

Problem 2866: Beautiful Towers II

Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

You are given a

0-indexed

array

maxHeights

of

n

integers.

You are tasked with building

n

towers in the coordinate line. The

i

th

tower is built at coordinate

i

and has a height of

heights[i]

.

A configuration of towers is

beautiful

if the following conditions hold:

$1 \leq \text{heights}[i] \leq \text{maxHeights}[i]$

heights

is a

mountain

array.

Array

heights

is a

mountain

if there exists an index

i

such that:

For all

$0 < j \leq i$

,

$\text{heights}[j - 1] \leq \text{heights}[j]$

For all

$i \leq k < n - 1$

,

$\text{heights}[k + 1] \leq \text{heights}[k]$

Return

the

maximum possible sum of heights

of a beautiful configuration of towers

.

Example 1:

Input:

$\text{maxHeights} = [5, 3, 4, 1, 1]$

Output:

13

Explanation:

One beautiful configuration with a maximum sum is $\text{heights} = [5, 3, 3, 1, 1]$. This configuration is beautiful since: $-1 \leq \text{heights}[i] \leq \text{maxHeights}[i]$ - heights is a mountain of peak $i = 0$. It can be shown that there exists no other beautiful configuration with a sum of heights greater than

13.

Example 2:

Input:

maxHeights = [6,5,3,9,2,7]

Output:

22

Explanation:

One beautiful configuration with a maximum sum is heights = [3,3,3,9,2,2]. This configuration is beautiful since: - $1 \leq \text{heights}[i] \leq \text{maxHeights}[i]$ - heights is a mountain of peak $i = 3$. It can be shown that there exists no other beautiful configuration with a sum of heights greater than 22.

Example 3:

Input:

maxHeights = [3,2,5,5,2,3]

Output:

18

Explanation:

One beautiful configuration with a maximum sum is heights = [2,2,5,5,2,2]. This configuration is beautiful since: - $1 \leq \text{heights}[i] \leq \text{maxHeights}[i]$ - heights is a mountain of peak $i = 2$. Note that, for this configuration, $i = 3$ can also be considered a peak. It can be shown that there exists no other beautiful configuration with a sum of heights greater than 18.

Constraints:

$1 \leq n == \text{maxHeights.length} \leq 10$

5

$1 \leq \text{maxHeights}[i] \leq 10$

9

Code Snippets

C++:

```
class Solution {  
public:  
    long long maximumSumOfHeights(vector<int>& maxHeights) {  
  
    }  
};
```

Java:

```
class Solution {  
public long maximumSumOfHeights(List<Integer> maxHeights) {  
  
}  
}
```

Python3:

```
class Solution:  
    def maximumSumOfHeights(self, maxHeights: List[int]) -> int:
```

Python:

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

JavaScript:

```
/**  
 * @param {number[]} maxHeights  
 * @return {number}  
 */  
var maximumSumOfHeights = function(maxHeights) {  
  
};
```

TypeScript:

```
function maximumSumOfHeights(maxHeights: number[]): number {  
  
};
```

C#:

```
public class Solution {  
    public long MaximumSumOfHeights(IList<int> maxHeights) {  
  
    }  
}
```

C:

```
long long maximumSumOfHeights(int* maxHeights, int maxHeightsSize) {  
  
}
```

Go:

```
func maximumSumOfHeights(maxHeights []int) int64 {  
  
}
```

Kotlin:

```
class Solution {  
    fun maximumSumOfHeights(maxHeights: List<Int>): Long {  
  
    }  
}
```

Swift:

```
class Solution {  
    func maximumSumOfHeights(_ maxHeights: [Int]) -> Int {  
        }  
    }  
}
```

Rust:

```
impl Solution {  
    pub fn maximum_sum_of_heights(max_heights: Vec<i32>) -> i64 {  
        }  
    }  
}
```

Ruby:

```
# @param {Integer[]} max_heights  
# @return {Integer}  
def maximum_sum_of_heights(max_heights)  
  
end
```

PHP:

```
class Solution {  
  
    /**  
     * @param Integer[] $maxHeights  
     * @return Integer  
     */  
    function maximumSumOfHeights($maxHeights) {  
  
    }  
}
```

Dart:

```
class Solution {  
    int maximumSumOfHeights(List<int> maxHeights) {  
        }  
    }
```

Scala:

```
object Solution {  
    def maximumSumOfHeights(maxHeights: List[Int]): Long = {  
  
    }  
}
```

