Problem 2787: Ways to Express an Integer as Sum of Powers

Problem Information

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

Problem Description

Given two

positive

integers

n

and

x

.

Return

the number of ways

n

can be expressed as the sum of the

x

th

power of

unique

positive integers, in other words, the number of sets of unique integers

[n

1

, n

2

, ..., n

k

]

where

n = n

1

x

+ n

2

x

+ ... + n

k

x

.

Since the result can be very large, return it modulo

10

9

+ 7

.

For example, if

n = 160

and

x = 3

, one way to express

n

is

n = 2

3

+ 3

3

+ 5

3

.

Example 1:

Input:

n = 10, x = 2

Output:

1

Explanation:

We can express n as the following: n = 3

2

+ 1

2

= 10. It can be shown that it is the only way to express 10 as the sum of the 2

nd

power of unique integers.

Example 2:

Input:

n = 4, x = 1

Output:

2

Explanation:

We can express n in the following ways: - n = 4

1

= 4. - n = 3

1

+ 1

1

= 4.

Constraints:

1 <= n <= 300

1 <= x <= 5

Code Snippets

C++:

```
class Solution {
public:
int numberOfWays(int n, int x) {

}
};
```

Java:

```
class Solution {
public int numberOfWays(int n, int x) {

}
```

```
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function numberOfWays(n: number, x: number): number {

};
```

C#:

```
public class Solution {
public int NumberOfWays(int n, int x) {

}
}
```

C:

```
int numberOfWays(int n, int x) {

}
```

Go:

```
func numberOfWays(n int, x int) int {

}
```

Kotlin:

```
class Solution {
fun numberOfWays(n: Int, x: Int): Int {

}
}
```

Swift:

```
class Solution {
func numberOfWays(_ n: Int, _ x: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn number_of_ways(n: i32, x: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} n
# @param {Integer} x
# @return {Integer}
def number_of_ways(n, x)
```

```
end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @param Integer $x
* @return Integer
*/
function numberOfWays($n, $x) {

}
}
```

Dart:

```
class Solution {
int numberOfWays(int n, int x) {

}
}
```

Scala:

```
object Solution {
def numberOfWays(n: Int, x: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec number_of_ways(n :: integer, x :: integer) :: integer
def number_of_ways(n, x) do

end
end
```

Erlang:

```
-spec number_of_ways(N :: integer(), X :: integer()) -> integer().
number_of_ways(N, X) ->
.
```

Racket:

```
(define/contract (number-of-ways n x)
(-> exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Ways to Express an Integer as Sum of Powers
* Difficulty: Medium
* Tags: dp
*
* 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 numberOfWays(int n, int x) {

}
};
```

Java Solution:

```
/**
* Problem: Ways to Express an Integer as Sum of Powers
* Difficulty: Medium
* Tags: dp
*
* 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 numberOfWays(int n, int x) {

}
}
```

Python3 Solution:

```
"""
Problem: Ways to Express an Integer as Sum of Powers
Difficulty: Medium
Tags: dp

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Ways to Express an Integer as Sum of Powers
* Difficulty: Medium
* Tags: dp
```

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

};
```

TypeScript Solution:

```
/**
* Problem: Ways to Express an Integer as Sum of Powers
* Difficulty: Medium
* Tags: dp
*
* 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 numberOfWays(n: number, x: number): number {

};
```

C# Solution:

```
/*
* Problem: Ways to Express an Integer as Sum of Powers
* Difficulty: Medium
* Tags: dp
*
* 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 NumberOfWays(int n, int x) {

}
}
```

C Solution:

```
/*
* Problem: Ways to Express an Integer as Sum of Powers
* Difficulty: Medium
* Tags: dp
*
* 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 numberOfWays(int n, int x) {

}
```

Go Solution:

```
// Problem: Ways to Express an Integer as Sum of Powers
// Difficulty: Medium
// Tags: dp
//
// 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 numberOfWays(n int, x int) int {

}
```

Kotlin Solution:

```
class Solution {
fun numberOfWays(n: Int, x: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func numberOfWays(_ n: Int, _ x: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Ways to Express an Integer as Sum of Powers
// Difficulty: Medium
// Tags: dp
//
// 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 number_of_ways(n: i32, x: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @param {Integer} x
# @return {Integer}
def number_of_ways(n, x)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @param Integer $x
* @return Integer
*/
function numberOfWays($n, $x) {

}
}
```

Dart Solution:

```
class Solution {
int numberOfWays(int n, int x) {

}
}
```

Scala Solution:

```
object Solution {
def numberOfWays(n: Int, x: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec number_of_ways(n :: integer, x :: integer) :: integer
def number_of_ways(n, x) do

end
end
```

Erlang Solution:

```
-spec number_of_ways(N :: integer(), X :: integer()) -> integer().
number_of_ways(N, X) ->
.
```

Racket Solution:

```
(define/contract (number-of-ways n x)
(-> exact-integer? exact-integer? exact-integer?)
)
```