```
class Solution(object):
def maxNumberOfApples(self, weight):
"""
:type weight: List[int]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} weight
* @return {number}
*/
var maxNumberOfApples = function(weight) {

};
```

TypeScript:

```
function maxNumberOfApples(weight: number[]): number {

};
```

C#:

```
public class Solution {
public int MaxNumberOfApples(int[] weight) {

}
}
```

C:

```
int maxNumberOfApples(int* weight, int weightSize) {

}
```

Go:

```
func maxNumberOfApples(weight []int) int {

}
```

Kotlin:

```
class Solution {
fun maxNumberOfApples(weight: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func maxNumberOfApples(_ weight: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn max_number_of_apples(weight: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} weight
# @return {Integer}
def max_number_of_apples(weight)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $weight
* @return Integer
*/
function maxNumberOfApples($weight) {

}
}
```

Dart:

```
class Solution {
int maxNumberOfApples(List<int> weight) {

}
}
```

Scala:

```
object Solution {
def maxNumberOfApples(weight: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec max_number_of_apples(weight :: [integer]) :: integer
def max_number_of_apples(weight) do

end
end
```

Erlang:

```
-spec max_number_of_apples(Weight :: [integer()]) -> integer().
max_number_of_apples(Weight) ->
.
```

Racket:

```
(define/contract (max-number-of-apples weight)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: How Many Apples Can You Put into the Basket
* Difficulty: Easy
* Tags: array, greedy, sort
*
* 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 maxNumberOfApples(vector<int>& weight) {
}
};
```

Java Solution:

```
/**
* Problem: How Many Apples Can You Put into the Basket
* Difficulty: Easy
* Tags: array, greedy, sort
*
* 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 maxNumberOfApples(int[] weight) {

}
}
```

Python3 Solution:

```
"""
Problem: How Many Apples Can You Put into the Basket
Difficulty: Easy
Tags: array, greedy, sort

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 maxNumberOfApples(self, weight: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def maxNumberOfApples(self, weight):
"""
:type weight: List[int]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: How Many Apples Can You Put into the Basket
* Difficulty: Easy
* Tags: array, greedy, sort
*
* 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[]} weight
* @return {number}
*/
var maxNumberOfApples = function(weight) {

};
```

TypeScript Solution:

```
/**
* Problem: How Many Apples Can You Put into the Basket
* Difficulty: Easy
* Tags: array, greedy, sort
*
* 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 maxNumberOfApples(weight: number[]): number {

};
```

C# Solution:

```
/*
* Problem: How Many Apples Can You Put into the Basket
* Difficulty: Easy
* Tags: array, greedy, sort
*
* 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 MaxNumberOfApples(int[] weight) {

}
}
```

C Solution:

```
/*
* Problem: How Many Apples Can You Put into the Basket
* Difficulty: Easy
* Tags: array, greedy, sort
*
* 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 maxNumberOfApples(int* weight, int weightSize) {

}
```

Go Solution:

```
// Problem: How Many Apples Can You Put into the Basket
// Difficulty: Easy
// Tags: array, greedy, sort
//
// 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 maxNumberOfApples(weight []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun maxNumberOfApples(weight: IntArray): Int {

}
}
```

Swift Solution:

```
class Solution {
func maxNumberOfApples(_ weight: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: How Many Apples Can You Put into the Basket
// Difficulty: Easy
// Tags: array, greedy, sort
//
// 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 max_number_of_apples(weight: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} weight
# @return {Integer}
def max_number_of_apples(weight)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $weight
* @return Integer
*/
function maxNumberOfApples($weight) {

}
}
```

Dart Solution:

```
class Solution {
int maxNumberOfApples(List<int> weight) {

}
}
```

Scala Solution:

```
object Solution {
def maxNumberOfApples(weight: Array[Int]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec max_number_of_apples(weight :: [integer]) :: integer
def max_number_of_apples(weight) do

end
```

```
end
```

Erlang Solution:

```
-spec max_number_of_apples(Weight :: [integer()]) -> integer().
max_number_of_apples(Weight) ->
.
```

Racket Solution:

```
(define/contract (max-number-of-apples weight)
(-> (listof exact-integer?) exact-integer?)
)
```