Problem 2559: Count Vowel Strings in Ranges

Problem Information

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

Problem Description

You are given a

0-indexed

array of strings

words

and a 2D array of integers

queries

.

Each query

queries[i] = [l

i

, r

i

]

asks us to find the number of strings present at the indices ranging from

l

i

to

r

i

(both

inclusive

) of

words

that start and end with a vowel.

Return

an array

ans

of size

queries.length

, where

ans[i]

is the answer to the

i

th

query

.

Note

that the vowel letters are

'a'

,

'e'

,

'i'

,

'o'

, and

'u'

.

Example 1:

Input:

words = ["aba","bcb","ece","aa","e"], queries = [[0,2],[1,4],[1,1]]

Output:

[2,3,0]

Explanation:

The strings starting and ending with a vowel are "aba", "ece", "aa" and "e". The answer to the query [0,2] is 2 (strings "aba" and "ece"). to query [1,4] is 3 (strings "ece", "aa", "e"). to query [1,1] is 0. We return [2,3,0].

Example 2:

Input:

words = ["a","e","i"], queries = [[0,2],[0,1],[2,2]]

Output:

[3,2,1]

Explanation:

Every string satisfies the conditions, so we return [3,2,1].

Constraints:

1 <= words.length <= 10

5

1 <= words[i].length <= 40

words[i]

consists only of lowercase English letters.

sum(words[i].length) <= 3 * 10

5

1 <= queries.length <= 10

5

0 <= l

i

<= r

i

< words.length

Code Snippets

C++:

```
class Solution {
public:
vector<int> vowelStrings(vector<string>& words, vector<vector<int>>& queries)
{

}
};
```

Java:

```
class Solution {
public int[] vowelStrings(String[] words, int[][] queries) {

}
}
```

Python3:

```
class Solution:
def vowelStrings(self, words: List[str], queries: List[List[int]]) ->
List[int]:
```

Python:

```
class Solution(object):
def vowelStrings(self, words, queries):
"""
:type words: List[str]
:type queries: List[List[int]]
:rtype: List[int]
"""
```

JavaScript:

```
/**
* @param {string[]} words
* @param {number[][]} queries
* @return {number[]}
*/
var vowelStrings = function(words, queries) {

};
```

TypeScript:

```
function vowelStrings(words: string[], queries: number[][]): number[] {

};
```

C#:

```
public class Solution {
public int[] VowelStrings(string[] words, int[][] queries) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* vowelStrings(char** words, int wordsSize, int** queries, int
queriesSize, int* queriesColSize, int* returnSize) {

}
```

Go:

```
func vowelStrings(words []string, queries [][]int) []int {

}
```

Kotlin:

```
class Solution {
fun vowelStrings(words: Array<String>, queries: Array<IntArray>): IntArray {

}
}
```

Swift:

```
class Solution {
func vowelStrings(_ words: [String], _ queries: [[Int]]) -> [Int] {

}
}
```

Rust:

```
impl Solution {
pub fn vowel_strings(words: Vec<String>, queries: Vec<Vec<i32>>) -> Vec<i32>
{

}
}
```

Ruby:

```
# @param {String[]} words
# @param {Integer[][]} queries
# @return {Integer[]}
def vowel_strings(words, queries)

end
```

PHP:

```
class Solution {

/**
* @param String[] $words
* @param Integer[][] $queries
* @return Integer[]
*/
function vowelStrings($words, $queries) {

}
}
```

Dart:

```
class Solution {
List<int> vowelStrings(List<String> words, List<List<int>> queries) {

}
}
```

Scala:

```
object Solution {
def vowelStrings(words: Array[String], queries: Array[Array[Int]]):
Array[Int] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec vowel_strings(words :: [String.t], queries :: [[integer]]) :: [integer]
def vowel_strings(words, queries) do

end
end
```

