Problem 2305: Fair Distribution of Cookies

Problem Information

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

Problem Description

You are given an integer array

cookies

, where

cookies[i]

denotes the number of cookies in the

i

th

bag. You are also given an integer

k

that denotes the number of children to distribute

all

the bags of cookies to. All the cookies in the same bag must go to the same child and cannot be split up.

The

unfairness

of a distribution is defined as the

maximum

total

cookies obtained by a single child in the distribution.

Return

the

minimum

unfairness of all distributions

.

Example 1:

Input:

cookies = [8,15,10,20,8], k = 2

Output:

31

Explanation:

One optimal distribution is [8,15,8] and [10,20] - The 1

st

child receives [8,15,8] which has a total of 8 + 15 + 8 = 31 cookies. - The 2

nd

child receives [10,20] which has a total of 10 + 20 = 30 cookies. The unfairness of the distribution is max(31,30) = 31. It can be shown that there is no distribution with an unfairness less than 31.

Example 2:

Input:

cookies = [6,1,3,2,2,4,1,2], k = 3

Output:

7

Explanation:

One optimal distribution is [6,1], [3,2,2], and [4,1,2] - The 1

st

child receives [6,1] which has a total of 6 + 1 = 7 cookies. - The 2

nd

child receives [3,2,2] which has a total of 3 + 2 + 2 = 7 cookies. - The 3

rd

child receives [4,1,2] which has a total of 4 + 1 + 2 = 7 cookies. The unfairness of the distribution is max(7,7,7) = 7. It can be shown that there is no distribution with an unfairness less than 7.

Constraints:

2 <= cookies.length <= 8

1 <= cookies[i] <= 10

5

2 <= k <= cookies.length

Code Snippets

C++:

```
class Solution {
public:
int distributeCookies(vector<int>& cookies, int k) {

}
};
```

Java:

```
class Solution {
public int distributeCookies(int[] cookies, int k) {

}
}
```

Python3:

```
class Solution:
def distributeCookies(self, cookies: List[int], k: int) -> int:
```

Python:

```
class Solution(object):
def distributeCookies(self, cookies, k):
"""
:type cookies: List[int]
:type k: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} cookies
* @param {number} k
* @return {number}
*/
var distributeCookies = function(cookies, k) {

};
```

TypeScript:

```
function distributeCookies(cookies: number[], k: number): number {

};
```

C#:

```
public class Solution {
public int DistributeCookies(int[] cookies, int k) {

}
}
```

C:

```
int distributeCookies(int* cookies, int cookiesSize, int k) {

}
```

Go:

```
func distributeCookies(cookies []int, k int) int {

}
```

Kotlin:

```
class Solution {
fun distributeCookies(cookies: IntArray, k: Int): Int {

}
}
```

Swift:

```
class Solution {
func distributeCookies(_ cookies: [Int], _ k: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn distribute_cookies(cookies: Vec<i32>, k: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} cookies
# @param {Integer} k
# @return {Integer}
def distribute_cookies(cookies, k)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $cookies
* @param Integer $k
* @return Integer
*/
function distributeCookies($cookies, $k) {

}
}
```

Dart:

```
class Solution {
int distributeCookies(List<int> cookies, int k) {
```

```
}
}
```

Scala:

```
object Solution {
def distributeCookies(cookies: Array[Int], k: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec distribute_cookies(cookies :: [integer], k :: integer) :: integer
def distribute_cookies(cookies, k) do

end
end
```

Erlang:

```
-spec distribute_cookies(Cookies :: [integer()], K :: integer()) ->
integer().
distribute_cookies(Cookies, K) ->
.
```

Racket:

```
(define/contract (distribute-cookies cookies k)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Fair Distribution of Cookies
```

```
* Difficulty: Medium
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
int distributeCookies(vector<int>& cookies, int k) {

}
};
```

Java Solution:

```
/**
* Problem: Fair Distribution of Cookies
* Difficulty: Medium
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int distributeCookies(int[] cookies, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: Fair Distribution of Cookies
Difficulty: Medium
Tags: array, dp

Approach: Use two pointers or sliding window technique
```

```
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

class Solution:
def distributeCookies(self, cookies: List[int], k: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def distributeCookies(self, cookies, k):
"""
:type cookies: List[int]
:type k: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Fair Distribution of Cookies
* Difficulty: Medium
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {number[]} cookies
* @param {number} k
* @return {number}
*/
var distributeCookies = function(cookies, k) {

};
```

TypeScript Solution:

```
/**
* Problem: Fair Distribution of Cookies
* Difficulty: Medium
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function distributeCookies(cookies: number[], k: number): number {

};
```

C# Solution:

```
/*
* Problem: Fair Distribution of Cookies
* Difficulty: Medium
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public int DistributeCookies(int[] cookies, int k) {

}
}
```

C Solution:

```
/*
* Problem: Fair Distribution of Cookies
* Difficulty: Medium
* Tags: array, dp
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
```

```
*/

int distributeCookies(int* cookies, int cookiesSize, int k) {

}
```

Go Solution:

```
// Problem: Fair Distribution of Cookies
// Difficulty: Medium
// Tags: array, dp
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func distributeCookies(cookies []int, k int) int {

}
```

Kotlin Solution:

```
class Solution {
fun distributeCookies(cookies: IntArray, k: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func distributeCookies(_ cookies: [Int], _ k: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Fair Distribution of Cookies
// Difficulty: Medium
// Tags: array, dp
```

```
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn distribute_cookies(cookies: Vec<i32>, k: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} cookies
# @param {Integer} k
# @return {Integer}
def distribute_cookies(cookies, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $cookies
* @param Integer $k
* @return Integer
*/
function distributeCookies($cookies, $k) {

}
}
```

Dart Solution:

```
class Solution {
int distributeCookies(List<int> cookies, int k) {

}
}
```

Scala Solution:

```
object Solution {
def distributeCookies(cookies: Array[Int], k: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec distribute_cookies(cookies :: [integer], k :: integer) :: integer
def distribute_cookies(cookies, k) do

end
end
```

Erlang Solution:

```
-spec distribute_cookies(Cookies :: [integer()], K :: integer()) ->
integer().
distribute_cookies(Cookies, K) ->
.
```

Racket Solution:

```
(define/contract (distribute-cookies cookies k)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```