Problem 84: Largest Rectangle in Histogram

Problem Information

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

Problem Description

Given an array of integers

heights

representing the histogram's bar height where the width of each bar is

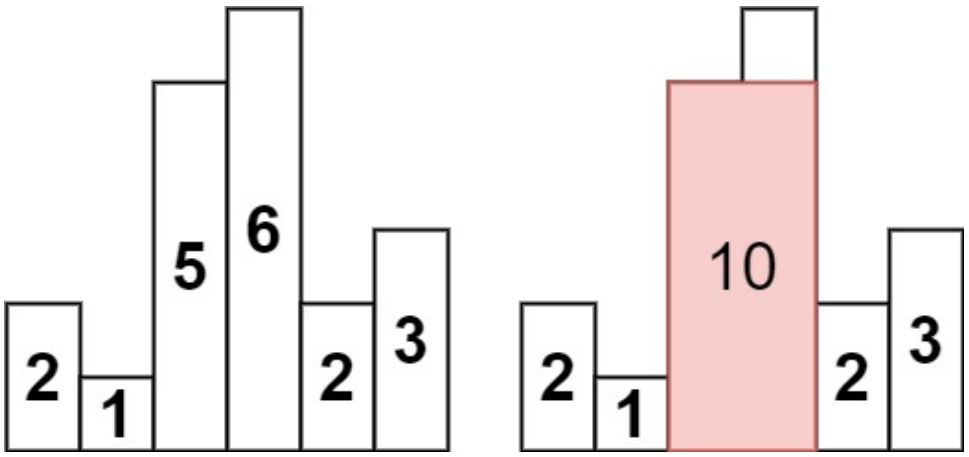
1

, return

the area of the largest rectangle in the histogram

.

Example 1:



Input:

heights = [2,1,5,6,2,3]

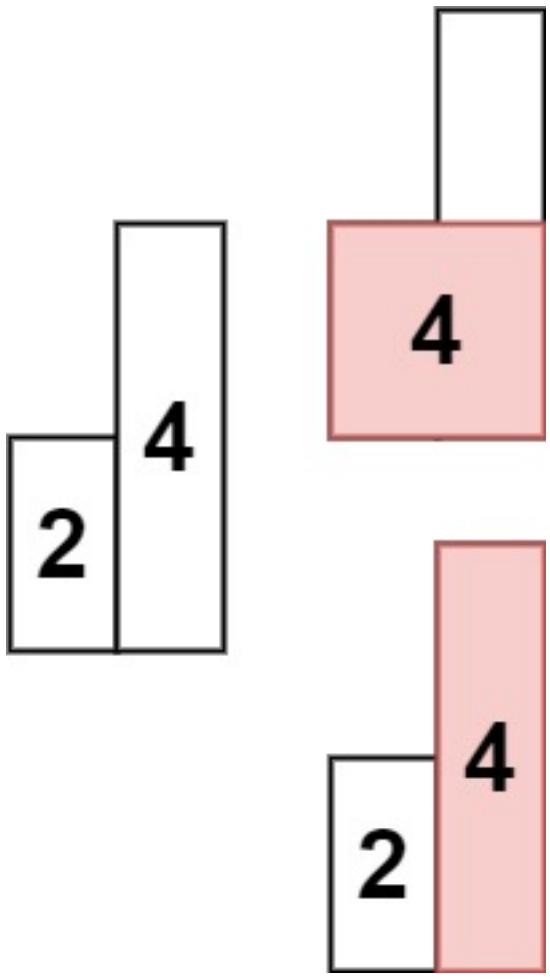
Output:

10

Explanation:

The above is a histogram where width of each bar is 1. The largest rectangle is shown in the red area, which has an area = 10 units.

