Problem 264: Ugly Number II

Problem Information

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

Problem Description

An

ugly number

is a positive integer whose prime factors are limited to

2

,

3

, and

5

.

Given an integer

n

, return

the

n

th

ugly number

.

Example 1:

Input:

n = 10

Output:

12

Explanation:

[1, 2, 3, 4, 5, 6, 8, 9, 10, 12] is the sequence of the first 10 ugly numbers.

Example 2:

Input:

n = 1

Output:

1

Explanation:

1 has no prime factors, therefore all of its prime factors are limited to 2, 3, and 5.

Constraints:

1 <= n <= 1690

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int nthUglyNumber(int n) {

}
```

Go:

```
func nthUglyNumber(n int) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Ugly Number II
* Difficulty: Medium
* Tags: dp, math, hash, queue, heap
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/
```

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

}
};
```

Java Solution:

```
/**
* Problem: Ugly Number II
* Difficulty: Medium
* Tags: dp, math, hash, queue, heap
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int nthUglyNumber(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Ugly Number II
Difficulty: Medium
Tags: dp, math, hash, queue, heap

Approach: Dynamic programming with memoization or tabulation
Time Complexity: O(n * m) where n and m are problem dimensions
Space Complexity: O(n) or O(n * m) for DP table
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Ugly Number II
* Difficulty: Medium
* Tags: dp, math, hash, queue, heap
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Ugly Number II
* Difficulty: Medium
* Tags: dp, math, hash, queue, heap
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

function nthUglyNumber(n: number): number {
};
```

C# Solution:

```
/*
* Problem: Ugly Number II
* Difficulty: Medium
* Tags: dp, math, hash, queue, heap
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public int NthUglyNumber(int n) {

}
}
```

C Solution:

```
/*
* Problem: Ugly Number II
* Difficulty: Medium
* Tags: dp, math, hash, queue, heap
*
* Approach: Dynamic programming with memoization or tabulation
* Time Complexity: O(n * m) where n and m are problem dimensions
* Space Complexity: O(n) or O(n * m) for DP table
*/

int nthUglyNumber(int n) {

}
```

Go Solution:

```
// Problem: Ugly Number II
// Difficulty: Medium
// Tags: dp, math, hash, queue, heap
//
// Approach: Dynamic programming with memoization or tabulation
// Time Complexity: O(n * m) where n and m are problem dimensions
// Space Complexity: O(n) or O(n * m) for DP table

func nthUglyNumber(n int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Ugly Number II
// Difficulty: Medium
// Tags: dp, math, hash, queue, heap
//
// Approach: Dynamic programming with memoization or tabulation
// Time Complexity: O(n * m) where n and m are problem dimensions
// Space Complexity: O(n) or O(n * m) for DP table

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

```
end
end
```

Erlang Solution:

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

Racket Solution:

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