Problem 956: Tallest Billboard

Problem Information

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

Problem Description

You are installing a billboard and want it to have the largest height. The billboard will have two steel supports, one on each side. Each steel support must be an equal height.

You are given a collection of

rods

that can be welded together. For example, if you have rods of lengths

1

,

2

, and

3

, you can weld them together to make a support of length

6

.

Return

the largest possible height of your billboard installation

. If you cannot support the billboard, return

0

.

Example 1:

Input:

rods = [1,2,3,6]

Output:

6

Explanation:

We have two disjoint subsets {1,2,3} and {6}, which have the same sum = 6.

Example 2:

Input:

rods = [1,2,3,4,5,6]

Output:

10

Explanation:

We have two disjoint subsets {2,3,5} and {4,6}, which have the same sum = 10.

Example 3:

Input:

rods = [1,2]

Output:

0

Explanation:

The billboard cannot be supported, so we return 0.

Constraints:

1 <= rods.length <= 20

1 <= rods[i] <= 1000

sum(rods[i]) <= 5000

Code Snippets

C++:

```
class Solution {
public:
int tallestBillboard(vector<int>& rods) {

}
};
```

Java:

```
class Solution {
public int tallestBillboard(int[] rods) {

}
}
```

Python3:

```
class Solution:
def tallestBillboard(self, rods: List[int]) -> int:
```

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function tallestBillboard(rods: number[]): number {

};
```

C#:

```
public class Solution {
public int TallestBillboard(int[] rods) {

}
}
```

C:

```
int tallestBillboard(int* rods, int rodsSize) {

}
```

Go:

```
func tallestBillboard(rods []int) int {

}
```

Kotlin:

```
class Solution {
fun tallestBillboard(rods: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func tallestBillboard(_ rods: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn tallest_billboard(rods: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} rods
# @return {Integer}
def tallest_billboard(rods)

end
```

PHP:

```
class Solution {

/**
```

```
* @param Integer[] $rods
* @return Integer
*/
function tallestBillboard($rods) {

}
}
```

Dart:

```
class Solution {
int tallestBillboard(List<int> rods) {

}
}
```

Scala:

```
object Solution {
def tallestBillboard(rods: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec tallest_billboard(rods :: [integer]) :: integer
def tallest_billboard(rods) do

end
end
```

Erlang:

```
-spec tallest_billboard(Rods :: [integer()]) -> integer().
tallest_billboard(Rods) ->
.
```

Racket:

```
(define/contract (tallest-billboard rods)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Tallest Billboard
* Difficulty: Hard
* 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 tallestBillboard(vector<int>& rods) {

}
};
```

Java Solution:

```
/**
* Problem: Tallest Billboard
* Difficulty: Hard
* 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 tallestBillboard(int[] rods) {

}
```

```
}
```

Python3 Solution:

```
"""
Problem: Tallest Billboard
Difficulty: Hard
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 tallestBillboard(self, rods: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Tallest Billboard
* Difficulty: Hard
* 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[]} rods
* @return {number}
*/
var tallestBillboard = function(rods) {

};
```

TypeScript Solution:

```
/**
* Problem: Tallest Billboard
* Difficulty: Hard
* 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 tallestBillboard(rods: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Tallest Billboard
* Difficulty: Hard
* 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 TallestBillboard(int[] rods) {

}
}
```

C Solution:

```
/*
* Problem: Tallest Billboard
* Difficulty: Hard
* 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 tallestBillboard(int* rods, int rodsSize) {

}
```

Go Solution:

```
// Problem: Tallest Billboard
// Difficulty: Hard
// 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 tallestBillboard(rods []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun tallestBillboard(rods: IntArray): Int {

}
}
```

Swift Solution:

```
class Solution {
func tallestBillboard(_ rods: [Int]) -> Int {
```

```
}
}
```

Rust Solution:

```
// Problem: Tallest Billboard
// Difficulty: Hard
// 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 tallest_billboard(rods: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} rods
# @return {Integer}
def tallest_billboard(rods)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $rods
* @return Integer
*/
function tallestBillboard($rods) {

}
}
```

Dart Solution:

```
class Solution {
int tallestBillboard(List<int> rods) {

}
}
```

Scala Solution:

```
object Solution {
def tallestBillboard(rods: Array[Int]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec tallest_billboard(rods :: [integer]) :: integer
def tallest_billboard(rods) do

end
end
```

Erlang Solution:

```
-spec tallest_billboard(Rods :: [integer()]) -> integer().
tallest_billboard(Rods) ->
.
```

Racket Solution:

```
(define/contract (tallest-billboard rods)
(-> (listof exact-integer?) exact-integer?)
)
```