Example 2:



Input:

heights = [2,4]

Output:

4

Constraints:

1 <= heights.length <= 10

5

0 <= heights[i] <= 10

4

Code Snippets

C++:

```
class Solution {
public:
int largestRectangleArea(vector<int>& heights) {

}
};
```

Java:

```
class Solution {
public int largestRectangleArea(int[] heights) {

}
}
```

Python3:

```
class Solution:
def largestRectangleArea(self, heights: List[int]) -> int:
```

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function largestRectangleArea(heights: number[]): number {

};
```

C#:

```
public class Solution {
public int LargestRectangleArea(int[] heights) {

}
}
```

C:

```
int largestRectangleArea(int* heights, int heightsSize) {

}
```

Go:

```
func largestRectangleArea(heights []int) int {

}
```

Kotlin:

```
class Solution {
fun largestRectangleArea(heights: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func largestRectangleArea(_ heights: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn largest_rectangle_area(heights: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} heights
# @return {Integer}
def largest_rectangle_area(heights)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $heights
* @return Integer
*/
function largestRectangleArea($heights) {

}
```

```
}
```

Dart:

```
class Solution {
int largestRectangleArea(List<int> heights) {

}
}
```

Scala:

```
object Solution {
def largestRectangleArea(heights: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec largest_rectangle_area(heights :: [integer]) :: integer
def largest_rectangle_area(heights) do

end
end
```

Erlang:

```
-spec largest_rectangle_area(Heights :: [integer()]) -> integer().
largest_rectangle_area(Heights) ->
.
```

Racket:

```
(define/contract (largest-rectangle-area heights)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Largest Rectangle in Histogram
* Difficulty: Hard
* Tags: array, stack
*
* 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 largestRectangleArea(vector<int>& heights) {

}
};
```

Java Solution:

```
/**
* Problem: Largest Rectangle in Histogram
* Difficulty: Hard
* Tags: array, stack
*
* 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 largestRectangleArea(int[] heights) {

}
}
```

Python3 Solution:

```
"""
Problem: Largest Rectangle in Histogram
Difficulty: Hard
Tags: array, stack
```

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Largest Rectangle in Histogram
* Difficulty: Hard
* Tags: array, stack
*
* 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[]} heights
* @return {number}
*/
var largestRectangleArea = function(heights) {

};
```

TypeScript Solution:

```
/**
* Problem: Largest Rectangle in Histogram
* Difficulty: Hard
* Tags: array, stack
*
* 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 largestRectangleArea(heights: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Largest Rectangle in Histogram
* Difficulty: Hard
* Tags: array, stack
*
* 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 LargestRectangleArea(int[] heights) {

}
}
```

C Solution:

```
/*
* Problem: Largest Rectangle in Histogram
* Difficulty: Hard
* Tags: array, stack
*
* 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 largestRectangleArea(int* heights, int heightsSize) {

}
```

Go Solution:

```
// Problem: Largest Rectangle in Histogram
// Difficulty: Hard
// Tags: array, stack
//
// 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 largestRectangleArea(heights []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun largestRectangleArea(heights: IntArray): Int {

}
}
```

Swift Solution:

```
class Solution {
func largestRectangleArea(_ heights: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Largest Rectangle in Histogram
// Difficulty: Hard
// Tags: array, stack
```

```
//
// 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 largest_rectangle_area(heights: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} heights
# @return {Integer}
def largest_rectangle_area(heights)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $heights
* @return Integer
*/
function largestRectangleArea($heights) {

}
}
```

Dart Solution:

```
class Solution {
int largestRectangleArea(List<int> heights) {

}
}
```

Scala Solution:

```
object Solution {
def largestRectangleArea(heights: Array[Int]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec largest_rectangle_area(heights :: [integer]) :: integer
def largest_rectangle_area(heights) do

end
end
```

Erlang Solution:

```
-spec largest_rectangle_area(Heights :: [integer()]) -> integer().
largest_rectangle_area(Heights) ->
.
```

Racket Solution:

```
(define/contract (largest-rectangle-area heights)
(-> (listof exact-integer?) exact-integer?)
)
```