

Problem 1201: Ugly Number III

Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

An

ugly number

is a positive integer that is divisible by

a

,

b

, or

c

.

Given four integers

n

,

a

,

b

, and

c

, return the

n

th

ugly number

.

Example 1:

Input:

n = 3, a = 2, b = 3, c = 5

Output:

4

Explanation:

The ugly numbers are 2, 3, 4, 5, 6, 8, 9, 10... The 3

rd

is 4.

Example 2:

Input:

$n = 4, a = 2, b = 3, c = 4$

Output:

6

Explanation:

The ugly numbers are 2, 3, 4, 6, 8, 9, 10, 12... The 4

th

is 6.

Example 3:

Input:

$n = 5, a = 2, b = 11, c = 13$

Output:

10

Explanation:

The ugly numbers are 2, 4, 6, 8, 10, 11, 12, 13... The 5

th

is 10.

Constraints:

$1 \leq n, a, b, c \leq 10$

9

$1 \leq a * b * c \leq 10$

18

It is guaranteed that the result will be in range

[1, 2 * 10

9

]

.

Code Snippets

C++:

```
class Solution {  
public:  
    int nthUglyNumber(int n, int a, int b, int c) {  
  
    }  
};
```

Java:

```
class Solution {  
    public int nthUglyNumber(int n, int a, int b, int c) {  
  
    }  
}
```

Python3:

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

Python:

```

class Solution(object):
    def nthUglyNumber(self, n, a, b, c):
        """
        :type n: int
        :type a: int
        :type b: int
        :type c: int
        :rtype: int
        """

```

JavaScript:

```

/**
 * @param {number} n
 * @param {number} a
 * @param {number} b
 * @param {number} c
 * @return {number}
 */
var nthUglyNumber = function(n, a, b, c) {

};

```

TypeScript:

```

function nthUglyNumber(n: number, a: number, b: number, c: number): number {

};

```

C#:

```

public class Solution {
    public int NthUglyNumber(int n, int a, int b, int c) {

    }
}

```

C:

```

int nthUglyNumber(int n, int a, int b, int c) {

}

```

Go:

```
func nthUglyNumber(n int, a int, b int, c int) int {  
  
}
```

Kotlin:

```
class Solution {  
    fun nthUglyNumber(n: Int, a: Int, b: Int, c: Int): Int {  
  
    }  
}
```

Swift:

```
class Solution {  
    func nthUglyNumber(_ n: Int, _ a: Int, _ b: Int, _ c: Int) -> Int {  
  
    }  
}
```

Rust:

```
impl Solution {  
    pub fn nth_ugly_number(n: i32, a: i32, b: i32, c: i32) -> i32 {  
  
    }  
}
```

Ruby:

```
# @param {Integer} n  
# @param {Integer} a  
# @param {Integer} b  
# @param {Integer} c  
# @return {Integer}  
def nth_ugly_number(n, a, b, c)  
  
end
```

PHP:

```

class Solution {

    /**
     * @param Integer $n
     * @param Integer $a
     * @param Integer $b
     * @param Integer $c
     * @return Integer
     */
    function nthUglyNumber($n, $a, $b, $c) {

    }

}

```

Dart:

```

class Solution {
  int nthUglyNumber(int n, int a, int b, int c) {

  }

}

```

Scala:

```

object Solution {
  def nthUglyNumber(n: Int, a: Int, b: Int, c: Int): Int = {

  }

}

```

Elixir:

```

defmodule Solution do
  @spec nth_ugly_number(n :: integer, a :: integer, b :: integer, c :: integer)
    :: integer
  def nth_ugly_number(n, a, b, c) do

  end

end

```

Erlang:

```
-spec nth_ugly_number(N :: integer(), A :: integer(), B :: integer(), C ::
integer()) -> integer().
nth_ugly_number(N, A, B, C) ->
.
```

Racket:

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

Solutions

C++ Solution:

```
/*
 * Problem: Ugly Number III
 * Difficulty: Medium
 * Tags: math, search
 *
 * Approach: Optimized algorithm based on problem constraints
 * Time Complexity: O(n) to O(n^2) depending on approach
 * Space Complexity: O(1) to O(n) depending on approach
 */

class Solution {
public:
    int nthUglyNumber(int n, int a, int b, int c) {

    }
};
```