Elixir:

```
defmodule Solution do  
  @spec maximum_sum_of_heights(max_heights :: [integer]) :: integer  
  def maximum_sum_of_heights(max_heights) do  
  
  end  
end
```

Erlang:

```
-spec maximum_sum_of_heights(MaxHeights :: [integer()]) -> integer().  
maximum_sum_of_heights(MaxHeights) ->  
.
```

Racket:

```
(define/contract (maximum-sum-of-heights maxHeights)  
  (-> (listof exact-integer?) exact-integer?)  
)
```

Solutions

C++ Solution:

```
/*  
 * Problem: Beautiful Towers II  
 * Difficulty: Medium  
 * 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:
    long long maximumSumOfHeights(vector<int>& maxHeights) {
        }
    };

```

Java Solution:

```

/**
 * Problem: Beautiful Towers II
 * Difficulty: Medium
 * 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 long maximumSumOfHeights(List<Integer> maxHeights) {
    }
}

```

Python3 Solution:

```

"""
Problem: Beautiful Towers II
Difficulty: Medium
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 maximumSumOfHeights(self, maxHeights: List[int]) -> int:
        # TODO: Implement optimized solution

```

```
pass
```

Python Solution:

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

JavaScript Solution:

```
/**  
 * Problem: Beautiful Towers II  
 * Difficulty: Medium  
 * 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[]} maxHeights  
 * @return {number}  
 */  
var maximumSumOfHeights = function(maxHeights) {  
  
};
```

TypeScript Solution:

```
/**  
 * Problem: Beautiful Towers II  
 * Difficulty: Medium  
 * 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
```

```
*/\n\nfunction maximumSumOfHeights(maxHeights: number[]): number {\n\n};
```

C# Solution:

```
/*\n * Problem: Beautiful Towers II\n * Difficulty: Medium\n * Tags: array, stack\n *\n * Approach: Use two pointers or sliding window technique\n * Time Complexity: O(n) or O(n log n)\n * Space Complexity: O(1) to O(n) depending on approach\n */\n\npublic class Solution {\n    public long MaximumSumOfHeights(IList<int> maxHeights) {\n\n    }\n}
```

C Solution:

```
/*\n * Problem: Beautiful Towers II\n * Difficulty: Medium\n * Tags: array, stack\n *\n * Approach: Use two pointers or sliding window technique\n * Time Complexity: O(n) or O(n log n)\n * Space Complexity: O(1) to O(n) depending on approach\n */\n\nlong long maximumSumOfHeights(int* maxHeights, int maxHeightsSize) {\n\n}
```

Go Solution:

```

// Problem: Beautiful Towers II
// Difficulty: Medium
// 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 maximumSumOfHeights(maxHeights []int) int64 {
}

```

Kotlin Solution:

```

class Solution {
    fun maximumSumOfHeights(maxHeights: List<Int>): Long {
        return 0
    }
}

```

Swift Solution:

```

class Solution {
    func maximumSumOfHeights(_ maxHeights: [Int]) -> Int {
        return 0
    }
}

```

Rust Solution:

```

// Problem: Beautiful Towers II
// Difficulty: Medium
// 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 maximum_sum_of_heights(max_heights: Vec<i32>) -> i64 {
        return 0
    }
}

```

```
}
```

Ruby Solution:

```
# @param {Integer[]} max_heights
# @return {Integer}
def maximum_sum_of_heights(max_heights)

end
```

PHP Solution:

```
class Solution {

    /**
     * @param Integer[] $maxHeights
     * @return Integer
     */
    function maximumSumOfHeights($maxHeights) {

    }
}
```

Dart Solution:

```
class Solution {
int maximumSumOfHeights(List<int> maxHeights) {

}
```

Scala Solution:

```
object Solution {
def maximumSumOfHeights(maxHeights: List[Int]): Long = {

}
```

Elixir Solution:

```
defmodule Solution do
@spec maximum_sum_of_heights(max_heights :: [integer]) :: integer
def maximum_sum_of_heights(max_heights) do

end
end
```

Erlang Solution:

```
-spec maximum_sum_of_heights(MaxHeights :: [integer()]) -> integer().
maximum_sum_of_heights(MaxHeights) ->
.
```

Racket Solution:

```
(define/contract (maximum-sum-of-heights maxHeights)
(-> (listof exact-integer?) exact-integer?))
```