Problem 273: Integer to English Words

Problem Information

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

Problem Description

Convert a non-negative integer

num

to its English words representation.

Example 1:

Input:

num = 123

Output:

"One Hundred Twenty Three"

Example 2:

Input:

num = 12345

Output:

"Twelve Thousand Three Hundred Forty Five"

Example 3:

Input:

num = 1234567

Output:

"One Million Two Hundred Thirty Four Thousand Five Hundred Sixty Seven"

Constraints:

0 <= num <= 2

31

- 1

Code Snippets

C++:

```
class Solution {
public:
string numberToWords(int num) {

}
};
```

Java:

```
class Solution {
public String numberToWords(int num) {

}
}
```

Python3:

```
class Solution:
def numberToWords(self, num: int) -> str:
```

Python:

```
class Solution(object):
def numberToWords(self, num):
"""
:type num: int
:rtype: str
"""
```

JavaScript:

```
/**
* @param {number} num
* @return {string}
*/
var numberToWords = function(num) {

};
```

TypeScript:

```
function numberToWords(num: number): string {

};
```

C#:

```
public class Solution {
public string NumberToWords(int num) {

}
}
```

C:

```
char* numberToWords(int num) {

}
```

Go:

```
func numberToWords(num int) string {

}
```

Kotlin:

```
class Solution {
fun numberToWords(num: Int): String {

}
}
```

Swift:

```
class Solution {
func numberToWords(_ num: Int) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn number_to_words(num: i32) -> String {

}
}
```

Ruby:

```
# @param {Integer} num
# @return {String}
def number_to_words(num)

end
```

PHP:

```
class Solution {

/**
* @param Integer $num
* @return String
```

```
*/
function numberToWords($num) {

}
}
```

Dart:

```
class Solution {
String numberToWords(int num) {

}
}
```

Scala:

```
object Solution {
def numberToWords(num: Int): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec number_to_words(num :: integer) :: String.t
def number_to_words(num) do

end
end
```

Erlang:

```
-spec number_to_words(Num :: integer()) -> unicode:unicode_binary().
number_to_words(Num) ->
.
```

Racket:

```
(define/contract (number-to-words num)
(-> exact-integer? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Integer to English Words
* Difficulty: Hard
* Tags: string, math
*
* 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:
string numberToWords(int num) {

}
};
```

Java Solution:

```
/**
* Problem: Integer to English Words
* Difficulty: Hard
* Tags: string, math
*
* 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 String numberToWords(int num) {

}
}
```

Python3 Solution:

```
"""
Problem: Integer to English Words
Difficulty: Hard
Tags: string, math

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 numberToWords(self, num: int) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def numberToWords(self, num):
"""
:type num: int
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Integer to English Words
* Difficulty: Hard
* Tags: string, math
*
* 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} num
* @return {string}
*/
var numberToWords = function(num) {
```

```
};
```

TypeScript Solution:

```
/**
* Problem: Integer to English Words
* Difficulty: Hard
* Tags: string, math
*
* 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 numberToWords(num: number): string {

};
```

C# Solution:

```
/*
* Problem: Integer to English Words
* Difficulty: Hard
* Tags: string, math
*
* 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 string NumberToWords(int num) {

}
}
```

C Solution:

```
/*
* Problem: Integer to English Words
* Difficulty: Hard
```

```
* Tags: string, math
*
* 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
*/

char* numberToWords(int num) {

}
```

Go Solution:

```
// Problem: Integer to English Words
// Difficulty: Hard
// Tags: string, math
//
// 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 numberToWords(num int) string {

}
```

Kotlin Solution:

```
class Solution {
fun numberToWords(num: Int): String {

}
}
```

Swift Solution:

```
class Solution {
func numberToWords(_ num: Int) -> String {

}
}
```

Rust Solution:

```
// Problem: Integer to English Words
// Difficulty: Hard
// Tags: string, math
//
// 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 number_to_words(num: i32) -> String {

}
}
```

Ruby Solution:

```
# @param {Integer} num
# @return {String}
def number_to_words(num)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $num
* @return String
*/
function numberToWords($num) {

}
}
```

Dart Solution:

```
class Solution {
String numberToWords(int num) {
```

```
}
}
```

Scala Solution:

```
object Solution {
def numberToWords(num: Int): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec number_to_words(num :: integer) :: String.t
def number_to_words(num) do

end
end
```

Erlang Solution:

```
-spec number_to_words(Num :: integer()) -> unicode:unicode_binary().
number_to_words(Num) ->
.
```

Racket Solution:

```
(define/contract (number-to-words num)
(-> exact-integer? string?)
)
```