Problem 1492: The kth Factor of n

Problem Information

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

Problem Description

You are given two positive integers

n

and

k

. A factor of an integer

n

is defined as an integer

i

where

n % i == 0

.

Consider a list of all factors of

n

sorted in

ascending order

, return

the

k

th

factor

in this list or return

-1

if

n

has less than

k

factors.

Example 1:

Input:

n = 12, k = 3

Output:

3

Explanation:

Factors list is [1, 2, 3, 4, 6, 12], the 3

rd

factor is 3.

Example 2:

Input:

n = 7, k = 2

Output:

7

Explanation:

Factors list is [1, 7], the 2

nd

factor is 7.

Example 3:

Input:

n = 4, k = 4

Output:

-1

Explanation:

Factors list is [1, 2, 4], there is only 3 factors. We should return -1.

Constraints:

1 <= k <= n <= 1000

Follow up:

Could you solve this problem in less than O(n) complexity?

Code Snippets

C++:

```
class Solution {
public:
int kthFactor(int n, int k) {

}
};
```

Java:

```
class Solution {
public int kthFactor(int n, int k) {

}
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
* @param {number} n
* @param {number} k
* @return {number}
*/
var kthFactor = function(n, k) {

};
```

TypeScript:

```
function kthFactor(n: number, k: number): number {

};
```

C#:

```
public class Solution {
public int KthFactor(int n, int k) {

}
}
```

C:

```
int kthFactor(int n, int k) {

}
```

Go:

```
func kthFactor(n int, k int) int {

}
```

Kotlin:

```
class Solution {
fun kthFactor(n: Int, k: Int): Int {

}
}
```

Swift:

```
class Solution {
func kthFactor(_ n: Int, _ k: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn kth_factor(n: i32, k: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} n
# @param {Integer} k
# @return {Integer}
def kth_factor(n, k)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @param Integer $k
* @return Integer
*/
function kthFactor($n, $k) {

}
}
```

Dart:

```
class Solution {
int kthFactor(int n, int k) {

}
}
```

Scala:

```
object Solution {
def kthFactor(n: Int, k: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec kth_factor(n :: integer, k :: integer) :: integer
def kth_factor(n, k) do

end
end
```

Erlang:

```
-spec kth_factor(N :: integer(), K :: integer()) -> integer().
kth_factor(N, K) ->
.
```

Racket:

```
(define/contract (kth-factor n k)
(-> exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: The kth Factor of n
* Difficulty: Medium
* Tags: math, sort
*
* 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 kthFactor(int n, int k) {

}
};
```

Java Solution:

```
/**
* Problem: The kth Factor of n
* Difficulty: Medium
* Tags: math, sort
*
* 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 kthFactor(int n, int k) {

}
}
```

Python3 Solution:

```
"""
Problem: The kth Factor of n
Difficulty: Medium
Tags: math, sort

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: The kth Factor of n
* Difficulty: Medium
* Tags: math, sort
*
* 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} k
* @return {number}
*/
var kthFactor = function(n, k) {

};
```

TypeScript Solution:

```
/**
* Problem: The kth Factor of n
* Difficulty: Medium
* Tags: math, sort
*
* 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 kthFactor(n: number, k: number): number {

};
```

C# Solution:

```
/*
* Problem: The kth Factor of n
* Difficulty: Medium
* Tags: math, sort
*
* 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 KthFactor(int n, int k) {

}
}
```

C Solution:

```
/*
* Problem: The kth Factor of n
* Difficulty: Medium
* Tags: math, sort
*
* 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 kthFactor(int n, int k) {

}
```

Go Solution:

```
// Problem: The kth Factor of n
// Difficulty: Medium
// Tags: math, sort
//
// 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 kthFactor(n int, k int) int {

}
```

Kotlin Solution:

```
class Solution {
fun kthFactor(n: Int, k: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func kthFactor(_ n: Int, _ k: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: The kth Factor of n
// Difficulty: Medium
// Tags: math, sort
```

```
//
// 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 kth_factor(n: i32, k: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @param {Integer} k
# @return {Integer}
def kth_factor(n, k)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @param Integer $k
* @return Integer
*/
function kthFactor($n, $k) {

}
}
```

Dart Solution:

```
class Solution {
int kthFactor(int n, int k) {

}
}
```

Scala Solution:

```
object Solution {
def kthFactor(n: Int, k: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec kth_factor(n :: integer, k :: integer) :: integer
def kth_factor(n, k) do

end
end
```

Erlang Solution:

```
-spec kth_factor(N :: integer(), K :: integer()) -> integer().
kth_factor(N, K) ->
.
```

Racket Solution:

```
(define/contract (kth-factor n k)
(-> exact-integer? exact-integer? exact-integer?)
)
```