Problem 3125: Maximum Number That Makes Result of Bitwise AND Zero

Problem Information

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

Problem Description

Given an integer

n

, return the

maximum

integer

x

such that

x <= n

, and the bitwise

AND

of all the numbers in the range

[x, n]

is 0.

Example 1:

Input:

n = 7

Output:

3

Explanation:

The bitwise

AND

of

[6, 7]

is 6.

The bitwise

AND

of

[5, 6, 7]

is 4.

The bitwise

AND

of

[4, 5, 6, 7]

is 4.

The bitwise

AND

of

[3, 4, 5, 6, 7]

is 0.

Example 2:

Input:

n = 9

Output:

7

Explanation:

The bitwise

AND

of

[7, 8, 9]

is 0.

Example 3:

Input:

n = 17

Output:

15

Explanation:

The bitwise

AND

of

[15, 16, 17]

is 0.

Constraints:

1 <= n <= 10

15

Code Snippets

C++:

```
class Solution {
public:
long long maxNumber(long long n) {

}
};
```

Java:

```
class Solution {
public long maxNumber(long n) {
```

```
}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

```
public class Solution {
public long MaxNumber(long n) {

}
}
```

C:

```
long long maxNumber(long long n) {

}
```

Go:

```
func maxNumber(n int64) int64 {

}
```

Kotlin:

```
class Solution {
fun maxNumber(n: Long): Long {

}
}
```

Swift:

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

}
}
```

Rust:

```
impl Solution {
pub fn max_number(n: i64) -> i64 {

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

```
object Solution {
def maxNumber(n: Long): Long = {

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (max-number n)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Number That Makes Result of Bitwise AND Zero
* Difficulty: Medium
* Tags: string, greedy, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
long long maxNumber(long long n) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Number That Makes Result of Bitwise AND Zero
* Difficulty: Medium
* Tags: string, greedy, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public long maxNumber(long n) {
```

```
}
}
```

Python3 Solution:

```
"""
Problem: Maximum Number That Makes Result of Bitwise AND Zero
Difficulty: Medium
Tags: string, greedy, sort

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def maxNumber(self, n: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Maximum Number That Makes Result of Bitwise AND Zero
* Difficulty: Medium
* Tags: string, greedy, sort
*
* Approach: String manipulation with hash map or two pointers
* 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 maxNumber = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Number That Makes Result of Bitwise AND Zero
* Difficulty: Medium
* Tags: string, greedy, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function maxNumber(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Number That Makes Result of Bitwise AND Zero
* Difficulty: Medium
* Tags: string, greedy, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public long MaxNumber(long n) {

}
```

```
}
```

C Solution:

```
/*
* Problem: Maximum Number That Makes Result of Bitwise AND Zero
* Difficulty: Medium
* Tags: string, greedy, sort
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

long long maxNumber(long long n) {

}
```

Go Solution:

```
// Problem: Maximum Number That Makes Result of Bitwise AND Zero
// Difficulty: Medium
// Tags: string, greedy, sort
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func maxNumber(n int64) int64 {

}
```

Kotlin Solution:

```
class Solution {
fun maxNumber(n: Long): Long {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Maximum Number That Makes Result of Bitwise AND Zero
// Difficulty: Medium
// Tags: string, greedy, sort
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn max_number(n: i64) -> i64 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

```
object Solution {
def maxNumber(n: Long): Long = {

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (max-number n)
(-> exact-integer? exact-integer?)
)
```