Problem 1646: Get Maximum in Generated Array

Problem Information

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

Problem Description

You are given an integer

n

. A

0-indexed

integer array

nums

of length

n + 1

is generated in the following way:

nums[0] = 0

nums[1] = 1

nums[2 * i] = nums[i]

when

2 <= 2 * i <= n

nums[2 * i + 1] = nums[i] + nums[i + 1]

when

2 <= 2 * i + 1 <= n

Return

the

maximum

integer in the array

nums

.

Example 1:

Input:

n = 7

Output:

3

Explanation:

According to the given rules: nums[0] = 0 nums[1] = 1 nums[(1 * 2) = 2] = nums[1] = 1 nums[(1 * 2) + 1 = 3] = nums[1] + nums[2] = 1 + 1 = 2 nums[(2 * 2) = 4] = nums[2] = 1 nums[(2 * 2) + 1 = 5] = nums[2] + nums[3] = 1 + 2 = 3 nums[(3 * 2) = 6] = nums[3] = 2 nums[(3 * 2) + 1 = 7] = nums[3] + nums[4] = 2 + 1 = 3 Hence, nums = [0,1,1,2,1,3,2,3], and the maximum is max(0,1,1,2,1,3,2,3) = 3.

Example 2:

Input:

n = 2

Output:

1

Explanation:

According to the given rules, nums = [0,1,1]. The maximum is max(0,1,1) = 1.

Example 3:

Input:

n = 3

Output:

2

Explanation:

According to the given rules, nums = [0,1,1,2]. The maximum is max(0,1,1,2) = 2.

Constraints:

0 <= n <= 100

Code Snippets

C++:

```
class Solution {
public:
int getMaximumGenerated(int n) {
```

```
}
};
```

Java:

```
class Solution {
public int getMaximumGenerated(int n) {

}
}
```

Python3:

```
class Solution:
def getMaximumGenerated(self, n: int) -> int:
```

Python:

```
class Solution(object):
def getMaximumGenerated(self, n):
"""
:type n: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} n
* @return {number}
*/
var getMaximumGenerated = function(n) {

};
```

TypeScript:

```
function getMaximumGenerated(n: number): number {

};
```

C#:

```
public class Solution {
public int GetMaximumGenerated(int n) {

}
}
```

C:

```
int getMaximumGenerated(int n) {

}
```

Go:

```
func getMaximumGenerated(n int) int {

}
```

Kotlin:

```
class Solution {
fun getMaximumGenerated(n: Int): Int {

}
}
```

Swift:

```
class Solution {
func getMaximumGenerated(_ n: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn get_maximum_generated(n: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} n
# @return {Integer}
def get_maximum_generated(n)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @return Integer
*/
function getMaximumGenerated($n) {

}
}
```

Dart:

```
class Solution {
int getMaximumGenerated(int n) {

}
}
```

Scala:

```
object Solution {
def getMaximumGenerated(n: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec get_maximum_generated(n :: integer) :: integer
def get_maximum_generated(n) do

end
end
```

Erlang:

```
-spec get_maximum_generated(N :: integer()) -> integer().
get_maximum_generated(N) ->
.
```

Racket:

```
(define/contract (get-maximum-generated n)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Get Maximum in Generated Array
* Difficulty: Easy
* Tags: array
*
* 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 getMaximumGenerated(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Get Maximum in Generated Array
* Difficulty: Easy
* Tags: array
*
* 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 getMaximumGenerated(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Get Maximum in Generated Array
Difficulty: Easy
Tags: array

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 getMaximumGenerated(self, n: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def getMaximumGenerated(self, n):
"""
:type n: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Get Maximum in Generated Array
* Difficulty: Easy
```

```
* Tags: array
*
* 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} n
* @return {number}
*/
var getMaximumGenerated = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Get Maximum in Generated Array
* Difficulty: Easy
* Tags: array
*
* 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 getMaximumGenerated(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Get Maximum in Generated Array
* Difficulty: Easy
* Tags: array
*
* 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 GetMaximumGenerated(int n) {

}
}
```

C Solution:

```
/*
* Problem: Get Maximum in Generated Array
* Difficulty: Easy
* Tags: array
*
* 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 getMaximumGenerated(int n) {

}
```

Go Solution:

```
// Problem: Get Maximum in Generated Array
// Difficulty: Easy
// Tags: array
//
// 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 getMaximumGenerated(n int) int {

}
```

Kotlin Solution:

```
class Solution {
fun getMaximumGenerated(n: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func getMaximumGenerated(_ n: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Get Maximum in Generated Array
// Difficulty: Easy
// Tags: array
//
// 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 get_maximum_generated(n: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @return {Integer}
def get_maximum_generated(n)

end
```

PHP Solution:

```
class Solution {
```

```
/**
* @param Integer $n
* @return Integer
*/
function getMaximumGenerated($n) {

}
}
```

Dart Solution:

```
class Solution {
int getMaximumGenerated(int n) {

}
}
```

Scala Solution:

```
object Solution {
def getMaximumGenerated(n: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec get_maximum_generated(n :: integer) :: integer
def get_maximum_generated(n) do

end
end
```

Erlang Solution:

```
-spec get_maximum_generated(N :: integer()) -> integer().
get_maximum_generated(N) ->
.
```

Racket Solution:

```
(define/contract (get-maximum-generated n)
(-> exact-integer? exact-integer?)
)
```