Problem 2171: Removing Minimum Number of Magic Beans

Problem Information

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

Problem Description

You are given an array of

positive

integers

beans

, where each integer represents the number of magic beans found in a particular magic bag.

Remove

any number of beans (

possibly none

) from each bag such that the number of beans in each remaining

non-empty

bag (still containing

at least one

bean) is

equal

. Once a bean has been removed from a bag, you are

not

allowed to return it to any of the bags.

Return

the

minimum

number of magic beans that you have to remove

.

Example 1:

Input:

beans = [4,1,6,5]

Output:

4

Explanation:

- We remove 1 bean from the bag with only 1 bean. This results in the remaining bags: [4,

0

,6,5] - Then we remove 2 beans from the bag with 6 beans. This results in the remaining bags: [4,0,

4

,5] - Then we remove 1 bean from the bag with 5 beans. This results in the remaining bags: [4,0,4,

4

] We removed a total of 1 + 2 + 1 = 4 beans to make the remaining non-empty bags have an equal number of beans. There are no other solutions that remove 4 beans or fewer.

Example 2:

Input:

beans = [2,10,3,2]

Output:

7

Explanation:

- We remove 2 beans from one of the bags with 2 beans. This results in the remaining bags: [

0

,10,3,2] - Then we remove 2 beans from the other bag with 2 beans. This results in the remaining bags: [0,10,3,

0

] - Then we remove 3 beans from the bag with 3 beans. This results in the remaining bags: [0,10,

0

,0] We removed a total of 2 + 2 + 3 = 7 beans to make the remaining non-empty bags have an equal number of beans. There are no other solutions that removes 7 beans or fewer.

Constraints:

1 <= beans.length <= 10

5

1 <= beans[i] <= 10

5

Code Snippets

C++:

```
class Solution {
public:
long long minimumRemoval(vector<int>& beans) {

}
};
```

Java:

```
class Solution {
public long minimumRemoval(int[] beans) {

}
}
```

Python3:

```
class Solution:
def minimumRemoval(self, beans: List[int]) -> int:
```

Python:

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

JavaScript:

```javascript
/**
* @param {number[]} beans
* @return {number}
*/
var minimumRemoval = function(beans) {

};
```

TypeScript:

```typescript
function minimumRemoval(beans: number[]): number {

};
```

C#:

```csharp
public class Solution {
public long MinimumRemoval(int[] beans) {

}
}
```

C:

```c
long long minimumRemoval(int* beans, int beansSize) {

}
```

Go:

```go
func minimumRemoval(beans []int) int64 {

}
```

Kotlin:

```kotlin
class Solution {
fun minimumRemoval(beans: IntArray): Long {

}
}
```

Swift:

```
class Solution {
func minimumRemoval(_ beans: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn minimum_removal(beans: Vec<i32>) -> i64 {

}
}
```

Ruby:

```
# @param {Integer[]} beans
# @return {Integer}
def minimum_removal(beans)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $beans
* @return Integer
*/
function minimumRemoval($beans) {

}
}
```

Dart:

```
class Solution {
int minimumRemoval(List<int> beans) {

}

}
```

Scala:

```
object Solution {
def minimumRemoval(beans: Array[Int]): Long = {

}
}
```

Elixir:

```
defmodule Solution do
@spec minimum_removal(beans :: [integer]) :: integer
def minimum_removal(beans) do

end
end
```

Erlang:

```
-spec minimum_removal(Beans :: [integer()]) -> integer().
minimum_removal(Beans) ->
.
```

Racket:

```
(define/contract (minimum-removal beans)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Removing Minimum Number of Magic Beans
* Difficulty: Medium
* 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:
long long minimumRemoval(vector<int>& beans) {

}
};
```

Java Solution:

```
/**
* Problem: Removing Minimum Number of Magic Beans
* Difficulty: Medium
* 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 long minimumRemoval(int[] beans) {

}
}
```

Python3 Solution:

```
"""
Problem: Removing Minimum Number of Magic Beans
Difficulty: Medium
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 minimumRemoval(self, beans: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Removing Minimum Number of Magic Beans
* Difficulty: Medium
* 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[]} beans
* @return {number}
*/
var minimumRemoval = function(beans) {

};
```

TypeScript Solution:

```
/**
* Problem: Removing Minimum Number of Magic Beans
* Difficulty: Medium
* 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 minimumRemoval(beans: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Removing Minimum Number of Magic Beans
* Difficulty: Medium
* 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 long MinimumRemoval(int[] beans) {

}
}
```

C Solution:

```
/*
* Problem: Removing Minimum Number of Magic Beans
* Difficulty: Medium
* 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
*/

long long minimumRemoval(int* beans, int beansSize) {
}
```

Go Solution:

```
// Problem: Removing Minimum Number of Magic Beans
// Difficulty: Medium
// 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 minimumRemoval(beans []int) int64 {

}
```

Kotlin Solution:

```
class Solution {
fun minimumRemoval(beans: IntArray): Long {

}
}
```

Swift Solution:

```
class Solution {
func minimumRemoval(_ beans: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Removing Minimum Number of Magic Beans
// Difficulty: Medium
// 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 minimum_removal(beans: Vec<i32>) -> i64 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} beans
# @return {Integer}
def minimum_removal(beans)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $beans
* @return Integer
*/
function minimumRemoval($beans) {

}
}
```

Dart Solution:

```
class Solution {
int minimumRemoval(List<int> beans) {

}
}
```

Scala Solution:

```
object Solution {
def minimumRemoval(beans: Array[Int]): Long = {


}
}
```

Elixir Solution:

```
defmodule Solution do
@spec minimum_removal(beans :: [integer]) :: integer
def minimum_removal(beans) do

end
end
```

Erlang Solution:

```
-spec minimum_removal(Beans :: [integer()]) -> integer().
minimum_removal(Beans) ->
.
```

Racket Solution:

```
(define/contract (minimum-removal beans)
(-> (listof exact-integer?) exact-integer?)
)
```