Problem 2785: Sort Vowels in a String

Problem Information

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

Problem Description

Given a

0-indexed

string

s

,

permute

s

to get a new string

t

such that:

All consonants remain in their original places. More formally, if there is an index

i

with

0 <= i < s.length

such that

s[i]

is a consonant, then

t[i] = s[i]

.

The vowels must be sorted in the

nondecreasing

order of their

ASCII

values. More formally, for pairs of indices

i

,

j

with

0 <= i < j < s.length

such that

s[i]

and

s[j]

are vowels, then

t[i]

must not have a higher ASCII value than

t[j]

.

Return

the resulting string

.

The vowels are

'a'

,

'e'

,

'i'

,

'o'

, and

'u'

, and they can appear in lowercase or uppercase. Consonants comprise all letters that are not vowels.

Example 1:

Input:

s = "lEetcOde"

Output:

"lEOtcede"

Explanation:

'E', 'O', and 'e' are the vowels in s; 'l', 't', 'c', and 'd' are all consonants. The vowels are sorted according to their ASCII values, and the consonants remain in the same places.

Example 2:

Input:

s = "lYmpH"

Output:

"lYmpH"

Explanation:

There are no vowels in s (all characters in s are consonants), so we return "lYmpH".

Constraints:

1 <= s.length <= 10

5

s

consists only of letters of the English alphabet in uppercase and lowercase

.

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
char* sortVowels(char* s) {

}
```

Go:

```
func sortVowels(s string) string {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Sort Vowels in a String
* Difficulty: Medium
* Tags: string, sort
*
* 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 sortVowels(string s) {

}
};
```

Java Solution:

```
/**
* Problem: Sort Vowels in a String
* Difficulty: Medium
* Tags: string, sort
*
* 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 sortVowels(String s) {

}
}
```

Python3 Solution:

```
"""
Problem: Sort Vowels in a String
Difficulty: Medium
Tags: string, sort

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 sortVowels(self, s: str) -> str:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Sort Vowels in a String
* Difficulty: Medium
* Tags: string, sort
*
* 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 {string} s
* @return {string}
*/
var sortVowels = function(s) {

};
```

TypeScript Solution:

```
/**
* Problem: Sort Vowels in a String
* Difficulty: Medium
* Tags: string, sort
*
* 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 sortVowels(s: string): string {

};
```

C# Solution:

```
/*
* Problem: Sort Vowels in a String
* Difficulty: Medium
* Tags: string, sort
*
* 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 SortVowels(string s) {

}
}
```

C Solution:

```
/*
* Problem: Sort Vowels in a String
* Difficulty: Medium
* Tags: string, sort
*
* 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* sortVowels(char* s) {

}
```

Go Solution:

```
// Problem: Sort Vowels in a String
// Difficulty: Medium
// Tags: string, sort
//
// 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 sortVowels(s string) string {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Sort Vowels in a String
// Difficulty: Medium
// Tags: string, sort
//
// 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 sort_vowels(s: String) -> String {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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