Problem 2178: Maximum Split of Positive Even Integers

Problem Information

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

Problem Description

You are given an integer

finalSum

. Split it into a sum of a

maximum

number of

unique

positive even integers.

For example, given

finalSum = 12

, the following splits are

valid

(unique positive even integers summing up to

finalSum

):

(12)

,

(2 + 10)

,

(2 + 4 + 6)

, and

(4 + 8)

. Among them,

(2 + 4 + 6)

contains the maximum number of integers. Note that

finalSum

cannot be split into

(2 + 2 + 4 + 4)

as all the numbers should be unique.

Return

a list of integers that represent a valid split containing a

maximum

number of integers

. If no valid split exists for

finalSum

, return

an

empty

list

. You may return the integers in

any

order.

Example 1:

Input:

finalSum = 12

Output:

[2,4,6]

Explanation:

The following are valid splits:

(12)

,

(2 + 10)

,

(2 + 4 + 6)

, and

(4 + 8)

. (2 + 4 + 6) has the maximum number of integers, which is 3. Thus, we return [2,4,6]. Note that [2,6,4], [6,2,4], etc. are also accepted.

Example 2:

Input:

finalSum = 7

Output:

[]

Explanation:

There are no valid splits for the given finalSum. Thus, we return an empty array.

Example 3:

Input:

finalSum = 28

Output:

[6,8,2,12]

Explanation:

The following are valid splits:

(2 + 26)

,

(6 + 8 + 2 + 12)

, and

(4 + 24)

.

(6 + 8 + 2 + 12)

has the maximum number of integers, which is 4. Thus, we return [6,8,2,12]. Note that [10,2,4,12], [6,2,4,16], etc. are also accepted.

Constraints:

1 <= finalSum <= 10

10

Code Snippets

C++:

```
class Solution {
public:
vector<long long> maximumEvenSplit(long long finalSum) {

}
};
```

Java:

```
class Solution {
public List<Long> maximumEvenSplit(long finalSum) {

}
}
```

Python3:

```
class Solution:
def maximumEvenSplit(self, finalSum: int) -> List[int]:
```

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function maximumEvenSplit(finalSum: number): number[] {

};
```

C#:

```
public class Solution {
public IList<long> MaximumEvenSplit(long finalSum) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
```

```
long long* maximumEvenSplit(long long finalSum, int* returnSize) {

}
```

Go:

```
func maximumEvenSplit(finalSum int64) []int64 {

}
```

Kotlin:

```
class Solution {
fun maximumEvenSplit(finalSum: Long): List<Long> {

}
}
```

Swift:

```
class Solution {
func maximumEvenSplit(_ finalSum: Int) -> [Int] {

}
}
```

Rust:

```
impl Solution {
pub fn maximum_even_split(final_sum: i64) -> Vec<i64> {

}
}
```

Ruby:

```
# @param {Integer} final_sum
# @return {Integer[]}
def maximum_even_split(final_sum)

end
```

PHP:

```
class Solution {

/**
* @param Integer $finalSum
* @return Integer[]
*/
function maximumEvenSplit($finalSum) {

}
}
```

Dart:

```
class Solution {
List<int> maximumEvenSplit(int finalSum) {

}
}
```

Scala:

```
object Solution {
def maximumEvenSplit(finalSum: Long): List[Long] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec maximum_even_split(final_sum :: integer) :: [integer]
def maximum_even_split(final_sum) do

end
end
```

Erlang:

```
-spec maximum_even_split(FinalSum :: integer()) -> [integer()].
maximum_even_split(FinalSum) ->
.
```

Racket:

```
(define/contract (maximum-even-split finalSum)
(-> exact-integer? (listof exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Split of Positive Even Integers
* Difficulty: Medium
* Tags: array, greedy, math
*
* 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:
vector<long long> maximumEvenSplit(long long finalSum) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Split of Positive Even Integers
* Difficulty: Medium
* Tags: array, greedy, math
*
* 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 List<Long> maximumEvenSplit(long finalSum) {
}
}
```

Python3 Solution:

```
"""
Problem: Maximum Split of Positive Even Integers
Difficulty: Medium
Tags: array, greedy, math

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Maximum Split of Positive Even Integers
* Difficulty: Medium
* Tags: array, greedy, math
*
* 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} finalSum
* @return {number[]}
*/
var maximumEvenSplit = function(finalSum) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Split of Positive Even Integers
* Difficulty: Medium
* Tags: array, greedy, math
*
* 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 maximumEvenSplit(finalSum: number): number[] {

};
```

C# Solution:

```
/*
* Problem: Maximum Split of Positive Even Integers
* Difficulty: Medium
* Tags: array, greedy, math
*
* 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 IList<long> MaximumEvenSplit(long finalSum) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Split of Positive Even Integers
* Difficulty: Medium
* Tags: array, greedy, math
*
* 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
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
long long* maximumEvenSplit(long long finalSum, int* returnSize) {

}
```

Go Solution:

```
// Problem: Maximum Split of Positive Even Integers
// Difficulty: Medium
// Tags: array, greedy, math
//
// 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 maximumEvenSplit(finalSum int64) []int64 {

}
```

Kotlin Solution:

```
class Solution {
fun maximumEvenSplit(finalSum: Long): List<Long> {

}
}
```

Swift Solution:

```
class Solution {
func maximumEvenSplit(_ finalSum: Int) -> [Int] {

}
}
```

Rust Solution:

```
// Problem: Maximum Split of Positive Even Integers
// Difficulty: Medium
// Tags: array, greedy, math
//
// 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 maximum_even_split(final_sum: i64) -> Vec<i64> {

}
}
```

Ruby Solution:

```
# @param {Integer} final_sum
# @return {Integer[]}
def maximum_even_split(final_sum)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $finalSum
* @return Integer[]
*/
function maximumEvenSplit($finalSum) {

}
}
```

Dart Solution:

```
class Solution {
List<int> maximumEvenSplit(int finalSum) {

}
}
```

Scala Solution:

```
object Solution {
def maximumEvenSplit(finalSum: Long): List[Long] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec maximum_even_split(final_sum :: integer) :: [integer]
def maximum_even_split(final_sum) do

end
end
```

Erlang Solution:

```
-spec maximum_even_split(FinalSum :: integer()) -> [integer()].
maximum_even_split(FinalSum) ->
.
```

Racket Solution:

```
(define/contract (maximum-even-split finalSum)
(-> exact-integer? (listof exact-integer?))
)
```