Problem 345: Reverse Vowels of a String

Problem Information

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

Problem Description

Given a string

s

, reverse only all the vowels in the string and return it.

The vowels are

'a'

,

'e'

,

'i'

,

'o'

, and

'u'

, and they can appear in both lower and upper cases, more than once.

Example 1:

Input:

s = "IceCreAm"

Output:

"AceCreIm"

Explanation:

The vowels in

s

are

['I', 'e', 'e', 'A']

. On reversing the vowels, s becomes

"AceCreIm"

.

Example 2:

Input:

s = "leetcode"

Output:

"leotcede"

Constraints:

1 <= s.length <= 3 * 10

5

s

consist of

printable ASCII

characters.

Code Snippets

C++:

```
class Solution {
public:
string reverseVowels(string s) {

}
};
```

Java:

```
class Solution {
public String reverseVowels(String s) {

}
}
```

Python3:

```
class Solution:
def reverseVowels(self, s: str) -> str:
```

Python:

```
class Solution(object):
def reverseVowels(self, s):
```

```
"""
:type s: str
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} s
* @return {string}
*/
var reverseVowels = function(s) {

};
```

TypeScript:

```
function reverseVowels(s: string): string {

};
```

C#:

```
public class Solution {
public string ReverseVowels(string s) {

}
}
```

C:

```
char* reverseVowels(char* s) {

}
```

Go:

```
func reverseVowels(s string) string {

}
```

Kotlin:

```
class Solution {
fun reverseVowels(s: String): String {

}
}
```

Swift:

```
class Solution {
func reverseVowels(_ s: String) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn reverse_vowels(s: String) -> String {

}
}
```

Ruby:

```
# @param {String} s
# @return {String}
def reverse_vowels(s)

end
```

PHP:

```
class Solution {

/**
* @param String $s
* @return String
*/
function reverseVowels($s) {

}
}
```

Dart:

```
class Solution {
String reverseVowels(String s) {

}
}
```

Scala:

```
object Solution {
def reverseVowels(s: String): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec reverse_vowels(s :: String.t) :: String.t
def reverse_vowels(s) do

end
end
```

Erlang:

```
-spec reverse_vowels(S :: unicode:unicode_binary()) ->
unicode:unicode_binary().
reverse_vowels(S) ->
.
```

Racket:

```
(define/contract (reverse-vowels s)
(-> string? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Reverse Vowels of a String
* Difficulty: Easy
* 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:
string reverseVowels(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Reverse Vowels of a String
* Difficulty: Easy
* 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 String reverseVowels(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Reverse Vowels of a String
Difficulty: Easy
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 reverseVowels(self, s: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def reverseVowels(self, s):
"""
:type s: str
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Reverse Vowels of a String
* Difficulty: Easy
* 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} s
* @return {string}
*/
var reverseVowels = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Reverse Vowels of a String
* Difficulty: Easy
* 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 reverseVowels(s: string): string {

};
```

C# Solution:

```
/*
* Problem: Reverse Vowels of a String
* Difficulty: Easy
* 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 string ReverseVowels(string s) {

}
}
```

C Solution:

```
/*
* Problem: Reverse Vowels of a String
* Difficulty: Easy
* 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
```

```
*/

char* reverseVowels(char* s) {

}
```

Go Solution:

```
// Problem: Reverse Vowels of a String
// Difficulty: Easy
// 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 reverseVowels(s string) string {

}
```

Kotlin Solution:

```
class Solution {
fun reverseVowels(s: String): String {

}
}
```

Swift Solution:

```
class Solution {
func reverseVowels(_ s: String) -> String {

}
}
```

Rust Solution:

```
// Problem: Reverse Vowels of a String
// Difficulty: Easy
// 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 reverse_vowels(s: String) -> String {

}
}
```

Ruby Solution:

```
# @param {String} s
# @return {String}
def reverse_vowels(s)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $s
* @return String
*/
function reverseVowels($s) {

}
}
```

Dart Solution:

```
class Solution {
String reverseVowels(String s) {

}
}
```

Scala Solution:

```
object Solution {
def reverseVowels(s: String): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec reverse_vowels(s :: String.t) :: String.t
def reverse_vowels(s) do

end
end
```

Erlang Solution:

```
-spec reverse_vowels(S :: unicode:unicode_binary()) ->
unicode:unicode_binary().
reverse_vowels(S) ->
.
```

Racket Solution:

```
(define/contract (reverse-vowels s)
(-> string? string?)
)
```