Problem 2575: Find the Divisibility Array of a String

Problem Information

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

Problem Description

You are given a

0-indexed

string

word

of length

n

consisting of digits, and a positive integer

m

.

The

divisibility array

div

of

word

is an integer array of length

n

such that:

div[i] = 1

if the

numeric value

of

word[0,...,i]

is divisible by

m

, or

div[i] = 0

otherwise.

Return

the divisibility array of

word

.

Example 1:

Input:

word = "998244353", m = 3

Output:

[1,1,0,0,0,1,1,0,0]

Explanation:

There are only 4 prefixes that are divisible by 3: "9", "99", "998244", and "9982443".

Example 2:

Input:

word = "1010", m = 10

Output:

[0,1,0,1]

Explanation:

There are only 2 prefixes that are divisible by 10: "10", and "1010".

Constraints:

1 <= n <= 10

5

word.length == n

word

consists of digits from

0

to

9

1 <= m <= 10

9

Code Snippets

C++:

```
class Solution {
public:
vector<int> divisibilityArray(string word, int m) {

}
};
```

Java:

```
class Solution {
public int[] divisibilityArray(String word, int m) {

}
}
```

Python3:

```
class Solution:
def divisibilityArray(self, word: str, m: int) -> List[int]:
```

Python:

```
class Solution(object):
def divisibilityArray(self, word, m):
"""
:type word: str
:type m: int
:rtype: List[int]
```

```
"""
```

JavaScript:

```
/**
* @param {string} word
* @param {number} m
* @return {number[]}
*/
var divisibilityArray = function(word, m) {

};
```

TypeScript:

```
function divisibilityArray(word: string, m: number): number[] {

};
```

C#:

```
public class Solution {
public int[] DivisibilityArray(string word, int m) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* divisibilityArray(char* word, int m, int* returnSize) {

}
```

Go:

```
func divisibilityArray(word string, m int) []int {

}
```

Kotlin:

```
class Solution {
fun divisibilityArray(word: String, m: Int): IntArray {

}
}
```

Swift:

```
class Solution {
func divisibilityArray(_ word: String, _ m: Int) -> [Int] {

}
}
```

Rust:

```
impl Solution {
pub fn divisibility_array(word: String, m: i32) -> Vec<i32> {

}
}
```

Ruby:

```
# @param {String} word
# @param {Integer} m
# @return {Integer[]}
def divisibility_array(word, m)

end
```

PHP:

```
class Solution {

/**
* @param String $word
* @param Integer $m
* @return Integer[]
*/
function divisibilityArray($word, $m) {
```

```
}
}
```

Dart:

```
class Solution {
List<int> divisibilityArray(String word, int m) {

}
}
```

Scala:

```
object Solution {
def divisibilityArray(word: String, m: Int): Array[Int] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec divisibility_array(word :: String.t, m :: integer) :: [integer]
def divisibility_array(word, m) do

end
end
```

Erlang:

```
-spec divisibility_array(Word :: unicode:unicode_binary(), M :: integer()) ->
[integer()].
divisibility_array(Word, M) ->
.
```

Racket:

```
(define/contract (divisibility-array word m)
(-> string? exact-integer? (listof exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the Divisibility Array of a String
* Difficulty: Medium
* Tags: array, string, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
vector<int> divisibilityArray(string word, int m) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Divisibility Array of a String
* Difficulty: Medium
* Tags: array, string, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int[] divisibilityArray(String word, int m) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Divisibility Array of a String
Difficulty: Medium
Tags: array, string, math

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def divisibilityArray(self, word: str, m: int) -> List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def divisibilityArray(self, word, m):
"""
:type word: str
:type m: int
:rtype: List[int]
"""
```

JavaScript Solution:

```
/**
* Problem: Find the Divisibility Array of a String
* Difficulty: Medium
* Tags: array, string, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {string} word
* @param {number} m
* @return {number[]}
*/
```

```
var divisibilityArray = function(word, m) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Divisibility Array of a String
* Difficulty: Medium
* Tags: array, string, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function divisibilityArray(word: string, m: number): number[] {

};
```

C# Solution:

```
/*
* Problem: Find the Divisibility Array of a String
* Difficulty: Medium
* Tags: array, string, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int[] DivisibilityArray(string word, int m) {

}
}
```

C Solution:

```
/*
* Problem: Find the Divisibility Array of a String
* Difficulty: Medium
* Tags: array, string, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* divisibilityArray(char* word, int m, int* returnSize) {

}
```

Go Solution:

```
// Problem: Find the Divisibility Array of a String
// Difficulty: Medium
// Tags: array, string, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func divisibilityArray(word string, m int) []int {

}
```

Kotlin Solution:

```
class Solution {
fun divisibilityArray(word: String, m: Int): IntArray {

}
}
```

Swift Solution:

```
class Solution {
func divisibilityArray(_ word: String, _ m: Int) -> [Int] {

}
}
```

Rust Solution:

```
// Problem: Find the Divisibility Array of a String
// Difficulty: Medium
// Tags: array, string, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn divisibility_array(word: String, m: i32) -> Vec<i32> {

}
}
```

Ruby Solution:

```
# @param {String} word
# @param {Integer} m
# @return {Integer[]}
def divisibility_array(word, m)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $word
* @param Integer $m
* @return Integer[]
*/
function divisibilityArray($word, $m) {
```

```
}
}
```

Dart Solution:

```
class Solution {
List<int> divisibilityArray(String word, int m) {

}
}
```

Scala Solution:

```
object Solution {
def divisibilityArray(word: String, m: Int): Array[Int] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec divisibility_array(word :: String.t, m :: integer) :: [integer]
def divisibility_array(word, m) do

end
end
```

Erlang Solution:

```
-spec divisibility_array(Word :: unicode:unicode_binary(), M :: integer()) ->
[integer()].
divisibility_array(Word, M) ->
.
```

Racket Solution:

```
(define/contract (divisibility-array word m)
(-> string? exact-integer? (listof exact-integer?))
)
```