Problem 1220: Count Vowels Permutation

Problem Information

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

Problem Description

Given an integer

n

, your task is to count how many strings of length

n

can be formed under the following rules:

Each character is a lower case vowel (

'a'

,

'e'

,

'i'

,

'o'

,

'u'

)

Each vowel

'a'

may only be followed by an

'e'

.

Each vowel

'e'

may only be followed by an

'a'

or an

'i'

.

Each vowel

'i'

may not

be followed by another

'i'

.

Each vowel

'o'

may only be followed by an

'i'

or a

'u'

.

Each vowel

'u'

may only be followed by an

'a'

.

Since the answer may be too large, return it modulo

10^9 + 7

.

Example 1:

Input:

n = 1

Output:

5

Explanation:

All possible strings are: "a", "e", "i" , "o" and "u".

Example 2:

Input:

n = 2

Output:

10

Explanation:

All possible strings are: "ae", "ea", "ei", "ia", "ie", "io", "iu", "oi", "ou" and "ua".

Example 3:

Input:

n = 5

Output:

68

Constraints:

1 <= n <= 2 * 10^4

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int countVowelPermutation(int n) {

}
```

Go:

```
func countVowelPermutation(n int) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (count-vowel-permutation n)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count Vowels Permutation
* Difficulty: Hard
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
int countVowelPermutation(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Count Vowels Permutation
* Difficulty: Hard
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public int countVowelPermutation(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Vowels Permutation
Difficulty: Hard
Tags: string, dp

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

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

Python Solution:

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

```
"""
```

JavaScript Solution:

```
/**
* Problem: Count Vowels Permutation
* Difficulty: Hard
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Count Vowels Permutation
* Difficulty: Hard
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function countVowelPermutation(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Count Vowels Permutation
* Difficulty: Hard
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

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

}
}
```

C Solution:

```
/*
* Problem: Count Vowels Permutation
* Difficulty: Hard
* Tags: string, dp
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

int countVowelPermutation(int n) {

}
```

Go Solution:

```
// Problem: Count Vowels Permutation
// Difficulty: Hard
// Tags: string, dp
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table
```

```
func countVowelPermutation(n int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Count Vowels Permutation
// Difficulty: Hard
// Tags: string, dp
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

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

}
}
```

Ruby Solution:

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

```
end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (count-vowel-permutation n)
(-> exact-integer? exact-integer?)
)
```