Erlang:

```
-spec vowel_strings(Words :: [unicode:unicode_binary()], Queries ::
[[integer()]]) -> [integer()].
vowel_strings(Words, Queries) ->
```

```
.
```

Racket:

```
(define/contract (vowel-strings words queries)
(-> (listof string?) (listof (listof exact-integer?)) (listof
exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count Vowel Strings in Ranges
* Difficulty: Medium
* Tags: array, string
*
* 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> vowelStrings(vector<string>& words, vector<vector<int>>& queries)
{

}
};
```

Java Solution:

```
/**
* Problem: Count Vowel Strings in Ranges
* Difficulty: Medium
* Tags: array, string
*
* 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[] vowelStrings(String[] words, int[][] queries) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Vowel Strings in Ranges
Difficulty: Medium
Tags: array, string

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 vowelStrings(self, words: List[str], queries: List[List[int]]) ->
List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def vowelStrings(self, words, queries):
"""
:type words: List[str]
:type queries: List[List[int]]
:rtype: List[int]
"""
```

JavaScript Solution:

```
/**
* Problem: Count Vowel Strings in Ranges
```

```
* Difficulty: Medium
* Tags: array, string
*
* 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[]} words
* @param {number[][]} queries
* @return {number[]}
*/
var vowelStrings = function(words, queries) {

};
```

TypeScript Solution:

```
/**
* Problem: Count Vowel Strings in Ranges
* Difficulty: Medium
* Tags: array, string
*
* 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 vowelStrings(words: string[], queries: number[][]): number[] {

};
```

C# Solution:

```
/*
* Problem: Count Vowel Strings in Ranges
* Difficulty: Medium
* Tags: array, string
*
* 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[] VowelStrings(string[] words, int[][] queries) {

}
}
```

C Solution:

```
/*
* Problem: Count Vowel Strings in Ranges
* Difficulty: Medium
* Tags: array, string
*
* 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* vowelStrings(char** words, int wordsSize, int** queries, int
queriesSize, int* queriesColSize, int* returnSize) {

}
```

Go Solution:

```
// Problem: Count Vowel Strings in Ranges
// Difficulty: Medium
// Tags: array, string
//
// 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 vowelStrings(words []string, queries [][]int) []int {
```

```
}
```

Kotlin Solution:

```
class Solution {
fun vowelStrings(words: Array<String>, queries: Array<IntArray>): IntArray {

}
}
```

Swift Solution:

```
class Solution {
func vowelStrings(_ words: [String], _ queries: [[Int]]) -> [Int] {

}
}
```

Rust Solution:

```
// Problem: Count Vowel Strings in Ranges
// Difficulty: Medium
// Tags: array, string
//
// 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 vowel_strings(words: Vec<String>, queries: Vec<Vec<i32>>) -> Vec<i32>
{

}
}
```

Ruby Solution:

```
# @param {String[]} words
# @param {Integer[][]} queries
# @return {Integer[]}
```

```
def vowel_strings(words, queries)

end
```

PHP Solution:

```
class Solution {

/**
* @param String[] $words
* @param Integer[][] $queries
* @return Integer[]
*/
function vowelStrings($words, $queries) {

}
}
```

Dart Solution:

```
class Solution {
List<int> vowelStrings(List<String> words, List<List<int>> queries) {

}
}
```

Scala Solution:

```
object Solution {
def vowelStrings(words: Array[String], queries: Array[Array[Int]]):
Array[Int] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec vowel_strings(words :: [String.t], queries :: [[integer]]) :: [integer]
def vowel_strings(words, queries) do
```

```
end
end
```

Erlang Solution:

```
-spec vowel_strings(Words :: [unicode:unicode_binary()], Queries ::
[[integer()]]) -> [integer()].
vowel_strings(Words, Queries) ->
.
```

Racket Solution:

```
(define/contract (vowel-strings words queries)
(-> (listof string?) (listof (listof exact-integer?)) (listof
exact-integer?))
)
```