Java Solution:

```
/**
 * Problem: Ugly Number III
 * Difficulty: Medium
 * Tags: math, search
 *
 * Approach: Optimized algorithm based on problem constraints
 */
```

```

* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int nthUglyNumber(int n, int a, int b, int c) {

}
}

```

Python3 Solution:

```

"""
Problem: Ugly Number III
Difficulty: Medium
Tags: math, search

Approach: Optimized algorithm based on problem constraints
Time Complexity: O(n) to O(n^2) depending on approach
Space Complexity: O(1) to O(n) depending on approach
"""

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

```

Python Solution:

```

class Solution(object):
    def nthUglyNumber(self, n, a, b, c):
        """
        :type n: int
        :type a: int
        :type b: int
        :type c: int
        :rtype: int
        """

```

JavaScript Solution:

```

/**
 * Problem: Ugly Number III
 * Difficulty: Medium
 * Tags: math, search
 *
 * Approach: Optimized algorithm based on problem constraints
 * Time Complexity: O(n) to O(n^2) depending on approach
 * Space Complexity: O(1) to O(n) depending on approach
 */

/**
 * @param {number} n
 * @param {number} a
 * @param {number} b
 * @param {number} c
 * @return {number}
 */
var nthUglyNumber = function(n, a, b, c) {

};

```

TypeScript Solution:

```

/**
 * Problem: Ugly Number III
 * Difficulty: Medium
 * Tags: math, search
 *
 * Approach: Optimized algorithm based on problem constraints
 * Time Complexity: O(n) to O(n^2) depending on approach
 * Space Complexity: O(1) to O(n) depending on approach
 */

function nthUglyNumber(n: number, a: number, b: number, c: number): number {

};

```

C# Solution:

```

/*
 * Problem: Ugly Number III
 * Difficulty: Medium

```

```

* Tags: math, search
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int NthUglyNumber(int n, int a, int b, int c) {

}
}

```

C Solution:

```

/*
* Problem: Ugly Number III
* Difficulty: Medium
* Tags: math, search
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

int nthUglyNumber(int n, int a, int b, int c) {

}

```

Go Solution:

```

// Problem: Ugly Number III
// Difficulty: Medium
// Tags: math, search
//
// Approach: Optimized algorithm based on problem constraints
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(1) to O(n) depending on approach

func nthUglyNumber(n int, a int, b int, c int) int {

```

```
}
```

Kotlin Solution:

```
class Solution {  
    fun nthUglyNumber(n: Int, a: Int, b: Int, c: Int): Int {  
  
    }  
}
```

Swift Solution:

```
class Solution {  
    func nthUglyNumber(_ n: Int, _ a: Int, _ b: Int, _ c: Int) -> Int {  
  
    }  
}
```

Rust Solution:

```
// Problem: Ugly Number III  
// Difficulty: Medium  
// Tags: math, search  
//  
// Approach: Optimized algorithm based on problem constraints  
// Time Complexity: O(n) to O(n^2) depending on approach  
// Space Complexity: O(1) to O(n) depending on approach  
  
impl Solution {  
    pub fn nth_ugly_number(n: i32, a: i32, b: i32, c: i32) -> i32 {  
  
    }  
}
```

Ruby Solution:

```
# @param {Integer} n  
# @param {Integer} a  
# @param {Integer} b  
# @param {Integer} c  
# @return {Integer}
```

```
def nth_ugly_number(n, a, b, c)

end
```

PHP Solution:

```
class Solution {

    /**
     * @param Integer $n
     * @param Integer $a
     * @param Integer $b
     * @param Integer $c
     * @return Integer
     */
    function nthUglyNumber($n, $a, $b, $c) {

    }

}
```

Dart Solution:

```
class Solution {
  int nthUglyNumber(int n, int a, int b, int c) {

  }

}
```

Scala Solution:

```
object Solution {
  def nthUglyNumber(n: Int, a: Int, b: Int, c: Int): Int = {

  }

}
```

Elixir Solution:

```
defmodule Solution do
  @spec nth_ugly_number(n :: integer, a :: integer, b :: integer, c :: integer)
    :: integer
end
```

```
def nth_ugly_number(n, a, b, c) do  
  
end  
end
```

Erlang Solution:

```
-spec nth_ugly_number(N :: integer(), A :: integer(), B :: integer(), C ::  
integer()) -> integer().  
nth_ugly_number(N, A, B, C) ->  
.
```

Racket Solution:

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